1C:ENTERPRISE 8.3

Developer Guide

Moscow
1C Company
2019

THE COPYRIGHT FOR
SOFTWARE AND DOCUMENTATION REPLICATION
BELONGS TO "1C" COMPANY

By purchasing "1C: Entreprise" system you agree to avoid software and documentation copying without "1C" company written approval.

© 1C-Soft LLC, 1996 - 2018

1C Company, Moscow, 123056, P.O. box 64, Russia
Sales department: Seleznevskaya st., 21
Phone: +7 (495) 737-92-57
Fax: +7 (495) 681-44-07,
Email: int@1c.com
URL: https://1c-dn.com

Product development group- A. Abasov, A. Akimov, R. Aleynikov, A. Alekseev, V. Andryuschenko, Ya. Batura, M. Begletsov, A. Bezborodov, A. Belyak, D. Beskorovainov, E. Bobrova, A. Bushnev, P. Vasilets, A. Vinogradov, Ya. Virkovsky, A. Volkov, I. Golshtein, E. Gornostaev, N. Grebnev, A. Gudnev, S. Guriev, I. Gusarov, G. Damie, A. Darovskikh, O. Derut, M. Dzjuba, I. Djuplischev, N. Evgrafov, B. Evtifeev, A. Zabelinsky, D. Zadorin, I. Zapletnev, D. Zaretsky, D. Ivashov, A. Kaganovich, M. Kamnev, K. Karmakulov, E. Kiryakov, A. Kovalev, Ya. Kovalev, I. Kovalenko, A. Kozhevnikov, S. Kopienko, N. Korsakov, S. Kravchenko, V. Kudryavtsev, P. Kukushkin, A. Kulinich, A. Kunchenko, V. Kupriyanov, R. Kuskov, A. Lakutin, M. Leibovich, G. Leontiev, A. Lekhan, A. Makeev, S. Malachiev, A. Malyshenok, S. Martynenko, A. Matveev, A. Machnev, A. Medvedev, D. Mezhuev, S. Melnikov, E. Mitroshkin, A. Moiseev, S. Murzin, M. Mukhin, A. Nasibullin, A. Nuraliev, S. Nuraliev, S. Olenchuk, L. Onuchin, I. Orlov, M. Otstavnov, D. Pavlenko, I. Pivkin, V. Piskarev, A. Plyakin, P. Romanov, A. Rukin, D. Rusanov, M. Sablin, E. Silin, S. Sitnikov, D. Sluzhbin, A. Smirnov, E. Smirnov, Y. Smirnov, A. Sobolev, V. Sokolov, P. Solodky, A. Solyanik, V. Sosnovsky, E. Storozhenko, G. Suaridze, S. Suvorov, D. Sysoenkov, S. Sychev, D. Tishkov, A. Toporkov, A. Tretyakevich, A. Trofimchuk, A. Trubkin, V. Tunegov, A. Tyushkin, V. Filippov, A. Khasanov, T. Khusaenov, A. Tsilyabin, V. Cheskis, V. Cheremisinov, P. Chikov, A. Chicherin, A. Chkadua, P. Churbanov, S. Shvets, A. Shevchenko, M. Shirokov, V. Shulga, A. Scherbinin, A. Edemsky.

-
-

-

Book title:	1C:Enterprise 8.3. Developer Guide.
Edition number:	83.002.16.03
Publication date:	05 November 2019

TECHNICAL SUPPORT LINE

When you dial the hot line, ensure that you are not far from your computer and you have this guide and your registration card with you. Be prepared to provide the support representative with the brand and technical specifications of your computer and printer.

When you dial the hot line, you will be connected with a technical specialist. Be ready to provide the name of your company, your software version number (it can be found on the software distribution CD and on your registration card) and other registration information. The information that you provide will be verified against the registration form that you sent to 1C Company.

The technical support specialist might attempt to reproduce your situation on their computer. They might provide the solution immediately or consult software developers. Do not worry about contacting a specific expert: we guarantee that all our support specialists are highly trained. The log of all support calls is maintained, so when calling about a previous issue you can refer to the date and time of your previous call.

WE ARE ALWAYS HERE TO HELP YOU!

Contents

Introduction 31

Overview ... 31
What you need to know ... 32
Books included in the documentation ... 33
Update and migration notes delivered with 1C:Enterprise ... 34
Agreed notations ... 34

Chapter 1. General concepts ... 37

1.1. Configurability ... 37
1.2. Operations ... 37
1.3. Basic concepts ... 38
1.3.1. The concept of "configuration" ... 38
1.3.2. Configuration objects ... 39
1.3.3. Command interface ... 48
1.3.4. Forms ... 49
1.3.5. Modules ... 49
1.3.6. Templates ... 51
1.4. Operating modes, interaction modes, client application types ... 52
1.4.1. Application components ... 52
1.4.2. Operating modes ... 52
1.4.3. Types of component interactions ... 54
1.4.4. Types of client applications ... 54
1.4.5. Interface ... 55
1.4.6. Ordinary applications ... 55
1.4.7. Summary ... 56
1.5. Technological development tools ... 56

Chapter 2. Managing configurations ... 60

2.1. General information ... 60
2.2. Opening configurations ... 61
2.3. Saving configurations ... 61
2.4. Closing configurations ... 62
2.5. Saving configurations to file ... 62
2.6. Loading configurations from file ... 62
2.7. Configuration object tree ... 62
2.8. Metadata tree sorting settings ... 64
2.9. Creating and deleting configuration objects ... 64
2.9.1. Creating configuration objects ... 64
2.9.2. Deleting configuration objects ... 66
2.9.3. References to configuration objects ... 66
2.10. Property palette ... 67
2.11. More window ... 71

2.12. Object editing window 74
2.13. Creating Help Section 78
2.14. Operating with Database Configuration 80
2.14.1. Database Configuration Object Tree 80
2.14.2. Updating Database Configuration 80
2.14.3. Background database configuration update 85
2.14.4. Saving database configuration to file 89
2.14.5. Comparison of configuration and database configuration 89
2.14.6. Cancel configuration changes 89
2.15. Starting 1C:Enterprise 90
2.16. Unloading and Downoading Configuration Files 91
2.17. Dumping configurations to files Restoring configurations from files 92
2.17.1. General information 92
2.17.2. Unload format versions 93
2.17.3. Unloading Formats 93
2.17.4. Unloading 99
2.17.5. Loading 100
2.18. Configuration extensions 103
2.19. Mobile Application Operation 103
2.20. Mobile Client Operation 103
2.21. Configuration Report 103
2.22. Using Global Search and Replacement 104
2.23. Customizing the work area of Designer 108
2.23.1. Panel customization 108
2.23.2. Configuration Window 109
2.23.3. Using Window Display Modes 109
2.24. Saving to a configuration ID file 110
2.25. Operating with configuration using the 1C:Enterprise language 110

Chapter 3. Application interface 112

3.1. General information 112
3.2. Opening forms in separate windows 114
3.2.1. General information 114
3.2.2. Application Main Window 115
3.2.3. Auxiliary windows 119
3.3. Opening forms in tabs 120
3.3.1. General information 120
3.3.2. Application Main Window 121
3.3.3. Auxiliary windows 127
3.4. Taxi interface 127
3.4.1. General information 127
3.4.2. Application Main Window 128
3.4.3. Auxiliary windows 146
3.4.4. Nonvisual access to interface 149

Chapter 4. 1C:Enterprise language 153

4.1. Purpose ans brief description of the 1C:Enterprise language 153
4.2. Format of the modules source 154

4.2.1. What is module? 154
4.2.2. Module execution context 154
4.2.3. Module types 155
4.2.4. Module format 159
4.2.5. Special characters used in source code 162

4.3. Primitive data types 163
4.4. Assignment statement 166
4.5. Language expressions 167

4.5.1. Arithmetic operations 167
4.5.2. Concatenation 168
4.5.3. Logical operations 168
4.5.4. Priorities for evaluating 1C:Enterprise language expressions 170

4.6. Statements and syntax structures 171
4.7. Working with queries 191

4.7.1. Using object properties 191
4.7.2. Extending the context of objects and forms 192
4.7.3. Passing procedure and function parameters 193
4.7.4. Serialization/deserialization 201
4.7.5. Using value collections 202
4.7.6. Using numbers and indexes 203
4.7.7. Using system enumerations 203
4.7.8. Using predefined values 203
4.7.9. Synchronous and asynchronous operations 205

4.8. Differences between various system startup options 209

4.8.1. Procedures and functions execution 209
4.8.2. Annotations 217
4.8.3. Specific features of using objects, their properties and methods 217

Chapter 5. Configuration objects 221

5.1. Configuration properties 221

5.1.1. Specifying general properties 221
5.1.2. "Presentation" category 223
5.1.3. "Development" properties category 227
5.1.4. Properties related to development for mobile devices 227
5.1.5. "Help" properties category 230
5.1.6. "Compatibility" properties category 230

5.2. Managed application module 233
5.3. External connection module 234
5.4. Session module 234
5.5. "Common" configuration branch 234

5.5.1. Subsystems 234
5.5.2. Common modules 236
5.5.3. Session parameters 242
5.5.4. Roles and access rights 243

5.5.5. Common attributes ... 270
5.5.6. Exchange plans ... 273
5.5.7. Filter criteria ... 274
5.5.8. Event subscriptions ... 275
5.5.9. Scheduled jobs ... 276
5.5.10. Functional options and functional option parameters ... 278
5.5.11. Defined types ... 290
5.5.12. Settings storages ... 293
5.5.13. Common forms ... 300
5.5.14. Commom commands ... 301
5.5.15. Command groups ... 301
5.5.16. Common templates ... 301
5.5.17. Common pictures ... 301
5.5.18. XDTO packages ... 307
5.5.19. Web services ... 317
5.5.20. HTTP services ... 320
5.5.21. WS references ... 320
5.5.22. Styles and style elements ... 322
5.5.23. Languages ... 325

5.6. Common configuration object properties ... 327

5.6.1. General properties ... 327
5.6.2. Metadata objects presentation ... 328
5.6.3. Arbitrary data presentation ... 329
5.6.4. Standard attributes ... 332
5.6.5. Predefined data ... 333
5.6.6. Objects numeration ... 338
5.6.7. Input by string ... 342
5.6.8. Forms ... 350
5.6.9. Command ... 352
5.6.10. New object attributes entering mechanism ... 353
5.6.11. Attributes fill checking ... 355
5.6.12. Indexing object attributes ... 360
5.6.13. Rights ... 361
5.6.14. Quick choice ... 361
5.6.15. Choice history during the input ... 362
5.6.16. Creating new item on input ... 363
5.6.17. Choice Parameter Links ... 365
5.6.18. Choice parameters ... 368
5.6.19. Miscellaneous ... 369

5.7. Constants ... 370

5.7.1. General information ... 370
5.7.2. Constant properties ... 370

5.8. Catalogs ... 371

5.8.1. General information ... 371
5.8.2. Catalog properties ... 372
5.8.3. Catalog attributes properties ... 376
5.8.4. Predefined catalog items ... 376

5.9. Documents ... 377

5.9.1. General information ... 377
5.9.2. Document properties ... 378
5.9.3. Document posting mechanism ... 381
5.9.4. Numerators ... 384
5.9.5. Document sequences ... 385

5.9.6. Document generation ... 389
5.10. Document journals ... 390
5.10.1. Journal creation ... 390
5.10.2. Journal editing ... 390
5.10.3. Document journal columns ... 391
5.10.4. Interactive creation of new document ... 392
5.11. Enumerations ... 393
5.12. Reports and data processors ... 395
5.12.1. General information ... 395
5.12.2. Externaldata processors and reports ... 396
5.13. Charts of characteristic types ... 403
5.13.1. General description ... 403
5.13.2. Configuration object characteristics ... 405
5.13.3. Extra dimension of analytical accounting ... 419
5.13.4. Using characteristics in application ... 420
5.14. Registers ... 422
5.14.1. General information ... 422
5.14.2. Information registers ... 423
5.14.3. Accumulation registers ... 429

Chapter 6. Command interface ... 445

6.1. General command interface structure ... 445
6.1.1. Sections and subsections of the main application window ... 445
6.1.2. Types of commands ... 445
6.1.3. Command groups ... 450
6.2. Building global command interface ... 451
6.2.1. Subsystems ... 452
6.2.2. Command ... 453
6.2.3. Command parametization ... 454
6.2.4. Creating command interface by default ... 455
6.2.5. Command interface property ... 456
6.2.6. Editing command set ... 456
6.2.7. Role-based settings of command visibility by default ... 456
6.3. Service options for navigation ... 458
6.3.1. Using links ... 458
6.3.2. User Notifications ... 459
6.3.3. Displaying the status of time-consuming processes ... 461
6.3.4. Messages ... 463
6.3.5. Working with keyboard shortcuts ... 469
6.3.6. Working with the List of Forms ... 469
6.3.7. Using the Jump List ... 470
6.3.8. Installing preview window buttons for the Windows taskbar ... 471
6.3.9. Taskbar Icon Overlay Management ... 472
6.4. The Command Interface Development Order ... 474

Chapter 7. Forms 477
7.1. General information 477
7.2. Form Attributes 478
7.2.1. General information 478
7.2.2. The Form Data Types 479
7.2.3. Converting the applied objects data to form data and back 484
7.2.4. Properties of Attributes 486
7.3. Special types of form attributes 487
7.3.1. Dynamic list 487
7.3.2. Formatted Document 507
7.3.3. Planner 511
7.3.4. Charts 520
7.4. Form parameters 558
7.4.1. General information 558
7.4.2. The standard parameters of a form 560
7.4.3. Example of working with the form parameters 560
7.5. Form commands 563
7.5.1. General information 563
7.5.2. Features of using various commands 564
7.6. Form module 566
7.7. Form items 568
7.7.1. General information 568
7.7.2. Overall properties of the form items 569
7.7.3. Forms 574
7.7.4. Field 581
7.7.5. Decoration 587
7.7.6. Tables 588
7.7.7. Button 593
7.7.8. Group 595
7.7.9. Adding a form item 599
7.7.10. Special command bars 601
7.7.11. A conditional design of a form 603
7.7.12. Features of the form items conduct 605
7.7.13. Rules for placing form elements 607
7.8. Form command interface 621
7.9. Working with a form from the 1C:Enterprise language 622
7.9.1. Opening a form 622
7.9.2. Redefinition of the form being opened 623
7.9.3. Modifying properties of the form items 624
7.9.4. Form's modification 625
7.9.5. Working with a dynamic list 630
7.9.6. Transferring mechanism 639
7.9.7. Different approaches to modality 642

Chapter 8. Working with queries 649
8.1. General information 649
8.2. The sources of query data (table) 649
8.3. Virtual and standard fields 650

8.4. Query language ... 651
8.4.1. General information ... 651
8.4.2. Syntax diagram of a query language ... 651
8.4.3. Comments in query language ... 652
8.4.4. Using predefined configuration data ... 652
8.4.5. Bilingual presentation of keywords ... 653
8.4.6. Main sections of the query text ... 656
8.4.7. Description of a query ... 656
8.4.8. Description of sampling fields ... 660
8.4.9. Descripting sources of query ... 664
8.4.10. Filtration query results ... 672
8.4.11. Query results grouping ... 673
8.4.12. Conditions for aggregate function values ... 678
8.4.13. Query merging ... 678
8.4.14. Query results ordering ... 680
8.4.15. Query totals calculation ... 684
8.4.16. Expressions in query language ... 691
8.4.17. Conditions in query language ... 711
8.5. Execution and working with queries in the 1C:Enterprise language ... 721
8.5.1. Working with queries ... 721
8.5.2. Managing temporary tables ... 730
8.5.3. Working with batch queries ... 735
8.5.4. Software operations with queries ... 737

Chapter 9. Data operations ... 741

9.1. Object locks function ... 741
9.1.1. General information ... 741
9.1.2. Pessimistic lock ... 741
9.1.3. Pessimistic lock and transactions ... 744
9.1.4. Optimistic lock ... 744
9.2. Transaction function ... 744
9.2.1. General information ... 744
9.2.2. Using explicit transaction call ... 745
9.2.3. Nested transaction call ... 747
9.2.4. The impact of transactions on the software objects operation ... 748
9.3. Managed lock function ... 749
9.3.1. General information ... 749
9.3.2. Managed locks ... 750
9.3.3. Setting the lock mode in the configuration ... 752
9.3.4. Operations with managed locks using 1C:Enterprise language tools ... 753
9.3.5. Features of "Automatic and managed" mode ... 758
9.3.6. Features of using a large number of managed locks ... 759
9.3.7. Modifying configurations during the transition to the managed lock mode ... 760
9.4. Exclusive mode ... 767

Chapter 10. Data composition system 769

10.1. General data composition information 769
10.2. General data composition system objects 773

10.2.1."Use" property 773
10.2.2.Data composition system field 774
10.2.3.Data composition system parameters 774

10.3. Data composition schema 774

10.3.1.Data composition scheme components 775
10.3.2.Using multiple data sets 780
10.3.3.Query language extension for data composition system 784
10.3.4.Data composition system expression language 787
10.3.5.Data composition schema wizard 833
10.3.6.Data composition schema option settings 856
10.3.7.User Settings of Data Composition System 863

10.4. Data Composition Template 871

10.4.1.Data composition template components 872
10.4.2.Areas Templates 875
10.4.3.Appearance Template 878
10.4.4.Templates Areas Generator 880
10.4.5.Generate Data Composition Template 887

10.5. Data Composition Processor 889
10.6. Functional options and rights to view the field in the report 890
10.7. Data Composition Result 892

10.7.1.Outputting composition results to a spreadsheet document 894
10.7.2.Data composition result output to a table or value tree 895

10.8. Calculation of totals by balance fields in the data composition system 897

10.8.1.Calculation of totals by balance fields 897
10.8.2.Calculation of totals by book balances fields 898
10.8.3.Template Composition 899

10.9. Operation with a hierarchy in the data composition system 899

10.9.1.Hierarchical groupings 899
10.9.2.Hierarchical detailed records 900
10.9.3.Display one item in multiple parent records 901
10.9.4.Condition IN HIERARCHY 903

10.10.Using the data composition system to develop application solutions 903

10.10.1. Input parameters 907
10.10.2. Using metadata object properties in reports 907
10.10.3. Background Report Execution 907
10.10.4. Using a single color for a series 909

10.11.Features of using the data composition system 910

Chapter 11. Accounting 915

11.1. Overview 915
11.2. Charts of accounts 915

11.3. Analytical Accounting 917
11.4. Accounting Types 919
11.5. Records and record sets of accounting registers 920
11.6. Maintain Consolidated Accounting 920
11.7. Create Chart of Accounts 921
11.7.1. Properties of chart of accounts 921
11.7.2. Organizing of different types of accounting 923
11.7.3. Creating a list of accounts (predefined accounts) 923
11.8. Accounting registers 925
11.8.1. Accounting Register properties 925
11.8.2. Accounting Register records 926

Chapter 12. Periodic Calculations 929

12.1. General information 929
12.2. Basic concepts 929
12.3. Charts of calculation types 931
12.3.1. Specifying calculation properties 932
12.3.2. Predefined calculation types 932
12.4. Calculation Registers 935
12.4.1. General information on calculation registers 935
12.4.2. Editing Calculation Register 936

Chapter 13. Business processes and tasks 939

13.1. Basic concepts 939
13.2. General part 939
13.3. Routing 940
13.4. Addressing System 941
13.5. Business process start 944
13.6. Completing tasks 945
13.7. Separation and merger 946
13.8. Manual control 946
13.8.1. Business process completion flag 946
13.8.2. Task execution flag 947
13.8.3. Removing tasks 947
13.8.4. Adding tasks 948
13.9. Condition point 948
13.10. Switch 949
13.11. Generating tasks 949
13.12. Checking performance 951
13.13. Completing nested processes 952
13.14. Business process completion 954
13.15. Standard attributes of business processes and tasks 954
13.16. Business processes with multiple starting points 955
13.17. Feedback 956
13.18. Flowchart 959
13.19. Editing business process 959
13.20. Editing tasks 960

Chapter 14. Data analysis and forecasting 963

14.1. General information 963
14.2. Functionality layout 964
14.2.1. Feature main objects 965
14.2.2. Data analysis types 966
14.2.3. Forecast models 968
14.3. "General statistics" analysis type 968
14.4. "Search for associations" analysis type 973
14.4.1. Rule prune types 978
14.5. "Search for sequences" analysis type 979
14.6. "Decision tree" analysis type 984
14.7. "Clusterization" analysis type 992
14.7.1. Used metrics 998
14.7.2. Clusterization methods 1000

Chapter 15. Data exchange features 1003

15.1. Tasks and goals 1003
15.1.1. General information 1003
15.1.2. Universal data exchange features 1003
15.1.3. Distributed infobases 1003
15.2. Universal data exchange features 1004
15.2.1. General information 1004
15.2.2. XML documents reading and writing tools 1004
15.2.3. XML serialization 1004
15.2.4. Exchange plans 1020
15.3. Distributed infobases 1036
15.3.1. General principles 1036
15.3.2. Exchange plans 1038
15.3.3. Operating distributed infobase 1044
15.3.4. Scenarios of data exchange in the distributed infobase 1051

Chapter 16. Operation various data formats 1057

16.1. XDTO mechanism 1057
16.1.1. General information 1057
16.1.2. XDTO factory 1058
16.1.3. XDTO data types 1061
16.1.4. XDTO data instances 1067
16.1.5. XDTO-based XML serializer 1076
16.1.6. Recommendation on XML schema appearance 1077
16.1.7. Rules for XDTO factory verification 1079
16.2. JSON operations 1091
16.2.1. General information 1091
16.2.2. Object method 1093
16.2.3. Operating XDTO objects 1103
16.2.4. Srteams operation method 1109
16.2.5. Combining methods 1112

16.3. Binary data operations ... 1115
16.3.1. General information ... 1115
16.3.2. Reading binary data ... 1116
16.3.3. Writing binary data ... 1117
16.3.4. Operating binary data buffer ... 1117
16.3.5. Values in binary and hexadecimal notation and logical operations ... 1119
16.3.6. Specifications ... 1120
16.4. PDF files ... 1126
16.4.1. General information ... 1126
16.4.2. Attachments ... 1127
16.4.3. Digital signature ... 1128

Chapter 17. Online services mechanisms ... 1133

17.1. Web services ... 1133
17.1.1. General information ... 1133
17.1.2. Providing functions via web services ... 1135
17.1.3. Example of web service implementation ... 1136
17.1.4. Operation of other providers' web services ... 1139
17.1.5. Editing properties of the web service ... 1145
17.2. HTTP services ... 1147
17.2.1. Standard OData interface ... 1147
17.2.2. HTTP services ... 1183
17.3. Repeated usage of internet services sessions ... 1192
17.3.1. General information ... 1192
17.3.2. Types of repeated usage of sessions ... 1193
17.3.3. Settings of repeated usage of sessions ... 1194

Chapter 18. Web client and web site integration ... 1197

18.1. General information ... 1197
18.2. Application Interface ... 1197
18.3. Web Site Interface ... 1198
18.4. Implementation example ... 1200

Chapter 19. The functionality of jobs ... 1205

19.1. General information ... 1205
19.2. Background jobs ... 1206
19.2.1. General information ... 1206
19.2.2. Handling background jobs from the 1C:Enterprise language ... 1207
19.2.3. The background job and extensions ... 1210
19.3. Scheduled jobs ... 1210
19.4. Features of execution of background jobs in the file and client/server modes ... 1213
19.5. Creation of scheduled job meta data ... 1215

Chapter 20. Functionality of full-text search in the data 1219

20.1. Overall information 1219
20.2. Mechanism usage 1221
20.3. Using additional dictionaries 1224
20.4. Mobile platform 1226

Chapter 21. The temporary storage functionality, handling files and pictures 1227

21.1. Temporary storage 1227
21.2. Features of handling files in the asynchronous mode and in the web client 1229
21.3. File and temporary storage operation methods 1231

21.3.1. General information 1231
21.3.2. Migration of the file data to the temporary storage 1232
21.3.3. Data retrieval from the infobase to the file 1237
21.3.4. Data retrieval from the temporary storage 1239
21.3.5. Data removal from the temporary storage 1239
21.3.6. Check whether the address is attributed to the temporary storage 1240
21.3.7. Retrieve the address of the attribute 1240
21.3.8. Authorization for the group of procedures with files 1240
21.3.9. Operating temporary storage in the background job 1242
21.3.10. Access to standard directories 1242

21.4. Work with pictures 1243

21.4.1. Parameters of figure and operation of conversion 1243
21.4.2. Support of addresses in the image field 1245

Chapter 22. Event log 1249

22.1. General information 1249
22.2. Specification management 1249
22.3. Recording of events 1249
22.4. Event log management 1250

22.4.1. General information 1250
22.4.2. Setting of the Access event parameters 1250
22.4.3. Setting of parameters for the event "Access Denied" 1253

22.5. Getting event log records 1255
22.6. Additional methods 1260

Chapter 23. Cryptographic feature 1261

23.1. General information 1261
23.2. Basic concepts 1262
23.3. The general design to use cryptographic features 1262
23.4. Operation of cryptography modules 1264
23.5. Examples of usage 1265

23.6. The asynchronous operation of the cryptographic feature..... 1271

Chapter 24. External data sources 1275

24.1. General information..... 1275
24.2. Operation of relational external data sources 1275
24.2.1. General information..... 1275
24.2.2. The general operation design 1277
24.2.3. Editing the structure of the external data source..... 1278
24.2.4. Sample for generating the external data source..... 1287
24.2.5. ODBC connection string 1290
24.3. Operation of OLAP external data source..... 1292
24.3.1. General information..... 1292
24.3.2. The general operation design 1294
24.3.3. Editing the structure of the external data source..... 1295
24.3.4. Limitations of the query language when the analytical external data source is used 1299
24.3.5. String for connection with the OLAP server 1300
24.4. Operation of procedures and functions in external data sources..... 1301
24.4.1. General information..... 1301
24.4.2. The general operation design 1302
24.4.3. Editing functions 1302
24.5. Management of external data sources 1306
24.6. Connection to the external data source in the 1C:Enterprise 8 mode..... 1308
24.7. Features of external data sources..... 1309
24.7.1. Place for completion of queries to external data sources 1309
24.7.2. Usage of external data sources 1309
24.7.3. Rules of value transformation..... 1312
24.7.4. The external data source that is the component of the separator..... 1313

Chapter 25. Data separation feature..... 1315

25.1. General information..... 1315
25.2. Properties of the general attribute..... 1318
25.2.1. Separated data use..... 1318
25.2.2. The value of the separator and the flag of use..... 1319
25.2.3. Separation of users..... 1320
25.2.4. Separation of authentication and 1320
25.2.5. Separation of configuration extensions 1321
25.2.6. Conditional separation..... 1322
25.3. The definition of the ability to perform the action or set of data 1331
25.4. Setting the separator values at startup..... 1333
25.4.1. General information..... 1333
25.4.2. Client application command line..... 1334
25.4.3. Web client or thin client connected through a web server ... 1336
25.4.4. External connection and connection string 1336

25.5. Safe mode of data separation ... 1337
25.6. Deleting a data area ... 1338
25.7. Features of system objects behavior ... 1340

Chapter 26. Data history ... 1341

26.1. General information ... 1341
26.2. Mechanism structure ... 1342
26.3. Mechanism usage ... 1344
26.3.1. General information ... 1344
26.3.2. Managing the use of data history ... 1344
26.3.3. Write a version ... 1346
26.3.4. Processing after history writing ... 1348
26.3.5. Getting a version list ... 1349
26.3.6. Getting version data ... 1353
26.3.7. Version comments ... 1354
26.3.8. Version deletion ... 1355
26.3.9. Identifying differences between versions ... 1355
26.3.10. Data history and extension ... 1357

Chapter 27. Database copies mechanism ... 1359

27.1. General information ... 1359
27.2. Mechanism structure ... 1360
27.3. Data accelerator ... 1363
27.4. Specifics and limitations ... 1364
27.5. Copy content management ... 1365
27.6. Copy management ... 1368

Chapter 28. Global Search ... 1369

28.1. General information ... 1369
28.2. Global search plan ... 1370
28.3. Search execution and search result formation ... 1374
28.4. Search string processing and new type of search ... 1375
28.5. Search result processing ... 1377
28.6. Other specifics ... 1378

Chapter 29. Development for mobile devices ... 1381

29.1. General information ... 1381
29.2. Application interface ... 1383
29.2.1. General information ... 1383
29.2.2. Global command interface ... 1385
29.2.3. Home page ... 1385
29.2.4. User message ... 1387
29.3. Mobile version "1C:Enterprise" ... 1387
29.3.1. General information ... 1387
29.3.2. File operations ... 1387
29.3.3. Work with pictures ... 1389

29.3.4. Forms ... 1390
29.3.5. Special features of the mobile device ... 1406
29.3.6. Add-ins ... 1464

29.4. Application on the mobile platform ... 1466

29.4.1. General information ... 1466
29.4.2. Types of applications on the mobile platform ... 1467
29.4.3. Specifications ... 1469

29.5. "1C:Enterprise" Mobile client ... 1471

29.5.1. General information ... 1471
29.5.2. Preparing the configuration for operation in the mobile client ... 1472
29.5.3. Digital signature and configuration digest ... 1472
29.5.4. Operation with data associated with the current row of tables ... 1474

29.6. Offline mobile client ... 1475

29.6.1. General information ... 1475
29.6.2. Standalone configuration content ... 1477
29.6.3. Defining a server during a call ... 1480
29.6.4. Offline mobile application form ... 1481
29.6.5. Data exchange in operation ... 1483
29.6.6. Standalone/main server switchover ... 1485
29.6.7. Using an offline mobile application ... 1486

29.7. Mobile device interaction during development ... 1489

29.7.1. General information ... 1489
29.7.2. Installation of developer tools ... 1489
29.7.3. Publish a mobile application on a web-server ... 1495
29.7.4. Start and restart the application on the mobile platform ... 1498
29.7.5. Debugging an Application ... 1500
29.7.6. Mobile application management for developer ... 1500
29.7.7. Build a mobile application before publishing to the application store ... 1502

29.8. Mobile Application Builder ... 1503

29.8.1. General information ... 1503
29.8.2. Setting the builder ... 1503
29.8.3. Download the required data ... 1510
29.8.4. Setting of the application to be built ... 1516
29.8.5. Build the application ... 1529
29.8.6. Application publishing ... 1534
29.8.7. Additional features of the mobile application builder ... 1535

29.9. Application store publishing ... 1536

29.9.1. General information ... 1536
29.9.2. For OS iOS ... 1536
29.9.3. For OS Android ... 1538
29.9.4. On Windows ... 1539

Chapter 30. Collaboration system ... 1541

30.1. General information ... 1541
30.2. Collaboration system arrangement ... 1544
30.3. Interactive collaboration ... 1545

30.3.1. Users parameters ... 1545
30.3.2. Kind of conversations ... 1545
30.3.3. Kind of messages ... 1547
30.3.4. Notifications ... 1549
30.4. Non-interactive collaboration ... 1550
30.4.1. General information ... 1550
30.4.2. Register/register application cancel ... 1551
30.4.3. User management ... 1553
30.4.4. Using conversations ... 1559
30.4.5. Shared use ... 1570
30.5. Administration ... 1572

Chapter 31. Development tools ... 1573

31.1. Form editor ... 1573
31.1.1. The editor decription ... 1573
31.1.2. Role-based form customization ... 1580
31.2. Text editor ... 1581
31.2.1. Module edit ... 1582
31.2.2. Editing text templates ... 1606
31.2.3. Editing template texts ... 1615
31.2.4. Query text editor ... 1616
31.3. Configuration command interface editor ... 1616
31.3.1. General rules for setting visibility ... 1617
31.3.2. Filter by roles ... 1617
31.4. Start pagework area editor ... 1618
31.4.1. Using the start page work area editor ... 1618
31.4.2. Software ... 1618
31.5. Main Section Command Interface Editor ... 1621
31.6. Command interface editor ... 1622
31.7. All Subsystems Editor ... 1623
31.7.1. Subsystem Contents Setup ... 1625
31.7.2. Subsystem Command Interface Setting ... 1625
31.8. Query Wizard ... 1625
31.9. Query Wizard with result processing ... 1634
31.10. Register Records Wizard ... 1635
31.11. Print Wizard ... 1636
31.12. Create based on Wizard ... 1640
31.13. Configuration Object Form Wizard ... 1642
31.13.1. Constant Form Wizard Features ... 1644
31.14. Template Wizard ... 1646
31.15. Format String Wizard ... 1648
31.16. Wizard for Strings in Different Languages ... 1652
31.17. Spreadsheet document editor ... 1652
31.17.1. What is a spreadsheet document in 1C:Enterprise ... 1653
31.17.2. Templates ... 1653
31.17.3. Spreadsheet Document in Data Input Mode ... 1660
31.18. Flowchart editor ... 1665

31.18.1. Flowchart editing 1666
31.18.2. Flowchart items 1666
31.18.3. Flowchart items 1668
31.18.4. Modules 1671

31.19. Image Editor 1672

31.19.1. General information 1672
31.19.2. Editing images 1673
31.19.3. Image Collections 1676

31.20. HTML Document Editor 1677

31.20.1. General information 1677
31.20.2. Visual Editing 1678
31.20.3. Editing HTML document source code 1681
31.20.4. Previewing HTML documents 1681

31.21. Localization of Configurations 1682
31.22. Centralized Configuration Verification 1687
31.23. Usability analysis for solutions based on 1C:Enterprise platform 1694

Chapter 32. Debugging and testing of application solutions 1695

32.1. General information 1695
32.2. Debugger 1695

32.2.1. General information 1695
32.2.2. Debugger Use 1698
32.2.3. Breakpoint 1719
32.2.4. Step-by-Step Execution 1721
32.2.5. Debugging external data processors (reports) 1724
32.2.6. Debug management 1724
32.2.7. "Expression" window 1725
32.2.8. Immediate window 1727
32.2.9. "Local Variables" window 1728
32.2.10. Variable value change 1729
32.2.11. Call stack 1730
32.2.12. Debugging Protected Modules 1731

32.3. Performance Snapshot 1732

32.3.1. Available Action Options 1732
32.3.2. Measurement Results 1733
32.3.3. Sorting Measurement Results 1736
32.3.4. Selective Summarization of Measurement Results 1736
32.3.5. Performance Snapshot Features during Client/Server Infobase Operation 1737
32.3.6. Saving Results 1737

32.4. Server call delay imitation 1737
32.5. Server call history 1738
32.6. Automated testing of applied solutions 1741

32.6.1. General information 1741
32.6.2. Testing system start-up 1743
32.6.3. Description of features 1746
32.6.4. Sample Testing Scenario 1751

32.6.5.Service features..... 1751

Chapter 33. Compare and merge configurations..... 1753

33.1. General information..... 1753
33.2. Configuration comparison conditions..... 1753
33.3. Launch of the configuration compare and merge mode..... 1754
33.4. Comparing configurations..... 1755
33.5. "Compare and Merge Configurations" window..... 1756
33.6. Configuration Comparison Setup..... 1759
33.6.1.General information..... 1759
33.6.2.Master Configuration Objects Group..... 1760
33.6.3.Loaded Configuration Objects Group..... 1760
33.7. Object Editing..... 1760
33.8. Multiply development and support - editing features..... 1761
33.9. Object Mapping..... 1761
33.10.Comparison and merging features..... 1762
33.10.1. General information..... 1762
33.10.2. Text documents (including source code)..... 1763
33.10.3. Managed forms..... 1764
33.10.4. Comparing predefined data..... 1766
33.10.5. Comparing configuration extensions..... 1766
33.10.6. Data Types..... 1767
33.10.7. Exchange plans, subsystems and functional options contents..... 1768
33.10.8. Other objects..... 1769
33.11.Subordinate objects order setting..... 1770
33.12.Viewing differences..... 1770
33.13.Comparing and merging modules..... 1771
33.14.Selecting objects and starting the merge process..... 1774
33.15.Loading configurations from file..... 1776

Chapter 34. Distributed configuration development . 1777

34.1. General information..... 1777
34.2. Repository administration..... 1778
34.2.1.Creating repositories..... 1778
34.2.2.Connecting to Repository..... 1780
34.2.3.Connecting to repository..... 1781
34.2.4.Configuration window..... 1783
34.2.5.Maintain a list of repository users..... 1784
34.2.6.Cache versions of the repository..... 1785
34.2.7.Configuration repository administration..... 1786
34.2.8.Repository backup..... 1791
34.3. Configuration repository operations..... 1791
34.3.1.Repository filter..... 1792
34.3.2.Locking repository objects..... 1793
34.3.3.Storing objects in repository..... 1794
34.3.4.Retrieving objects from repository..... 1795
34.3.5.Unlocking objects..... 1796
34.3.6.Updating status of repository objects..... 1796

34.3.7. Comparing repository with current configuration 1797
34.3.8. Comparing a repository object with a current configuration object 1797
34.3.9. Saving repository as configuration 1798
34.3.10. Disconnecting from repository 1798
34.3.11. Repository history 1798
34.3.12. Object history 1803
34.3.13. Actions with Configuration Repository without Connection 1804

34.4. Working remotely with configuration repository 1805

34.4.1. General information 1805
34.4.2. Architecture Overview 1806
34.4.3. Setting up the repository server 1809
34.4.4. Configuring a web server to work with the repository 1809
34.4.5. Start the repository server 1814
34.4.6. Connecting to the Configuration Repository Server 1816
34.4.7. Creating a configuration repository 1817
34.4.8. Features of remote work with the configuration repository 1818

34.5. Recommendations for the use of repository 1818

Chapter 35. Configuration distribution and support .. 1819

35.1. General information 1819
35.2. Configuration distribution 1820

35.2.1. Distribution settings 1820
35.2.2. Creating distribution files 1822
35.2.3. Preparing the distribution package of replicable configurations 1823

35.3. Configuration support 1830

35.3.1. General information 1830
35.3.2. Setting Up Configuration Support 1831
35.3.3. Support Setting 1832
35.3.4. Updating configuration 1836
35.3.5. Template Directory Description File 1844

Chapter 36. Configuration extension 1847

36.1. General information 1847

36.1.1. Terms and definitions 1847
36.1.2. Brief description of the technique 1848
36.1.3. Specifics and limitations 1850

36.2. Managing extentions 1851

36.2.1. In Designer mode 1851
36.2.2. In 1C:Enterprise mode 1852
36.2.3. Deleting extensions 1852

36.3. Operations with and extension 1853

36.3.1. Creating an adopted object 1853
36.3.2. Operations with adopted object 1854
36.3.3. Creating native object 1856
36.3.4. Blocking extensions 1856

36.3.5. Shared extension development ... 1856
36.4. Extension objects ... 1857
36.4.1. Extension properties ... 1857
36.4.2. Modules ... 1858
36.4.3. Forms ... 1867
36.4.4. Form command interface ... 1872
36.4.5. Object forms ... 1873
36.4.6. Subsystems ... 1873
36.4.7. Access rights ... 1873
36.4.8. Reports and data processors ... 1879
36.4.9. XDTO packages ... 1880
36.4.10. Web services ... 1880
36.4.11. HTTP service ... 1880
36.4.12. The predefined elements ... 1881
36.4.13. Spreadsheet documents ... 1881
36.4.14. Exchange plans ... 1883
36.4.15. Catalogs ... 1883
36.4.16. Documents ... 1884
36.4.17. Enumerations ... 1884
36.4.18. Session parameters ... 1885
36.4.19. Functional options and functional option parameters 1885
36.4.20. Filter criteria ... 1885
36.5. Extension and distributed infobase ... 1886
36.6. Security profiles ... 1887
36.7. Attachment of extensions and time of their operation ... 1888
36.7.1. General information ... 1888
36.7.2. Operation at discrepancy of data structures in a configuration and infobase ... 1889
36.8. Operation with extensions from the 1C:Enterprise language ... 1890

Chapter 37. Service features ... 1891

37.1. Switching between application windows ... 1891
37.1.1. Window states (window display modes) ... 1891
37.1.2. Managing windows ... 1894
37.1.3. Use of New Window mode ... 1895
37.1.4. Service windows ... 1895
37.1.5. Closing windows ... 1896
37.1.6. Windows navigation history ... 1896
37.2. Setting Designer parameters ... 1896
37.2.1. Overall ... 1896
37.2.2. Texts ... 1898
37.2.3. Modules ... 1900
37.2.4. Debug ... 1907
37.2.5. 1C:Enterprise startup ... 1909
37.2.6. Comparison/merging ... 1917
37.2.7. Forms ... 1920
37.2.8. Help ... 1922
37.3. Using calculator ... 1923
37.4. Calendar ... 1923
37.5. Text templates ... 1923

37.5.1. Template list maintenance ... 1924
37.5.2. Editing a template ... 1926
37.6. Syntax Assistant ... 1937
37.6.1. General information ... 1937
37.6.2. Syntax Assistant settings ... 1939
37.6.3. "Content" Tab of Syntax Assistant ... 1940
37.6.4. Copying 1C:Enterprise language Items ... 1941
37.6.5. Searching in Syntax Assistant ... 1941
37.6.6. Using methodological materials ... 1945
37.6.7. Versions details ... 1945
37.7. Comparing files ... 1946
37.7.1. Comparison of external data processors ... 1947
37.8. Built-in help application ... 1947

Chapter 38. Add-ins ... 1949

38.1. General information ... 1949
38.2. Features of operation in thin and thick clients ... 1950
38.3. Operation features in web client ... 1951
38.4. Features of operations on server ... 1952
38.5. Features of operations in mobile application ... 1953
38.6. Asynchronous operation with an add-in ... 1953
38.7. Examples of add-in use ... 1954
38.7.1. Native API technology ... 1954
38.7.2. COM technology ... 1956

Chapter 39. Specifics of cross-platform application development ... 1957

39.1. General information ... 1957
39.2. File mode features ... 1957
39.3. Operating with external devices ... 1959
39.4. Rights limitations ... 1959
39.5. Fonts ... 1959
39.6. HTMLDocumentField ... 1960
39.7. COM, OLE and ActiveDocument mechanisms ... 1960

Chapter 40. Other mechanisms ... 1961

40.1. Solving systems of linear equations ... 1961
40.1.1. General information ... 1961
40.1.2. Graph cohesion components ... 1968
40.1.3. Cohesion components search mechanism use recommendations ... 1974

Appendix 1. Conversion of "1C:Enterprise 7.7 " infobases 1978
Appendix 2. URL formats 1983
2.1. General description of the link format 1983
2.2. Host address format 1984
2.3. Internal links 1985
2.3.1. General information 1985
2.3.2. Infobase objects links 1985
2.3.3. Collaboration system objects links 1990
Appendix 3. Rules of generating standard command texts and automatic form headers 1991
Appendix 4. List of automatically saved settings 1995
4.1. Storage: report options 1995
4.2. Storage: report settings 1995
4.3. Storage: form data 1996
4.4. Storage: dynamic list settings 1996
4.5. 1C:Enterprise storage 1996
Appendix 5. Full-text search expressions 2005
Appendix 6. Access rights description 2009
Appendix 7. System behavior features in different modes 2015
7.1. Features of slow connection speed mode 2015
7.2. Compatibility mode features 2016
7.2.1. Previous versions 2016
7.2.2. With version 8.3.5 2025
7.2.3. With 8.3.6 version 2028
7.2.4. In version 8.3.7 2031
7.2.5. With version 8.3.8 2034
7.2.6. With version 8.3.9 2037
7.2.7. With version 8.3.10 2044
7.2.8. With version 8.3.11 2052
7.2.9. With version 8.3.12 2057
7.2.10. With version 8.3.13 2062
7.2.11. With version 8.3.14 2066
7.2.12. With version 8.3.15 2069
7.3. Specifics of the extension compatibility mode 2070
7.4. Specifics of web client operations 2071
7.5. Features of web client operations on iPad 2079
7.6. Features of system object behavior with the enabled data separation mode 2081
7.6.1. Changing separator values 2081
7.6.2. Objects numeration 2082
7.6.3. Real-time timestamp 2082

7.6.4. Predefined data ... 2082
7.6.5. Access rights ... 2083
7.6.6. Users ... 2084
7.6.7. Exchange plans ... 2084
7.6.8. Functional options ... 2086
7.6.9. Scheduled jobs ... 2086
7.6.10. Background jobs ... 2087
7.6.11. Web services ... 2088
7.6.12. Constants ... 2088
7.6.13. Registers ... 2088
7.6.14. Document sequence ... 2092
7.6.15. Infobase parameters ... 2092
7.6.16. Session list ... 2092
7.6.17. Connection list ... 2092
7.6.18. Event log ... 2093
7.6.19. Settings and favorites ... 2096
7.6.20. History ... 2097
7.6.21. Settings OData standard interface ... 2097
7.6.22. Queries ... 2097
7.6.23. Managed transaction locks ... 2097
7.6.24. Exclusive mode ... 2098
7.6.25. Deleting marked objects ... 2098
7.6.26. Using full-text search ... 2099
7.6.27. Configuration extension ... 2100
7.6.28. Data history ... 2100
7.6.29. Collaboration system ... 2101
7.7. Specifics of client application operating on Linux OS ... 2101
7.8. Features of the client application on macOS ... 2102
7.9. Specifics mobile version of 1C:Enterprise ... 2103
7.10. Protection mechanism against dangerous actions ... 2105
7.10.1. General information ... 2105
7.10.2. Disabling the mechanism of protection against dangerous actions ... 2107

Appendix 8. Features of operating with different DBMS 2109

8.1. General features ... 2109
8.2. File database ... 2109
8.3. IBM DB2 ... 2109
8.4. Microsoft SQL Server ... 2111
8.5. Oracle Database ... 2112
8.6. PostgreSQL ... 2114

Appendix 9. XBase operations ... 2117

9.1. Fields and records ... 2117
9.2. A table, a table structure, a database file ... 2117
9.3. Indexes, expressions of an index and a filter, an index file ... 2118
9.4. Recording of changes to a database ... 2118
9.5. Operating with index files ... 2118
9.6. Record removing ... 2119

9.7. Creating a database, index, an index file 2119
9.8. Restrictions 2119

Appendix 10. Specifics of using some features 2121
Appendix 11. The rules of automatic form items names generation 2127
Appendix 12. A description of entities that are provided via the standard OData interface..... 2131

12.1. Constants 2131
12.2. Exchange plans 2131
12.2.1. Tabular section of exchange plan 2132
12.2.2. Tabular section row 2133
12.3. Catalogs 2133
12.3.1. Directory tabular section 2134
12.3.2. Tabular section row 2134
12.4. Documents 2134
12.4.1. Methods 2135
12.4.2. Directory tabular section 2136
12.4.3. Tabular section row 2136
12.5. Documents journals 2136
12.6. Charts of characteristic types 2137
12.6.1. Tabular section of the chart of characteristic types 2139
12.6.2. Tabular section row 2139
12.7. Charts of accounts 2139
12.7.1. Tabular section of chart of accounts 2140
12.7.2. Tabular section row 2141
12.7.3. Standard tabular section of extra dimension type string..... 2141
12.8. Charts of calculation types 2142
12.8.1. Tabular section of chart of calculations 2143
12.8.2. Tabular section row 2143
12.8.3. String of the standard tabular section “Leading calculation types” 2144
12.8.4. String of the standard tabular section “Displacing calculation types” 2144
12.8.5. String of the standard tabular section “Basic calculation types” 2144
12.9. Information registers 2145
12.9.1. Independent 2145
12.9.2. Subordinate to the recorder 2146
12.10. Accumulation register 2148
12.10.1. Functions 2149
12.10.2. Register record set 2150
12.10.3. String of accumulation register record set 2151
12.11. Accounting registers 2151
12.11.1. Functions 2152

12.11.2. Register record set ... 2155
12.11.3. String of the information register record set ... 2155
12.12. Calculation Registers ... 2156
12.12.1. Functions ... 2157
12.12.2. Register record set ... 2158
12.12.3. String of the information register record set ... 2159
12.13. Business processes ... 2160
12.13.1. Functions ... 2161
12.13.2. Tabular section of business process ... 2161
12.13.3. Tabular section row ... 2162
12.14. Tasks ... 2162
12.14.1. Functions ... 2163
12.14.2. Tabular section of business process ... 2163
12.14.3. Tabular section row ... 2164

Appendix 13. Namespace prefixes in JSON serialization ... 2165
Appendix 14. New and modified sections ... 2167

Introduction

This book is a guide for the 1C:Enterprise application developers.
This document is intended for professionals involved in the development, modification, and implementation of applications on the 1C:Enterprise platform.

Overview

The Guide describes the general concept of 1C:Enterprise, as well as operations with application objects, form creation, command interface, and the use of 1C:Enterprise features.
Chapter 1 describes the concept of 1C:Enterprise: general principles, general logic of operations, and more.
Chapter 2 describes operations with the configuration in general: editing configuration properties, copying configuration objects, saving configurations, merging configurations, using auxiliary modes of operation.
Chapter 3 describes the application interface and structure.
Chapter 4 contains general information about the 1C:Enterprise language, a description of the data types used, operators and syntactic structures, as well as the basic techniques for using the 1C:Enterprise language.
Chapter 5 describes the main configuration objects and their features.
Chapter 6 describes the command interface and its development logic.
Chapter 7 describes forms and form components, as well as form operations using 1C:Enterprise language.
Chapter 8 describes the query language and basic techniques for using queries in the 1C:Enterprise language.
Chapter 9 describes the general principles and features of data processing in 1C:Enterprise.
Chapter 10 describes data composition system.
Chapter 11 describes the organization of accounting processes in 1C:Enterprise.
Chapter 12 describes operations with periodic calculations used in 1C:Enterprise.
Chapter 13 describes business processes and tasks.
Chapter 14 describes the data analysis and forecasting functionality.
Chapter 15 describes the data exchange functionality.
Chapter 16 describes operations with data formats such as: XML (XDTO), JSON, binary data.
Chapter 17 describes the functionality of web services, HTTP services, and OData standard interface.
Chapter 19 describes the job functionality.
Chapter 20 describes the full-text database search functionality.

Chapter 21 describes operations with temporary data storage and how to use the temporary storage for file operations.
Chapter 22 describes operations with the event log.
Chapter 23 describes the cryptography functionality.
Chapter 24 describes operations with external data sources.
Chapter 25 describes the data separation functionality.
Chapter 26 describes the data versioning functionality.
Chapter 27 describes the work with copies of the databases (including the description of Data accelerator mechanism).
Chapter 29 describes the procedure of mobile application development.
Chapter 30 describes the functionality of collaboration system.
Ошибка! Источник ссылки не найден. describes the use of form editor, text editor, spreadsheet document editor, flowchart editor, and image editor.
Chapter 32 describes configuration debugging and testing.
Chapter 33 describes configuration comparing and merging functionality.
Chapter 34 describes the organization and use of group development of configurations.
Chapter 35 describes the mechanisms for creating distribution files, update files, and distribution packages. This chapter will be of interest to developers of replicable configurations. This chapter also describes the use of distribution and update files for standard configurations that are supported by user configuration developers.
Chapter 36 describes configuration extensions - a mechanism designed to enhance an extended configuration without changing it.
Chapter 37 describes the service modes: configuring the Designer settings, using the syntax assistant, setting up templates, using the built-in calculator and calendar, comparing files, performing operations with windows, localizing 1C:Enterprise configurations.
Chapter 38 describes the add-in attachment functionality.
Chapter 39 contains recommendations for development of cross-platform applications.
Chapter 40 describes the features not included in the chapters above.

What you need to know

This guide assumes that you are familiar with the operating system of the computer where 1C:Enterprise is installed (Microsoft Windows 2000, Microsoft Windows XP, Microsoft Windows Vista, Microsoft Windows 7, hereinafter referred to as - Microsoft Windows), and have the basic skills required to work with this operating system.
You must be familiar with the following concepts:

- How to use Start menu to start applications
- How to work with windows

- How to use mouse
- How to work with text and spreadsheet documents - entering text, entering values into cells of a spreadsheet document, editing, formatting, printing, etc.
- How to use menus
- How to operate dialog box controls
- How to use standard dialog boxes
- How Microsoft Windows clipboard operates, and how to use it
- How to customize a Microsoft Windows operating system using the control panel

If you have not completely mastered these concepts, we recommend that you refer to the operating system documentation.

Books included in the documentation

The documentation package includes the following books:

- 1C:Enterprise 8.3. User Manual (https://1c-dn.com/library/user_manual/). Describes the basic concepts and features that are common for all 1C:Enterprise applications.
- 1C:Enterprise 8.3. Developer Guide (https://1c-dn.com/library/developer_guide/). Describes how to customize applications to automate accounting procedures in a specific company, as well as how to develop new applications.
- 1C:Enterprise 8.3. Administrator Guide (https://1c-dn.com/library/administrator_guide_full/). Describes 1C:Enterprise administration, including the deployment of client/server systems.
- 1C:Enterprise 8.3. Client/server mode Administrator Guide (https://1c-dn.com/library/administrator_guide/). Describes how to deploy and operate 1C:Enterprise 8 infobases in client/server mode.
- The object model is described in the online help (Designer help and Syntax Assistant).

IMPORTANT! Some software distribution packages might not include some of these books.

Update and migration notes delivered with 1C:Enterprise

The 1C:Enterprise installer copies several text files that list the changes implemented in the platform version and instructions for migration to this platform version to the hard disk.

These files are located in the 1C:Enterprise installation directory, in the \docs\en subdirectory. If you do not change the default installation path, they will be located in C:\Program Files\1cv8\VersionNumber\docs\en. For example, the path for version 8.3.2.100 is as follows: C:\Program Files\1cv82\8.3.2.100\docs\en.

- The file V8Update.htm - contains the list of changes and migration instructions.

http://www.1c.ru/rus/partners/service.jsp

Agreed notations

Keys. Keys Enter, Esc, Del and others are given without quotation marks.

The Arrow keys phrase is used to specify all arrow keys at once. They are referred individually as Up Arrow, Down Arrow, Right Arrow, and Left Arrow.

Keyboard shortcuts. When a command requires a keyboard shortcut it will be stated as Ctrl + F3.

Buttons. Form buttons are given without quotation marks, like OK, Cancel, Delete and others.

1C:Enterprise language keywords. 1C:Enterprise language keywords are highlighted by specific font and are given as in modules, for example: `WorkingDate`. This manual contains references to some parts of 1C:Enterprise language description (sections, methods, attributes and others). For these descriptions, see the application help (1C:Enterprise language section).

Menu actions. The menu actions are described as follows: Menu - Submenu - Submenu - ... - Menu item. Example: "Use Table - View - Scope to change the scope of the picture" means: "Use the Scope item of the View submenu of the Table menu of the application's main menu". If any menu, other than the main menu, is referred, it is specified explicitly.

1C:Enterprise modes. 1C:Enterprise can run in two modes: configuration setup and verification (hereinafter - Designer), and configuration execution (hereinafter - 1C:Enterprise mode).

In the context of this Guide, "user" means an expert participating in application development or maintenance.

The %APPDATA% expression means the Windows environment variable with the current directory containing application data. If the operating system

was installed using default settings and the user name is Ivanov, this expression equals to

```
C:\Documents and Settings\Ivanov\Application Data
```

For Windows Vista and higher:

```
C:\Users\Ivanov\AppData\Roaming
```

The **%LOCALAPPDATA% expression** means the Windows Vista environment variable with the current directory for user-specific application data. If the operating system was installed using default settings and the user name is Ivanov, this expression equals to

```
C:\Users\Ivanov\AppData\Local
```

Chapter 1. General concepts

1C:Enterprise is a universal enterprise automation platform. It is a development and runtime environment that includes a set of tools, mechanisms, and services that provide everything you need for application development, operation, administration, and maintenance. 1C:Enterprise consists of:

- The **platform**, serving as the application development and runtime environment
- Platform-based **applications** of different scale and intended for various tasks
- **Methodology** of application development
- **Information and technological support** for users and developers

Due to its universality, 1C:Enterprise can be used for automation of various economic activities of the enterprise: accounting of goods and material assets, settlements with counterparties, and much more.

1.1. Configurability

Configurability is a major feature of 1C:Enterprise. 1C:Enterprise is basically a set of mechanisms designed to manipulate different types of objects of the subject domain. Each 1C:Enterprise configuration determines a set of objects, structures of information arrays, and information processing algorithms that best correspond to the task. A configuration combined with 1C:Enterprise acts as a ready-to-use software product focused on certain types of enterprises and categories of tasks.

Configurations are created and maintained through the regular 1C:Enterprise functionality. A configuration is usually supplied in its standard version, but it can be modified, enhanced by users, or redesigned. 1C:Enterprise provides support for standard configurations.

1.2. Operations

1C:Enterprise operations are divided into two stages - development (description of the subject domain model by means of the application) and runtime (processing of subject domain data).

The development stage includes:

- **Creating structure of the** processed **information**
- **Creating forms** for entering initial data and for viewing data lists

- Organizing **storage** of the entered and the final **information**
- **Designing reports and** data processors
- Creating **command interfaces** for different groups of users
- Creating **the list of users**
- Granting **rights** to users

The result of the development is a software product (configuration) representing a subject domain model.
In Designer, you can create, edit, compare, and merge configurations.
The development stage deals with documents, document journals, catalogs, attributes, forms, registers, and other universal 1C:Enterprise concepts. The combination of these concepts defines the concept of the application.
The concepts of objects and standard processing operations are defined at the platform level. Configuration tools allow you to describe the structure of data contained in these objects, implement algorithms for data processing, and reflect accounting specifics if necessary.
The data structure is designed in terms of types of subject domain objects (constants, catalogs, documents, registers, enumerations, etc.).
During the runtime stage, 1C:Enterprise uses objects described during the configuration stage (catalogs of goods and companies, invoices, bills, and more).
When in 1C:Enterprise mode, data processing is performed both by regular system tools and by custom algorithms created during the configuration stage.

1.3. Basic concepts

This section describes the basic 1C:Enterprise concepts. This material will be useful for those who are not yet familiar with 1C:Enterprise.
Descriptions of 1C:Enterprise concepts will be accompanied by examples. If you encounter an unfamiliar concept or term, just continue reading: the meaning of the terms used will be explained as you go. Refer to the relevant chapters of this Guide for further details.

1.3.1. The concept of "configuration"

Configuration is the most basic underlying idea in 1C:Enterprise.
A configuration is a set of interrelated components:

- Subsystems
- Accounting data structure, complete with data input, selection, and print forms
- Functionality for accounting data recording and calculation of totals
- Reports and data processors
- Command interface

- Roles defining user access rights
- Common procedures and functions (application module, managed application module, external connection module, session module, and common modules), spreadsheet document templates, and more
- Auxiliary objects:
  - Functional options and their parameters
  - Storage settings
  - Web tools (Web services, WS references, HTTP services)
  - Auxiliary data (pictures, templates, styles, etc.)

Configuration structure is basically a subject domain model. You can create a configuration in Designer. 1C:Enterprise uses configurations to create software environments suitable for certain accounting tasks.

Roles define the privileges of 1C:Enterprise users to access data that is processed by the configuration. The set of privileges granted to a user is generally determined by his or her work duties.

Roles are assigned to users in order to:

- Restrict the list of users that can access confidential information (a major issue for any accounting system)
- Prohibit users from performing unsafe operations, such as data deletion, to prevent loss of information

All components of the configuration are closely interrelated, so that any changes (especially for user rights) need to be consistent.

For example, role assignment can only be performed for existing configuration objects (specific documents, logs, catalogs, reports). Whenever you add a new object to the configuration structure, this needs to be accompanied by appropriate changes to the role.

Access rights are considered when generating command interface. If, for example, a user is not allowed to view a catalog, the command to open the list form of this catalog will be removed from the command interface automatically. Access rights are also considered when displaying forms.

1.3.2. Configuration objects

1C:Enterprise configuration object is a formal description of a group of concepts with similar characteristics and the same purpose.

For example, -1C:Enterprise configuration object `Catalog` is intended for management of lists containing similar data items, such as catalogs, card files, rules and standards lists. Configuration objects of this type are used to maintain any catalogs necessary for automation of enterprise activities.

As a rule, configuration objects of `Catalog` type are digital versions of catalogs actually used in the enterprise, for example, the catalog of employees, or list of products; though they can also represent lists that have no physical versions.

Note that a configuration object only describes data structure and does not contain the data itself. For example, the catalog Individuals contains no information on any specific person, but instead describes a list of attributes, as well as forms for entering attribute values, forms for viewing data lists, and templates for printing the data. In other words, the configuration creates a description scheme that all similar objects of the subject domain must follow (in the above example, all individuals follow the same description scheme specified in the Individuals catalog).

1.3.2.1. Properties of configuration objects

Each configuration object has its own set of properties. This set is described at the system level and cannot be modified while setting up the configuration. The set of properties of the configuration object is determined mainly by its purpose in 1C:Enterprise.

-Name is the main property of any configuration object. -When you create a new configuration object, its name is generated automatically. consisting of the object type and a number (for example, when you create an attribute, it will be named `Attribute1`, a document will be named `Document1`, and so on). You can change this name while editing the properties of the configuration object. The name uniqueness control is automatic. Configuration object name cannot be empty or longer than 80 characters.

If some properties of a configuration object are editable, they can be modified during configuration stage. The nature of allowed changes in monitored at the system level. A specialist can achieve the desired behavior of the configuration object by purposefully changing its properties. However, such changes do not affect the basic nature of the object and do not enable it to perform actions that are extrinsic to objects of this type.

For example,

the `Constant` configuration object is designed to store information that does not change over time or changes rarely. Previous values of the constant are irrelevant. For a simple example of a constant, let us review the name of an enterprise: it usually does not change during the enterprise lifetime (if you need to access the values of any accounting data that change in time, use an information register without dimensions instead of a constant).

Constants have a large set of editable properties, including:

- Constant name
- Synonym
- Comment
- Data type
- Data lock control mode
- Link that opens the constant manager module

In the general case, the constant value (such as the name of the enterprise) is entered only once.
-

1.3.2.2. Basic types of configuration objects

1C:Enterprise configuration objects are divided into several basic types. Each type of configuration object represents the building blocks used to create the configuration.
Formally, configuration objects are grouped into types in the configuration tree. When you open the Configuration window in Designer, the available types are listed at the first level of the configuration tree.

Fig. 1. Metadata tree

Despite the lack of formal definition, names of configuration object types are commonly used by 1C:Enterprise users.
For example, the goal of a specialist configuring 1C:Enterprise is developing a set of catalogs, documents, reports, and logs, required to implement an accounting system. The end users of 1C:Enterprise - managers, accountants, storekeepers - use catalogs, documents, reports, and so on, to perform their daily tasks. These two user categories communicate in terms of types of configuration objects.
Each document, report, log, constant, and so on, is a data object. As a rule, each data object is used to process specific type of information of subject domain.
Below is a brief description of the main types of configuration objects in 1C:Enterprise. Detailed information about the configuration objects represented by each of these types will be provided below.

1.3.2.2.1. Constants

Objects of `Constant` type are used in 1C:Enterprise for operations with constant and conditionally constant information. Information stored in constants is rarely changed, but often read. For example, a constant can store the name of the company, its TIN, the names of the director and the chief accountant, or other similar information.
An unlimited number of constants can be defined.

1.3.2.2.2. Catalogs

Objects of `Catalog` type are used in 1C:Enterprise for operations with constant and conditionally constant information that has a defined set of values.
Generally, catalogs are lists of materials, goods, organizations, currencies, employees, and more.
The properties of each catalog are described during the configuration stage. The customizable properties include, among others, length and type of ID codes, the number of hierarchy levels, code uniqueness flag, and a set of catalog attributes. In addition to the code and name, you can create a set of catalog attributes for storing any additional information about a catalog item (for example, for a product, you may want to specify purchasing and selling prices, manufacturer; for employee - position, education, address of residence, and so on), as well as tabular sections. A tabular section stores multiple data items of a specific kind, such as descriptions of products components, names of the employee's family members, phone numbers, and more.
Multiple types of forms can be specified for each catalog: item form, group form, list form, selection form, group selection form. You can create any number of forms for each form type.
You can use the subordinate catalogs to describe subordinate entities. In this scenario, each item of a subordinate catalog belongs to a certain item of the owner catalog.
In a configuration, you create as many catalogs as you need for storing data on objects used in automation of a given subject domain. For example, you might want to create catalogs for `Organizations`, `Goods`, `Employees`, and so on.

1.3.2.2.3. Enumerations

Objects of Enumeration type are used in 1C:Enterprise to describe constant sets of values that are not changed during the lifetime of a configuration.
During the configuration stage, you can define an almost unlimited number of enumeration types. Unlike the catalogs, the enumeration values are set during the configuration stage and cannot be changed at runtime.
Typical examples of enumerations include payment types (cash, cashless, barter), customer statuses (regular, one-time), and so on.

A major difference between enumerations and catalogs is that a set of enumeration values never changes while the end user is logged on to the application. -Let us review an example where the configuration supports two customer statuses: regular and one-time. In this scenario, the customer status is specified by selecting one of the two enumeration values. The users cannot create new statuses.
Unlike enumerations, users can add new values to catalogs, such as: names of products, names of counterparties, etc.

1.3.2.2.4. Documents

Objects of Document type are used in 1C:Enterprise to record business events of the enterprise that are related to the automated subject domain. For example, a configuration that was developed to perform accounting of trading operations may contain the following documents: invoices, receipts, delivery notes, and more. You can also use documents to record payments made from the current account, operations on cash accounts, warehouse transfers, or any other events of similar kind.
During the configuration stage, you can define an unlimited number of document types. Typical examples of document types are Payment order, Invoice, Receipt note, Delivery note, Internal transfer note, Cash receipt, and others. Each document type is intended to record a specific event type. This purpose defines its structure and properties in the configuration.
Each type of document can have an unlimited number of attributes and tabular sections. Multiple tabular sections are used when you need to register different but related events in a single document, such as: arrival of goods at the warehouse and the incurred additional expenses (transportation costs, handling costs, etc).-
An input form is a representation of a document on the computer display.- To allow the users to include document data into other forms, selection forms are created. To view a list of documents of the same type, list forms are created. The number of forms you can create is unlimited. Each document can also have an unlimited number of printed forms.
Each document has a number, date, and time. When you set up a document, you also specify the length of the document number, the number uniqueness check conditions, and more.
Documents play a central role in core functionality of 1C:Enterprise. All documents form a single chronological sequence. It reflects the actual sequence of events. The sequence of documents with the same date is determined by their respective time stamps. The document time data is used to clearly organize documents with the same date, rather than to record the actual registration time for each document. The data entered into a document (into the attributes and tabular sections of the document) usually contains information about an event that has occurred. For example, data entered into an invoice might describe which goods in what amount were shipped from which warehouse and what additional costs were incurred when purchasing the goods.-

Posting is a crucial action for any document. If a document is not postable, it means that the event it reflects does not affect the accounting status in the application. When a document is posted, this changes the state of related data. Posting may reflect the document event in configuration registers.
For example, in a trading company, the billing of invoice to the customer does not change the state of the company's goods or funds, as the invoice is only an expression of the customer's intention to purchase the goods.- In this event, it is not necessary to reflect the Invoice document in the accounting registers of a trading accounting application.
However, if the invoice is accompanied by goods reservation for the customer, the Invoice document is reflected in the accounting registers because the invoice transaction also freezes a certain quantity of goods, making them temporarily unavailable. In this event, the accounting application needs to be able to track the reserved items.

1.3.2.2.5. Documents journals

Document journals are intended for viewing documents of different types. Each document type can be displayed in several journals. A document journal does not add any new data to the application; this is only a tool for displaying multiple views in a single document list.
For example, you can create a Warehouse documents journal that displays all receipt notes, delivery notes, and internal transfer notes.
Journal columns can be defined for displaying attributes of different document types assigned to this journal. For example, the Trade Documents journal might contain the Counterparty column that displays the Consignor attribute from the Consignment goods acceptance document, the Company attribute from the Goods receipt note document, and so on.
Each journal can have an unlimited number of visual and printed forms.

1.3.2.2.6. Reports and data processors

An unlimited number of reports and data processors can be created to describe reports and data processing procedures during the configuration phase. A report or a data processor can have multiple forms intended, for example, for entering report generation parameters or for entering data processing parameters. For example, a warehouse selection form might be used to issue a warehouse statement.-
The algorithm for receiving the report can be described using the 1C:Enterprise language or automatically generated by the application, if the data composition system is used. Both text format and a specialized tabular report format (template-based) can be used for reports.
1C:Enterprise also supports development of external data processors that can stored in separate files rather than in the application.

1.3.2.2.7. Charts of characteristic types

In 1C:Enterprise, the Chart of characteristic types objects are intended for describing sets of analytical accounting objects, each set storing objects of a single type.

1.3.2.2.8. Charts of calculation types

Objects of this type are intended for creating types of calculations used in periodic calculation mechanisms.

1.3.2.2.9. Charts of accounts

A chart of accounts is one of the basic concepts in bookkeeping. A chart of accounts is a set of synthetic accounts intended for grouping data related to the business activities of an enterprise. Data stored in such synthetic accounts provides the user with a complete picture of the state of the enterprise funds in monetary terms.

1.3.2.2.10. Exchange plans

Objects of this type are intended for organizing data exchange between two infobases, or between an infobase and an external software system.

1.3.2.2.11. Business processes and tasks

You can create formalized descriptions of typical sequences of operations performed in your company and generate lists of tasks to be performed by an employee at a given time. For example, the process of selling a product item can be represented as a sequence of invoicing, approval, receipt of cash payment, and shipment of the item from the warehouse. Different employees can be responsible for different steps of the process. At any point in time, you can review the current status of the sales process and which employees are currently required to perform any actions.

1.3.2.2.12. Registers

Registers are intended for storage and processing of data reflecting economic or organizational activities of the enterprise and having no object nature.
Registers usually store information about changes in the states of objects or other information that does not reflect the objects of the subject domain directly. For example, the registers can store information about exchange rates or information about the receipt and consumption of goods.
There are 4 types of registers in 1C:Enterprise:

- Information registers
- Accumulation registers

- Calculation registers
- Accounting registers

1.3.2.2.13. Specialized configuration objects ("Common" branch)

In addition to objects that describe the accounting object domain, a configuration contains secondary objects that do not relate directly to enterprise operations, but are closely related to the functioning of 1C:Enterprise platform. These are mechanisms for user interaction with 1C:Enterprise (command interface, filter criteria, and user access rights); auxiliary objects that describe style-based and national language-based formatting; picture libraries; application module; common modules that contain procedures and functions accessible from other configuration modules; common print form templates, and much more.

1.3.2.3. Subordinate object groups

Depending on the type of configuration object, an object can have different subordinate groups of objects: attributes, dimensions, forms, tabular sections, and so on. The content of subordinate objects depends on the object type.

Attributes contain additional information about the **object, available only within this object**.-

Tabular sectionsare sets of additional information about the object represented as a table.-

---

IMPORTANT! The number of rows in a tabular section cannot exceed 100 000.

Attributes of tabular sections- content of the tabular section of the object, **available only within the tabular section of the object**.

Forms are used to enter, view, and edit the information stored in the configuration object. Each form contains the form module in the 1C:Enterprise language.-- Visual representation allows a configuration object to interact with the user. The nature of such interaction is defined by the specialists configuring 1C:Enterprise, and determined mainly by the type of configuration object. To develop forms, a comprehensive form editor is available in Designer. This editor allows to edit all form components comprehensively. An object can have multiple forms.

Commands are used to perform operations with the object.- Commands can be independent or parameterizable.

Templates are spreadsheets, HTML, or text documents (binary and Active documents can also be used) used for quick generation of print forms of an object.-

Charts are charts of the document journal.-

Dimensions are configuration objects whose data is recorded in a register.-

Resources contain the data to be recorded in a register.-

Groups of subordinate objects cannot be deleted and do not have any editable properties.

1.3.2.4. Typed and type-defining objects

Data type is one of the main properties for some configuration objects. This property determines what kind of information the configuration object can contain. The data type of a configuration object is assigned when you create or edit object properties as part of configuration setup.

Configuration objects that support data typing are referred to as **typed configuration objects**.

Configuration objects such as `Catalog`, `Document`, `Data processor` are not typed objects because they contain complex information and include typed configuration objects.

The data types that a configuration object supports are divided into two groups.

The first group consists of **primitive data types**: `Number`, `String`, `Date`, and `Boolean`. For these types, data stored in the configuration object can be a number, an arbitrary string of characters, a date, or a logical value. `NULL`, `Undefined`, and `Type` are also referred to as primitive types (see the "Primitive data types" section of the 1C:Enterprise language help for details).

In addition, some of the 1C:Enterprise configuration objects can also form **data types**. For example, a `DocumentRef` data type can be assigned to a constant. In this event, the value of the constant will be a reference to one of the documents stored in 1C:Enterprise.

Configuration objects that can form configuration value types are referred to as **type-defining configuration objects** in 1C:Enterprise. Such objects include:

- Catalogs
- Documents
- Charts of characteristic types
- Charts of accounts
- Charts of calculation types
- Exchange plans
- Business processes
- Tasks
- enumerations.

Note that type-defining configuration objects form data types as soon as you create an object of any of these types in Designer. This forms three new type kinds: `Ref`, `Object`, and `List`. For example, when a new catalog is created in Designer, new data types appear in the list of data types: `CatalogRef.<CatalogName>`, `CatalogObject.<CatalogName>`, and `CatalogList.<CatalogName>`. These data types can be assigned to any of the typed configuration objects.

Some data can have composite type. To enable a composite data type, select the Composite data type check box in the data type edit window and specify the types that can receive data. You can also choose a special `AnyRef` type.

When you select an attribute data type, 1C:Enterprise, in addition to selecting the types defined in the application, gives the developer the option to select sets of types. Examples of type sets are AnyRef, CatalogRef, Characteristic.<name>, and so on.
Type sets, as well as the composite data types, contain a list of the types defined in the application, but unlike the composite types, this list is automatically generated as a result of metadata analysis.
For example, the application has catalogs Items and Counterparties. If you have defined the attribute of a composite data type that includes the CatalogRef.Items and CatalogRef.Counterparties types, you can also define an attribute that contains a CatalogRef type set. In both cases, you can use the attribute to store the references both to the Items catalog and the Counterparties catalog.
After adding a new Prices catalog, you can only store the references to Item and Counterparties catalogs in the attribute of composite type; however, you can use the attribute described as a type set to store references to any catalogs available in this configuration, including the Prices catalog.
When you start an application, a type set is usually converted to a composite type containing all the types to be included in the set. Therefore, in the second case, the new Prices catalog will be included in the type set.
However, a type set is not always converted to a composite data type. If a type set only includes one value type, the type set will be converted to that value type. This situation is possible, for example, when a chart of characteristic types (let's assume it's named Properties) has a single value type in the property CharacteristicsValueType. Then, Characteristic.Properties type set will be converted to the single value type that is defined for the chart of characteristic types, rather than to a composite data type that contains one type of value.
This can be important, for example, when filling check is performed for an attribute whose type is defined as Characteristic.Properties. When Characteristic.Properties is converted to a composite data type, the value must be checked for Undefined; when the Characteristic.Properties is converted to a specific value type, the default value of that type must be checked.

1.3.3. Command interface

Command interface navigates the user through configuration functionality.- The command interface is based on subsystems. The configuration developer includes applied objects into the subsystems.
Based on subsystem structure and object distribution between subsystems, command interface is automatically generated for the user. The user can view the

application structure (hierarchy of subsystems) and can run the standard commands to access the functionality of applied objects (display lists of catalogs, view documents, open reports, run data processors, and more). However, the developer can modify the default command interface proposed by the system (by changing the order or visibility of commands). The command interface editor can be used for a particular subsystem or for all subsystems.
The commands included in the command interface (opening lists, creating new objects, opening reports, etc.) are provided automatically. However, the developers can create their own commands and add them to the command interface.
The purpose of creating a command interface is to grant the users with access to the information they need to perform their duties.-

1.3.4. Forms

The combination of on-screen dialog box, module, attributes, and commands is called a **form**.
Most of the configuration objects in 1C:Enterprise can have a visual form. Generally, a form of a configuration object consists of the following components:

- On-screen dialog box used to enter and edit information.
- Module - a program written in 1C:Enterprise language. As a rule, the form module processes the information entered into the dialog box for input control, calculations, and more.
- List of attributes.
- Commands used in the form.

All of these form components are optional, meaning that none of them are required to contain data.
Forms provide interaction between the user and applied objects. The nature of such interaction is defined by a specialist configuring the 1C:Enterprise application.
A form editor that allows editing all form components comprehensively is available in Designer.

1.3.5. Modules

A module is a program written in 1C:Enterprise language. Modules are located in specific configuration points and are called at predefined 1C:Enterprise runtime points. You can use modules to describe complicated algorithms of configuration objects interaction if visual tools in Designer are insufficient.
There are several types of modules:

- Managed application
- Ordinary application module
- External connection module
- Session module

- Common modules
- Form modules
- Modules of configuration objects (constant value managers, catalogs, documents, charts of characteristic types, charts of accounts, charts of calculation types, exchange plans, business processes, tasks, reports, data processors, register record sets)
- Modules of managers of configuration objects (catalogs, documents, charts of characteristic types, charts of accounts, charts of calculation types, exchange plans, business processes, tasks, reports, data processors, information registers, accumulation registers, accounting registers, calculation registers, enumerations, document journals, settings storages)
- Modules of record sets (information registers, accumulation registers, accounting registers, calculation registers)
- Command modules

To access a module, select Open module in the context menu of the configuration object. For the root configuration object, select a managed application module, a session module, an external connection module, or an ordinary application module. Some objects (such as constants or document journals) do not have a module.

For a detailed description of modules, see section "What is a module?" in the built-in language help.

The objects modules allow declaration of variables, procedures, and functions that are accessible using 1C:Enterprise language from outside of the object and supplement the object context. These modules have procedures for handling events, such as input on the basis. They also have procedures for performing actions on an object that are initiated outside of the object (for example, printing).

The manager module extends functionality of the application managers by introducing procedures and functions in 1C:Enterprise language. You can describe methods of a configuration object (for example, a catalog) that are related to the configuration object itself rather than to an object instance in the database. A manager module cannot have variables or module body.

When functions or procedures of a manager module are declared as exported, they can be accessed through the object manager:

```
// Manager module for Counterparty catalog.
Function GetDebtorsList()
…
EndFunction

// A call from application code.
Debtors = Catalogs.Counterparty.GetDebtorsList();
```

1.3.6. Templates

A template is a 1C:Enterprise configuration object intended for generation of print forms.

Common templates of the print forms are located in the Common templates subbranch of the Common branch of the configuration tree: print forms of configuration objects (catalogs, documents, document journals, charts of accounts, charts of calculation types, charts of calculation types, registers, reports, data processors, and other objects) are stored in the subordinate `Templates` objects, as well as in external files (in this case, the `Template` spreadsheet document property must be set).

Templates can be of the following types:

- **Spreadsheet document** assumes that a standard spreadsheet document is used as a template.- The template is prepared using a spreadsheet editor.
- **Text document** assumes that a text document is used as a template.- The text template is prepared using the text template editor.
- **Binary data** assumes that binary data is used in the template.-
- **ActiveDocument** assumes the use of OLE Active document technology.- Microsoft Office OpenXML documents cannot be used for ActiveDocument templates.
- **HTML document** assumes the use of an HTML document editor.-
- **Geographical schema** assumes the use of a geographical schema prepared in the Geographical Schema editor as a template.-
- **Graphical schema** assumes the use of a graphical schema prepared in the editor.-
- **Data composition schema** assumes using a data composition schema that is prepared in the wizard.-
- **Data composition layout template** assumes using a design layout of data composition system.-
- **Add-in**assumes storing an add-in that is used in a mobile or ordinary application.-

1.4. Operating modes, interaction modes, client application types

1.4.1. Application components

When building an information system on the basis of 1C:Enterprise, a variety of components is used. These components are:

- **Client application**: a dedicated application that directly or indirectly interacts with the database in order to obtain the necessary data.- There are several types of client applications and access technologies.
- **1C:Enterprise server** (or **server cluster**): a program that forms an intermediate layer between the 1C:Enterprise client application and the database.-
- Database Management System (DBMS): a software system that provides management of physical data tables, guarantees their integrity and consistency, and provides access to this data using a special data language.-
- Web server: a software product that receives HTTP requests from clients and passes these requests for execution to the appropriate handlers for execution.- If the requests come from a 1C:Enterprise client application, interaction with the 1C:Enterprise infobase is performed during processing of requests.

The framework of any 1C:Enterprise-based information system includes one or several of the above components. The information systems can have different operating modes and different types of interaction used by client applications and access technologies.

1.4.2. Operating modes

1.4.2.1. General information

1C:Enterprise supports two operating modes:

- File mode
- client/server mode

In both modes, all applications express completely identical behavior. The file mode is primarily intended for personal use and small workgroups, while the client/server mode is intended for use in workgroups or at enterprise scale.-

1.4.2.2. File mode

File mode of infobase interactions is intended for a single user or a small number of users in the local network. In this mode, all infobase data (configuration, database, administrative information) is stored in a single file.

This mode ensures easy installation and operation of the automated system. While in file mode, operations with the infobase do not require any additional software; an operating system and 1C:Enterprise platform are sufficient.
1C:Enterprise file mode provides high integrity of the infobase and easy creation of backups. The risk of the user mistake resulting in using a wrong infobase file (for example, when copying the infobase) and bringing the system to the inoperable condition is eliminated.
In addition, backup procedure is very simple; to create a file-mode infobase backup, you simply copy a single file.
However, despite the ease of use, the file mode has limitations. You should also remember that using file mode with a web server connection is only recommended for a small number of users, provided that no time-consuming operations are performed.

1.4.2.3. Client/server mode

The client/server mode is intended for use in workgroups or at enterprise scale. It is implemented on the basis of a three-tier client/server architecture.
The client application interacts with the 1C:Enterprise server, which in turn interacts with a supported DBMS. The cluster of 1C:Enterprise servers and DBMS can be physically located on the same computer or on different computers. This allows the administrator to distribute the load between the servers if necessary.
In client/server mode, the most resource-consuming data processing operations are performed on the 1C:Enterprise server cluster. For example, when a client application initiates a complex query, the final query result is sent from the server to the client, while all of the intermediate calculations are executed on the server. Increasing server cluster performance is usually much easier than upgrading all client computers.
Another important aspect of using a 3-tier architecture is easy administration and streamlined user access to the infobase. In this mode, the user does not need to know the physical location of the configuration or database. All access is carried out through the 1C:Enterprise server cluster. When accessing an infobase, the user only specifies the cluster name and the infobase name, and the application requests for the username and password accordingly. More information about system administration can be found in 1C:Enterprise 8.3. Administrator Guide (https://1c-dn.com/library/administrator_guide_full/).http://its.1c.ru/db/v83doc/bookmark/adm
Improved data security is another important aspect of the 3-tier architecture. The data is processed by the application server, and the client application is only allowed to access the application server and the data accessible by the user on whose behalf the client application runs.
Whenever possible, 1C:Enterprise ensures identical behavior of the application for all supported DBMS. Situations when identical behavior cannot be achieved are described in the documentation; each DBMS is reviewed separately.

An important feature of the client/server mode is the ability to run the 1C:Enterprise server and database servers on different operating systems (both Windows family and Linux family).

1.4.3. Types of component interactions

In order to build an information system, it is necessary to interconnect all system components, ensuring their interaction. The following interactions between components are distinguished:

1. **Direct connection**. In this case, the client application connects directly to the database file (when in file mode) or to the 1C:Enterprise server.
2. **Connection via a web server**. In this case, the client application connects to the web server, and the web server (or, more specifically, the **web server extension**) performs all further data access activities:- directly with the database (when in file mode), or using the application server (when in client/server mode).
3. **Mobile platform** - is an ad hoc version of 1C:Enterprise intended to be installed on mobile devices running any of the supported mobile operating systems.

An information system often supports multiple access types. For example, the internal network of an enterprise might use a direct connection, while remote users access the same infobase via a web server, and external users can use mobile devices to get the required data.

1.4.4. Types of client applications

Operating modes and access types do not determine how the user accesses the infobase data.

Various client applications and technologies are used to access the data:

- **Thin client** is an application that can perform a limited set of actions on a client computer.- To perform data operations, the user calls the application server. The majority of actions creating a significant system load are performed at the server side. In the thin client, the difference between client and server code is clearly expressed. The developers need to clearly understand where the developed code is being executed and what it can and must do.
- **Web client** is a web application running in a supported web browser.- Due to the peculiarities of the web browser security model, applications that run in the web client have more restrictions than the applications that run in the thin client.
- **Mobile client** - is a client application running on mobile devices (iOS, Android, Windows). The peculiarity of this type of client application is that in

addition to the standard functionality of 1C:Enterprise, it provides access to mobile-specific features: access to camera, geo positioning, notifications, etc.

- **Mobile platform** is a special version of 1C:Enterprise, designed to operate on mobile devices with the supported mobile operating systems (iOS, Android, Windows).- The mobile platform is implemented in the thin client architecture and intended for the file-mode infobases located on a mobile device. The combination of a mobile platform and a configuration forms an **application on a mobile platform**. The mobile application does not require a permanent link to any component of the external network infrastructure. If necessary, you can implement external interactions using a variety of the mobile platform mechanisms.
- **Mobile client with a standalone mode** - is a client application managed by mobile operating systems (Android, iOS, Windows). It is a mobile client which supports a standalone mode of operation, inter alia when connection to a web server is unstable or interrupted. You need to define further settings to be able to operate in a standalone mode.

You can say that some of the client applications (thin client, mobile client, mobile application, and thick client) are fully functional under the supported operating systems, while the web client requires a web browser environment to run. The first include thin client, mobile client, mobile application, and thick client. -

1.4.5. Interface

Interaction with the application users is based on the managed interface. **Managed interface** is a declarative description of the application interface.- The application developer specifies location of the form items relative to other items and general alignment and grouping of menu commands, rather than the exact coordinates of each item on the form or the location of each command in the menu. Applications that use the managed interface are called **Managed applications**.

Declarative nature of the interface allows to use the same forms with little or no customization in different client applications and on different devices (personal computers, mobile devices).

1.4.6. Ordinary applications

In addition to the managed interface, 1C:Enterprise supports the applications that use the regular interface. **Regular interface** assumes the most precise (up to one pixel) placement of items in the form. When building a regular interface, you need to manually specify how the form items will behave in case of form resizing, and where each command will be located in the menu.

Applications that use the regular interface (referred to as **ordinary applications**) can only be used on a personal computer. These applications cannot be used to

access the infobase from web browsers or mobile devices. The ordinary applications are only accessible through thick client.-

Thick client can perform almost all the actions supported by the application.- In terms of 1C:Enterprise language, thick client allows operations with the entire list of configuration objects without requiring a server to process the data. When in the client/server mode, the line between running on the thick client and on the server side is blurred.

Development of new applications that use the regular interface **is strongly not recommended**. This technology is only **supported for compatibility** with earlier application versions.

1.4.7. Summary

Summarizing the above, the infobase access methods (depending on the interface used) are as follows:

- Managed application:
  - File and client/server access modes.
  - The client application can run under operating systems: Android, iOS, Linux, macOS, Windows (conventional and mobile).
  - You can access the Web server by using a thin client, a Web client, or a mobile client.
  - Mobile devices are supported.
- Ordinary application:
  - File and client/server access modes.
  - The client application can run under operating systems: Linux, macOS, Windows (conventional only).
  - Cannot use web server.
  - Mobile devices are not supported.

It should also be noted that, to achieve maximum lifetime for modern applications being developed, it is recommended to use managed interface. Applications that use managed interface allow to build flexible solutions compatible with the maximum number of platforms and operating systems. Using applications with regular interface, as well as running applications through thick client, is strongly not recommended. These features are only supported for compatibility with previously implemented applications.

1.5. Technological development tools

Several technological mechanisms are used in 1C:Enterprise to describe the specific data processing algorithms and to create an interface aimed at streamlined representation of the data described in the configuration.

1C:Enterprise language. The need for an embedded language is defined by the concept of system customizability. The 1C:Enterprise language syntax fully meets the standards of high-level languages.

The language is subject-oriented. It supports specialized domain data types that are defined by the application configuration. Operations with these data types are organized using object technologies. The language is aimed at users of different proficiency. The 1C:Enterprise language is distinguished by soft data typing that allows for rapid development of software modules, and strict control of syntactic constructions that reduces the probability of errors.

Since 1C:Enterprise includes both visual and language-based configuration tools, the use of the 1C:Enterprise language has an event-dependent orientation, meaning that language modules are used in specific places to test individual algorithms that are configured during the configuration process. For example, you can create an algorithm for automatic attribute filling when entering a new document. This procedure will be called automatically whenever required.

Query mechanism. For the purpose of generating arbitrary reports with complex structure, 1C:Enterprise provides a subject-oriented query mechanism. This tool relies on the existing conditionally-variable infobase structure, which allows relatively easy description of complex queries.

The built-in text editor is used to create software modules in the 1C:Enterprise language and to edit text documents.

The editor supports contextual highlighting of the syntactic constructions of the 1C:Enterprise language, as well as grouping of syntactic constructions.

Contextual hints and templates make typing code in the 1C:Enterprise language faster and easier.

Due to powerful text processing capabilities of 1C:Enterprise language, text format is sufficient to exchange a wide variety of data with other applications.

Built-in form editor. Operations with customizable data structures and operations in Windows interface require arbitrary configuration of forms for entering and editing data. A built-in form editor is available for this purpose in 1C:Enterprise.

The editor allows to arrange most of the application windows that are used for entering and viewing data (document forms, catalog forms, report settings).

Built-in spreadsheet document editor. All output documents (primary documents and reports) are maintained in a uniform spreadsheet format.-

Spreadsheet document editor is a powerful tool that combines spreadsheet design capabilities and vector graphics.- It can be used both to create small documents with a complex line structure (such as a payment order) and long lists, journals, and other similar documents.

The spreadsheet document editor provides users with a rich set of design features (customizable fonts, colors, lines, patterns). You can display data in a graphical form (as charts).

The spreadsheet document is aimed at report generation using the 1C:Enterprise language. Flexible reporting is made possible thanks to the mechanism for manipulating the named areas of the document. The spreadsheet document editor allows to manipulate both horizontal and also vertical areas, making it possible to create reports that scale both in height and in width. By applying the capabilities of the editor to an object such as a data composition system, you can create universal reports that allow to process and present data in different sections and different details without additional developer intervention.
On the other hand, a spreadsheet document can act as a form management item and thus be used for data entry.
Built-in image editor. The editor allows to create images of arbitrary sizes to use them as toolbar icons, button pictures, and for other design purposes.
Built-in HTML document editor. The editor allows to create custom descriptions and has extensive design features (hyperlink support, use of styles, placing pictures, etc.).
Wizards are auxiliary tools that facilitate the development of standard 1C:Enterprise components.- The application includes, for example, wizards for building forms for constants, catalogs, documents, document journal, reports and other objects, print forms wizards, register records wizards, and more.
Wizards are used to build visual components of these objects, and also in some cases (input on basis, printing, output forms, etc.), to create the program modules.
User interface customization system. In order for the interface of a specific configuration to fully reflect the configured data structures and algorithms, 1C:Enterprise (in addition to the editor of dialog forms and spreadsheet documents) offers customization of the system command interface.
The command interface is automatically modified according to the access rights of the user who is logged on. The user can view only those application objects they are allowed to access.
Subsystems. In Designer, you can define configuration subsystems (for example, trade accounting and research) during the development stage. For each subsystem, you can specify the configuration objects it includes. You can include an object into multiple subsystems. In fact, subsystems define the main configuration sections the user will deal with. Due to the fact that the structure of subsystems defines the configuration interface, it is necessary to pay special attention to the issues of designing (and their hierarchies).
System for configuring access rights (roles). This system allows to describe sets of rights that correspond to user jobs or activities. The structure of rights is determined in each specific system configuration. For example, such sets of rights as Chief accountant, Storekeeper, Manager, or Head of department, can be created.
In addition, for objects stored in the database (catalogs, documents, registers, and so on), you can define access rights for individual fields and records.
A list of users is created for each specific organization. Each user is assigned one or more roles, a primary interface, and an application language.

Debugger. Debugger is available in the application for easy configuration development. The debugger allows to trace execution of configuration modules, measure comparative execution times, and view the values of variables.
Configuration repository. Configuration repository functionality is commonly used for shared configuration development. It allows to allocate access rights to edit the configuration objects and make the necessary changes concurrently.
Configuration support. For convenience of updating configurations, developers are given a mechanism for creating standard configurations with delivery files and distribution packages (including the installer), and a mechanism for updating standard supported configurations.

Chapter 2. Managing configurations

2.1. General information

In this chapter, you will learn general techniques for operating with configuration objects of any type. The specifics of creating and editing different types of configuration objects are discussed further, in the relevant chapters of this Guide.
This chapter focuses on the visual tools intended for configuration object management that are available in Designer. For a description of the 1C:Enterprise language, see the 1C:Enterprise language help.
Description of the procedures for backing up and restoring infobases can be found in 1C:Enterprise 8.3. Administrator Guide.
A 1C:Enterprise infobase stores two configurations:

- **Basic configuration** (editable, hereinafter referred to as "configuration")
- **Database configuration**

The database configuration defines the structure of the database tables and the functionality accessible to users. The basic configuration is used only for editing purposes. This allows to make and save configuration changes without user interruption. You cannot make changes to the database configuration without user interruption. All main configuration commands (for both basic and database configuration) are located in the Configuration menu.
In addition, the configuration extension windows can be opened in Designer. Appearance of a configuration extension window is almost identical to the basic configuration editing window. However, actions that the user can perform with the extension configuration are not available from the Configuration menu (except for opening the list of extensions and saving the extension configuration). Commands of extension configuration operations are located in the Actions - Configuration menu of the configuration extension window or in the context menu of the extension list.
Unless otherwise specified, all actions described below are related to the basic configuration if they are executed from the Configuration menu in Designer. The commands that are available during extension management are related to the corresponding extensions. These commands are not described separately.

2.2. Opening configurations

Configuration operations are performed in the Configuration window. To open a configuration, click Configuration- Open configuration in Designer menu. The Configuration window will appear on the screen.

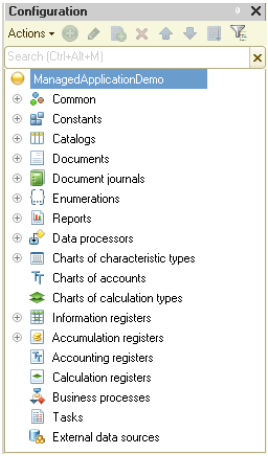


Fig. 2. Configuration

To maximize the work area for operations with multiple objects, you can temporarily close the Configuration window. Closing the Configuration window does not prevent you from working with the individual components that are available for editing. When you close the configuration window, you do not close the configuration itself. To open the Configuration window again, click Configuration -Configuration window in Designer menu.

2.3. Saving configurations

When you edit a configuration, you can create, modify, or delete objects or subordinate objects (forms, attributes, etc). Any such change results in a configuration modification. Modified configurations are marked with asterisk (*) character in the header of the Configuration window.

To save the configuration, click Configuration - Save configuration in Designer menu. This menu item is only available when the configuration is modified, unlike the text, spreadsheet, and HTML editors where this menu item is always available.

You can save a modified configuration at any moment, even if you are running in 1C:Enterprise mode or debugging mode.

This also applies to the configuration extension window.

2.4. Closing configurations

To close a configuration, click Configuration - Close configuration in Designer menu. If the configuration has been modified, Designer displays the following message: Configuration '<Configuration name>' has been changed. Save changes? Click Yes to save your changes.
You can save the changes and close the configuration at any moment, even if you are running in 1C:Enterprise mode or debugging mode.

2.5. Saving configurations to file

To save a configuration to a .cf file on the disk, select Configuration- Save configuration to file in Designer menu. This will open the standard file selection dialog box. Select a folder and enter the name for the .cf file.
The saved configuration file can be used for comparing and merging configuration.

NOTE. If the configuration is supported, the vendor configuration is always stored in the infobase.

2.6. Loading configurations from file

To completely replace the current configuration with the configuration from a .cf file, select Configuration - Load configuration from file in Designer menu.

2.7. Configuration object tree

A configuration is represented as a tree structure, where each branch describes a specific configuration component. Configuration objects in the configuration tree are represented under their own names. For example, the Documents branch contains the objects of all documents included in the configuration, while the SalesInvoice subbranch describes the object of Sales invoice document; the Common- Roles branch contains all roles (access rights) used for different types of users, while the Manager subbranch describes access rights for sales managers; and so on.--
Each root branch of the object tree contains configuration objects that are logically related and have a common purpose.
For example, the Documents branch contains objects of Sequences and Document numerators types, as well as of Document type. All these objects are intended for entering documents into 1C:Enterprise.

Keyboard shortcuts are convenient for operations in Designer. The list of keyboard shortcuts that can be used in Designer is available in the built-in Help (section Keyboard shortcuts (Designer)).

You can arrange configuration objects within a group in the order you want. This order will be used to display configuration objects in lists.

To rearrange the order of configuration objects, select an object in the Configuration window and click Actions – Move up, Actions – Move down, or Actions – Sort in the configuration window menu.---

For the convenience of searching the configuration tree for objects currently being edited (object editing window, forms, templates, modules), click Edit - Find in tree in Designer menu. First, switch to the object edit window. After the command is executed, the object currently being edited will be automatically selected in the configuration tree.

You can search the metadata tree in several ways:

- Start typing the object name (the search will only be performed in the expanded branches of the tree).
- Use the standard search functionality by pressing Ctrl + F, typing the object name, and clicking Find. This will open the Search results window, from which you can navigate to any found metadata object.
- Use a search field located between the list of objects and the command bar of the configuration window. To navigate to this field from anywhere in Designer, press Ctrl + Alt + M. To start searching, type in two or more non-space characters in this field. The search scope includes the Name, Synonym, and Comment properties of the configuration objects. The search is performed by the first occurrence of the entered string (see **Ошибка! Источник ссылки не найден.**). Only the configuration objects that match the entered string are displayed in the configuration tree; all other objects are hidden.

  When the filter by name is enabled, pressing ESC clears the input field and cancels the search if the input focus is in the configuration tree or search field. To navigate between the configuration object tree and the search field, press TAB.

  If the user adds an object that does not match the filter, the filter will be canceled and the entire configuration tree will be displayed.-

The names of configuration objects and subordinate objects (attributes, tabular sections, forms, etc.) can be dragged into modules and text documents using the mouse.

2.8. Metadata tree sorting settings

This mode is used to sort configuration objects. Configuration objects of the same type that are subordinated to the same configuration object can be sorted at one time.

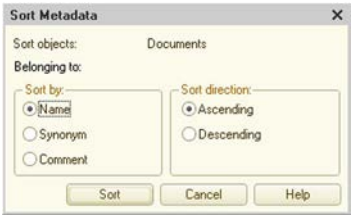


Fig. 3. Sorting metadata

For example, you can sort forms of a specific catalog.

- Sort by selects a property to sort by:-
  - Name. Sorting by the names of the configuration objects.
  - Synonym. Sorting by the synonyms of the configuration objects.
  - Comment. Sorting by the comments of the configuration objects.
- Sorting direction selects the sorting direction:-
  - Ascending. Sorting will be performed in the ascending order.
  - Descending. Sorting will be performed in the descending order.

2.9. Creating and deleting configuration objects

2.9.1. Creating configuration objects

Most configuration objects are managed in the Configuration window. In this section, you will learn the general techniques for creating configuration objects of any type.

To create a new configuration object:

- In the configuration tree, select the name of the desired type of the configuration object
- select Actions - Add of the Configuration window.

To create a new subordinate object:

- Open the object editing window by clicking Actions – Change in the Configuration window menu-
- Specify the desired type of subordinate object
- Click Add

As a result of these actions, a new object will appear on the current branch of the configuration tree, and the property palette will be automatically displayed to edit the properties of that object. For objects with multiple editable properties, the edit window might be displayed in addition to the property palette.

A new configuration object is assigned a conditional name that consists of the type of object being created and the sequence number of the object.- For example, when you create a new catalog, its name will begin with "Catalog." The property palette of the object will contain the default property values.

Designer examines the names of objects before updating the database configuration and before generating the delivery package.

For objects that can have subordinate objects (for example, a catalog can have attributes, tabular sections, forms, and templates), the required number of subordinate objects with required content is created. They are created and configured using a variety of Designer tools.

Wizards are available to facilitate creation of some object components (queries, templates, printing procedures, register records, input on the basis) in Designer.

For the types of objects that can contain forms, Designer offers form wizards that make it easier to design object forms.- **Form wizard** runs when you create a new form.

To edit forms, use the **Form editor**.

Templates that are based on a spreadsheet document are edited using the **Spreadsheet document editor**. To create templates, you can use the **Print wizard** and the **Output form wizard**.

Templates that are based on text documents are edited using the **Text document editor**.

Module code in the 1C:Enterprise language is created using **Text editor**.

You can drag objects to create new objects. When you drag an object and drop it inside or outside of its branch, a new object is created. Designer automatically checks whether properties of the original object can be copied to the new object. If the new object can have properties, they will be copied. For example, properties are copied when you transfer templates or attributes of a catalog to a document. When you copy the same catalog to an object of type `Style`, only the name, synonym, and comment will be copied.

When you drag a source object containing subordinate objects (attributes, forms, templates, etc) to the same level (for example, drag the `Catalog.Currencies` object to the `Documents` branch), the attributes, forms, templates, and tabular sections are copied. However, some properties can be changed (for example, `Parent` property is not available for documents).

2.9.2. Deleting configuration objects

To delete a configuration object, select it in the configuration tree and click Actions – Delete in the Configuration window menu.- If no references to the object are found in other configuration objects, the object is deleted.
If references to the object are found, meaning that the object is in use, this message is displayed: The object cannot be deleted because other objects contain references to it. A list of objects using this object is displayed in the message window.
fig. 4 shows a message window when you try to delete a `GoodsReceipt` document.

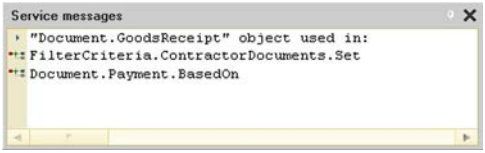


Fig. 4. Service messages

To navigate to an object that contains a reference to the object to be deleted, double-click the name of the object in the message window.
Depending on the Check modules before deletion option, when you delete a configuration object, all configuration modules will be checked for references to the objects. If this check detects the object is referenced in any 1C:Enterprise language texts, a list of reference instances will be displayed and you will be prompted to confirm deletion of the object.

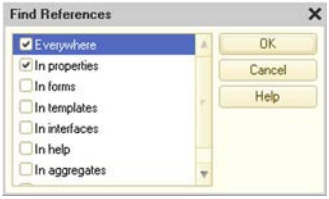


Fig. 5. Checking whether the object is referenced in modules before deletion

2.9.3. References to configuration objects

Some configuration objects contain no references (for example, Purchase price attribute of `Number` type), while some are contain references to other objects

(for example, Unit of measure attribute of the Products catalog that references the Units of measure catalog).
A list of configuration objects referencing a given object, or referenced by it, is often useful. To view the lists of objects referencing to, or referenced in, a given object, click Actions – Find references to object or Actions – Find references in object of the Configuration window menu.--
This opens a dialog box (see fig. 6).

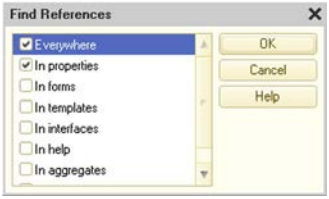


Fig. 6. References to object

Specify the search scope (properties, forms, templates, and so on), and click OK. If any references are found, the message window will display the list of found references. If no references are found, one of the following messages will be displayed:

- There are no references to this object
- Reference not found in object

The Actions – Find references in object command displays a list of all the aggregate types that are used in the object.-
To open the reference, double-click the corresponding line containing the name of the reference object in the message window.

2.10. Property palette

Editing a configuration object is changing its properties in order to achieve the desired behavior.
Property palette is a window containing a set of properties definable for a configuration object.- The list of properties may vary depending on the type of object you are editing.

This section describes the basic techniques for using the property palette.
To view the property palette, select a configuration object and click Edit – Properties in Designer menu.-
In the property palette, the object properties are grouped by category. The number of categories and the list of properties in each category may vary depending on the type of object. For example, the property palette for a document attribute contains only the General, Use, and Presentation categories. The list of properties in a category also varies depending on the type of object. Even for objects

of the same type, the lists of properties available in their palettes are determined by individual settings of each object. For example, for a hierarchical catalog, the list of properties of tabular fields in the Use category includes Presentation, Display root, and more.

The list of properties also varies depending on the selected values of other properties. For example, in the Type property, when you select a primitive type Number, the following properties describing this type are added: Length, Precision, and Nonnegative.

To quickly find a property in the property palette, use the search field. This field is accessed by pressing Ctrl + Alt + I. The search is performed by the Properties view in the property palette. To start searching, type in two or more non-space characters in this field. To clear the search field, click the Clear button or press ESC while in the search field.

The property palette toolbar contains five buttons (see fig. 7). The first three buttons control how the properties are displayed.

Fig. 7. Property window buttons

The buttons are:

- Sort alphabetically. Displays the properties of the object sorted alphabetically when enabled. The property categories are hidden while in this mode. If the display of property names is enabled, the sorting is performed by name; otherwise, by the property representation.- If the property does not have a name, the sorting is always performed by the property representation. The display of property names is controlled by the Display property names item in the context menu of the property palette.
- Sort by categories. Displays the properties of the object sorted by categories when enabled.
- Show only important elements. Displays only the important properties of the object when enabled. Displays all properties when disabled.-
- Cancel Editing. Cancels all changes made to the property text field.
- Save. Saves changes made to the property text field.

Property categories can be displayed as tabs or as a list. When you choose to view the properties as a list, the properties can be displayed by category or alphabetically (category names are not shown).
To select the category display mode (only available when display by categories is enabled by clicking Sort by categories), open the context menu of the property palette window and select a display method: Tabs or List.- If you select Tabs, the Sort alphabetically and Sort by categories buttons become unavailable.
While in Tabs display mode, click the appropriate tab to switch between property categories.
To return to the List display mode, open the context menu of the property palette window and click List.
If you select List display mode, the properties of the object are arranged by categories or alphabetically. For alphabetical arrangement, click the Sort alphabetically button on the property palette toolbar. All properties are displayed in alphabetical order.
For arrangement by categories, click the Sort by categories button on the property palette toolbar. This groups all properties into a vertical list of categories. The names of categories are displayed in **bold**. To the left of each category name, a button for expanding or collapsing the category is located. The button hides or shows all properties in the category.
Double-clicking a category name expands this category and collapses all other categories.
When you click the Show only important elements button, only the important properties of the object are displayed (regardless of the display mode: list or tabs, by category or alphabetically). To view all properties, click the Show only important elements button again.
Properties of objects can or cannot be edited, depending on the object type. For example, object properties of a text document are intended only for displaying the number of lines, number of characters, and the status of the document; they are not editable.
Each property in the property palette has a name and a detailed description. The detailed description mode is enabled by using the command Comment in the context menu of the property palette. If this mode is enabled, a detailed description is displayed at the bottom of the property palette when a property is selected. In addition to the description, the property name used to access the value from the module can be displayed.
For properties that can be accessed using the 1C:Enterprise language tools, you can configure them to display property names or synonyms. For example, a property can have synonym Value type and name `ValueType`. You can change the display mode by using the Show property names command in the context menu of the property palette. Names of properties are displayed according to the selected 1C:Enterprise language option.

IMPORTANT! Property names of the metadata objects and the Configuration metadata object are not displayed.

Method for entering values in the property palette depends on the type of property you are editing.

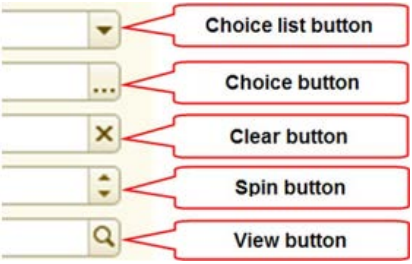


Fig. 8. Available actions

For text attributes, you type the text or copy it from clipboard; for the check box properties, you simply click them.-- Some property values are selected from lists. The fields of these properties have a **Select from list** button (see fig. 8). If a property field has a **Select button** (see fig. 8), clicking it will open a window for selecting the property value (or for viewing it when the object cannot be edited, for example, for objects that are not stored in the configuration repository, or for objects that are supported with no editing capabilities). These properties include, among others, selection of an image file, color definition, and more.

For text data, the **View** button (see fig. 8) opens the row edit dialog box in different languages; for events, it opens the form module procedure that handles the event; for properties from the Presentation category, it opens the existing form of the data object.-- The **Clear** button (see fig. 8) clears the value of the specified property. The **Adjustment** button (see fig. 8) increases or decreases the numeric value of the field by 1.

A field can have multiple buttons.

When you start editing a text field in the property palette, the property palette toolbar buttons Cancel editing (see fig. 7) and Save (seefig. 7) become available. Clicking Cancel Editing cancels the changes. Clicking Save saves the changes.

The property palette can contain links that open forms associated with the selected object. For example, you can call up reference information (description) for the configuration object, forms, and procedures in the form module. Such links are underlined. Clicking a link opens the window associated with that link.

If you view the properties of a read-only object (for example, an object that is not stored in the repository), you can open the Edit Type dialog box in view-only mode.

2.11. More window

You can use the More window to view and modify the main interface properties of a configuration object. To open it, click Edit – Advanced in Designer menu.- To view the properties of an object, select the object in the Configuration window. Its properties will be immediately shown in the More window.
The information in the window is grouped into tabs.
You can assign metadata objects to subsystems regardless of the mutual subordination of the subsystems. An object can be assigned both to the parent and the subordinate subsystem at the same time. To do this, go to the Subsystems tab.

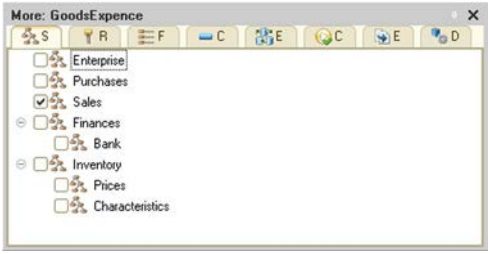


Fig. 9. "Subsystems" tab

The Rights tab lists the roles and rights of each role for the object.

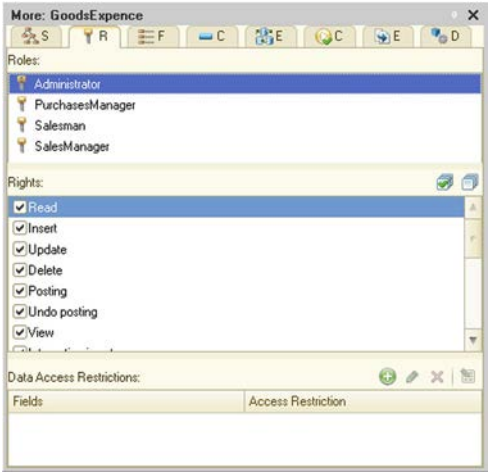


Fig. 10. "Rights" tab

In the tabular field Data access restrictions, you can restrict access to specific data at the level of individual fields and record.

The Functional options tab lists the functional options available in the application. You can check the functional options this metadata object is linked to.

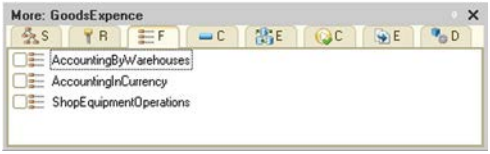


Fig. 11. "Functional options" tab

The Common attributes tab allows to specify the common attributes the object belongs to. The editing rules for this window are the same as the editing rules for the Content window of common attributes.

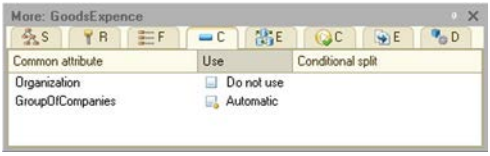


Fig. 12. "Common attributes" tab

The Exchange plans tab lists the exchange plans. The exchange plans used to record changes made to the object are marked in the list.

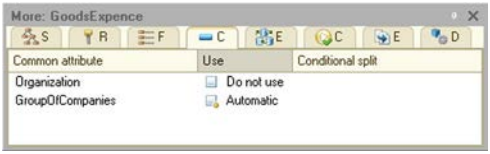


Fig. 13. "Exchange plans" tab

On the Command interface tab, you can edit visibility of standard and custom commands for the selected metadata object across different subsystems. The commands displayed on the home page are grouped in the Desktop node.

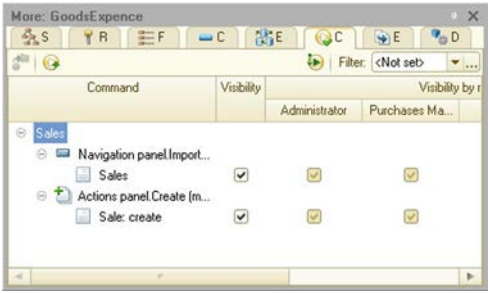


Fig. 14. "Command interface" tab

The Enter based on tab displays the list of objects on whose basis this object can be entered, and the list of objects used as the basis of exchange plans.

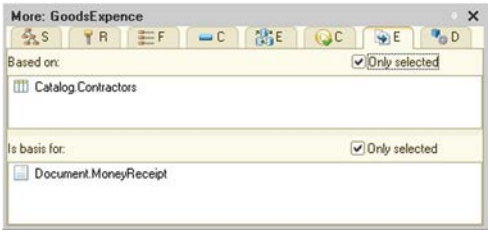


Fig. 15. "Enter based on" tab

Using the Selected only check box, you can display selected objects only, or all objects that can be entered on basis (the selected objects are checked).
On the Delivery settings tab, you can select a delivery rule for the selected configuration object, similar to the delivery settings dialog box.

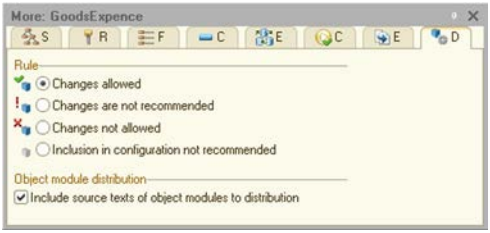


Fig. 16. "Delivery settings" tab

2.12. Object editing window

For the main configuration objects (catalogs, documents, document journals, etc), the Object Editing window is a convenient method of editing object properties, managing content of the subordinate objects, setting up interaction between objects.

To open the Object Editing window, click Actions – Change in the Configuration window menu.-

Editable properties are located in several tabs. Each tab contains a set of attributes for setting up properties of objects of a certain type. For example, in the Main tab, you can enter the Name, Synonym, and Comment properties; and in the Subsystems tab, you specify the subsystems the object is used in. You can navigate through the tabs by clicking Next > and < Back buttons, or by clicking the tabs. The form has Actions button that displays the context menu of the object when clicked. Using commands of this menu, you can open a form, open the object module (if any), start a wizard, and perform other actions.

The list of tabs and the controls on each tab may vary depending on the type of the configuration object.

For example, for a `Catalog` object, the editing window looks like this:

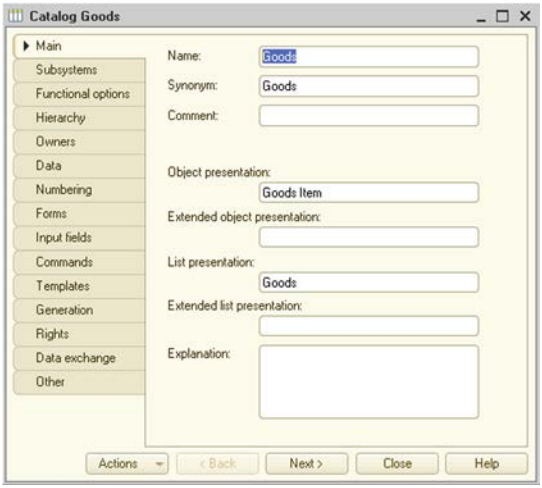


Fig. 17. "Main" tab

On the Main tab, you can specify the Name, Synonym, and Comment properties. In addition, this tab is used to define the properties that are involved in generation of the object representation in the command interface.

If you change the name of an object by editing the Name property, Designer searches the configuration for occurrences of the old name. The search is performed in string constants, queries (including the data composition system and dynamic lists), and data access restriction texts.
In the search process, there are cases when the old name is actually used - the fact of use does not need to be clarified, for example in the code there is a structure `Catalogs.Item` (when renaming this catalog), as well as cases of possible use of the old name - cases when it is difficult to well define the fact of use. If only actual usage cases were detected during the search, all replacements are performed automatically (including the object renaming). If the search process detects cases of possible use, then the next steps are configured using the Configuration Settings dialog box.
Working with the list of usages in the code of the metadata object is as follows:

The left column shows the case of using the configuration object presented in this row: "i" describes the case of actual use, and "!" describes the case of possible use.
In the far right column, the Replace... hyperlink is displayed, clicking this link will replace the old name of the configuration object with a new name in one specific location. After replacing the hyperlink will disappear. If there is no hyperlink in this column, the replacement is already performed by the application.
To speed up the replacement process, you can select multiple rows in the search results and select Replace... from the context menu.
In the Subsystem bookmark you can specify the subsystems, in which the object is used. It is possible to include an object in certain subsystems regardless of the mutual subordination of the subsystems. The object can be assigned both to parent and subordinate subsystems. The object assigning to the subsystems determines, in which parts of the command interface the commands of the object being edited will be displayed.
The Data bookmark creates the attributes, resources, dimensions, tabular sections and tabular sections attributes, as well as other subordinate objects (depending on the object type). In addition, the bookmark may contain other controls to customize properties specific to particular object types.

For example, the length of the name and code is determined for the catalog, the type of code and the main representation of the catalog item is specified.

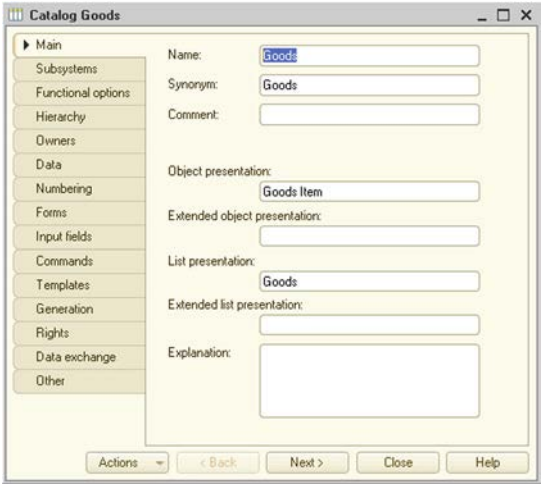


Fig. 18. "Data" bookmark

Use the toolbar buttons above the list of subordinate objects to add, remove, and arrange these objects. The properties of the subordinate objects are set in the property palette.

If the Edit window opened of the object that is not allowed to be edited (for example, the object is not captured in the repository), you can open the Edit Type dialog box in "Preview only" mode.

The Forms bookmark controls the forms of the object and selects the main forms.

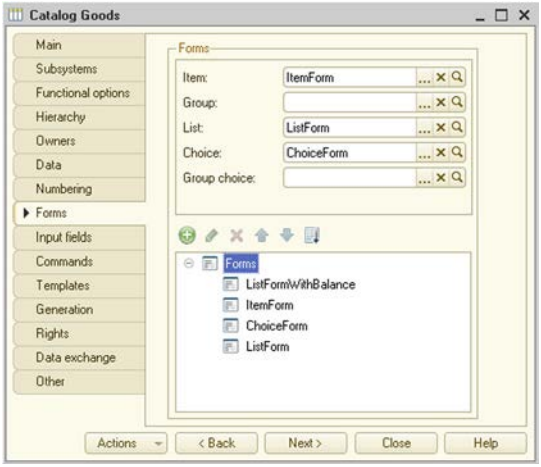


Fig. 19. "Forms" bookmark

The Input by Row group describes the behavior of this object when entering by row.
The Quick choice property is responsible for the default selection mode.
In the Commands bookmark, you can specify custom commands associated with this object.
When you add a new form, you run the form constructor to select the form view, the composition of the attributes to be placed in the form, and create the form itself.
The Templates bookmark controls the templates of the object.
When you add a new template, the Template Constructor is strarted.
Below the list of templates the Constructors button is located, clicking on which opens a submenu to select the Constructor view (the structure of the constructors depends on the object type):

- When you click Print Constructor, the print constructor is started, which creates a template and a procedure for printing.
- Selecting Output Form Constructor starts the output (reporting) form constructor.

The Rights bookmark defines the rights on objects of this type for each created role.
For application objects (catalogs, documents, charts of characteristic types, charts of accounts, charts of calculation types, registers, business processes and tasks) in the Exchange Plans bookmark you can see the list of objects of

ExchangePlan type. You should mark the exchange plans, in which record the changes to the object you are editing.
In the Other bookmark there are buttons for opening module of object, module of manager, reference information, and also the button Predefined can be located there to open list of predefined elements of object (for catalogs, charts of characteristic types, charts of accounts, charts of calculation types). The bookmark can also contain the attributes of the lock management and the settings for using full-text search. These attributes are present only for the following application objects:

- Catalogs
- Documents
- Charts of characteristic types
- Charts of accounts
- Charts of calculation types
- Registers
- Business processes
- Tasks

Some objects may contain special bookmarks that are specific to this type of object:

- for the Catalog object - this is the Hierarchy, Owners, Numbering, Input on the basis bookmarks;
- for the Document object - this is Numbering, Movements, Journals and Sequences, Input on the basis bookmarks;
- for the ChartOfCharacteristicTypes object - Hierarchy;
- for the ChartofCalculationTypes object - Calculation and Input on the basis;
- for the ChartOfAccounts object - Subaccount and Input on the basis;
- for the CalculationRegister object - Recalculation;
- for the Task object - Addressing and Input on the basis ;
- for all registers - Recorders;
- for the object that can be recorded by exchange plans- Data Exchange;
- for the external data source - Functions;
- for the external data source table- Input on the basis.

2.13. Creating Help Section

You can attach text to some configuration objects to explain the purpose and the order, in which the object is used. This text is called a user description. When operating with the 1C:Enterprise application the user can display the description on the screen for viewing.

You create and adjust a user description by using the built-in HTML editor, which you can call from the object Property Palette by clicking the Open link of the Help Information property. Creation and editing of the description text are made by means of HTML-editor. To set the title, in the chapter text create a first-level header using the <H1> markup language tag. Keep in mind that the <STYLE> tag (and its contents), which can be specified by manually editing the Help text outside the <BODY> tag, will be ignored when the Help chapter is displayed in 1C:Enterprise mode. If you want to apply a special design to the Help page, define it inside the <BODY> tag.

If the Include in Help content property is set, the item can be included in the Help content by the value of the object synonym. Regardless of setting this property, the list of indexes (for search in Help) includes a string by the value of the object synonym. Remember that the appearance of an object in the content tree depends on the value of the property of the same name in the subsystems that include the object.

2.14. Operating with Database Configuration

2.14.1. Database Configuration Object Tree

You can open the Database Configuration window to view the structure of the database configuration, properties, forms, templates, and other information about objects. To do this, select the option Configuration - Database Configuration - Open the DB Configuration window. It has the same view as Configuration window.

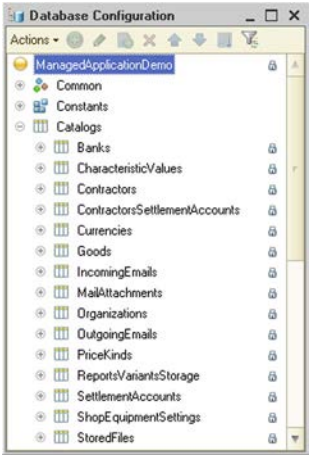


Fig. 20. Database Configuration

Methods of operating with database configuration objects coincide with the methods of operating in the Configuration window with the only difference that all objects are read-only (preview only).

2.14.2. Updating Database Configuration

2.14.2.1. General information

When you edit a configuration, you can create new objects, modify existing ones, or delete them. The current database structure may differ from the configuration structure. The difference of configurations is shown in the header of the Configuration window by symbols <!>.

The difference mark <!> of configurations will appear only after you save the changes in the main configuration. After you save the main configuration, you can continue to make changes, in this case the Configuration window header will show signs of change for both configurations.
To match the configuration and configuration of the database, update the database configuration. Provides several options of such update:

- With the help of the command Configuration -Update database configuration.
- With the help of the command Configuration -Database configuration -Update database configuration on the server.

The major difference between these two options of the update is that, in the first case, before the restructuring, the dialog listing the changes to be made will be shown to the user. The restructuring can be allowed or disallowed in the dialog. In the second case (update on the server), no dialog will be shown and the restructuring will be performed unconditionally.
If the main configuration has not yet been saved, the Designer will first save it and then update the database configuration. If a message window was opened when updating the database configuration, it is cleared.

IMPORTANT! Updating the database configuration may require all users to stop operating.

Before updating, you can compare configurations and merge them.
If at the time of the database configuration update the dubugging was performing, the following question is displayed after saving the current configuration: To update the database configuration, stop debugging. Continue? When you answer Yes, debugging stops and the database configuration is updated. When you answer No, no update is performed, and debugging does not stop.
Updating the database configuration requires exclusive access of the Designer to the infobase. Depending on the presence of users operating with the database and their modes of operation, there may be several options of application behavior:

- the Designer generates an exclusive locking error if:
  - the file mode of the database is used;
  - there are sessions connected to the infobase without using the web server;
  - there are no sessions running through the web server;
  - updating the configuration requires the database to be restructured.
- the Designer prompts you to end all sessions and retry the update if:
  - updating the configuration requires the database to be restructured;
  - Web clients or thin clients connected through the web server operate with the file mode of the infobase;
- in other cases, the Designer prompts you to perform a dynamic update.

NOTE 1. When the diagnostic messages appear, you will see the characteristics of the sessions that interfere with the action. If the number of sessions is less than

or equal to 5, a detailed list of sessions (with computer name, application type, etc.) is displayed, otherwise the total number of sessions is displayed.
NOTE 2. Switching the infobase to exclusive mode does not automatically switch the Microsoft SQL Server database to single-user mode.
NOTE 3. To speed up the infobase restructuring when using Microsoft SQL Server, it is recommended to set the recovery mode to Simple or With incomplete logging for the database. You can temporarily change the mode before restructuring, or permanently if you do not need to restore the database often. **Before changing the database restoring mode, you should backup the database.**

2.14.2.2. Situations possible during restructuring

2.14.2.2.1. Exclusive Access Error

If the application cannot obtain exclusive access, you can only wait until users are disconnected from the infobase and retry the update operation.

Fig. 21. Exclusive Locking Error

2.14.2.2.2. Session termination and attempt to update

If to update the database configuration, all sessions need to be stopped, a user receives a message.

If you select End Sessions and Retry, the user is prompted for confirmation of the selected action (Terminating sessions will cause users operations to abnormal shutdown! Do you want to end sessions?) And if the answer is yes, an attempt is made to complete all sessions of the infobase. Then, the attempt to save the database configuration is repeated.
Terminating all sessions will cause all client applications to abnormal shutdown. There may be situations when the session cannot be terminated. For example, the file mode of the infobase is published on a Web server that requires a client certificate or the Web server is configured for authentication and requires a username and password to be accessed. In this case, connections are made to the infobase only using a Web server. The Designer does not support authenticated access to the Web server and therefore cannot attempt to terminate sessions. In

the event that the Designer was unable to complete the access sessions to infobase, you can attempt to update the database configuration later or shut down the sessions in other ways.

2.14.2.2.3. Dynamic Update

If it is possible to perform a dynamic update, a special message is shown to the user (see fig. 22).

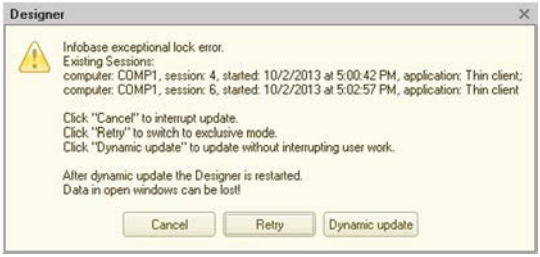


Fig. 22. Dynamic Update

If you select Update dynamically, the update is performed without user shutdown. It is assumed that the changes made will be written dynamically as a version of the configuration changes (the database configuration does not change). You can make repeated changes to the main configuration. If an exclusive mode of operation can be set on the next attempt to update the database configuration, the Designer updates the database configuration with regard to all changes (both current and previous).
If a dynamic update has been performed, the users who are running at this time continue to operate with the old configuration. In order to start operating with the updated configuration, the user needs to restart the 1C:Enterprise application. To monitor and notify users of dynamic changes that have been made, use the global scope `DataBaseConfigurationChangedDynamically()` method.

NOTE. After the database configuration update is complete, all versions created by the dynamic update will be deleted.

2.14.2.3. Restructuring completion

If changes have been detected that require a database to be restructured, a dialog box is displayed that lists the changes to confirm the update.

Fig. 23. Reorganization of Information

To confirm the saving, press the Accept button, to cancel - the Cancel button.

2.14.2.4. Restructuring mechanism selection

There are two ways of restructuring:

- Usual restructuring mechanism. The only option of restructuring in 1C:Enterprise version 8.3.10 and later.
  In this mode, the restructuring is always performed by means of copying each changed table with the subsequent restructuring of each row of data in the configurator or on the server (depending on the restructuring settings).
- Optimized restructuring mechanism. It is true only for the client/server mode of the infobase if Microsoft SQL Server or PostgreSQL is used as a DBMS.
  In this mode, an attempt to perform the maximum number of actions at the DBMS and the modification of existing data and indexes are made instead of copy and transfer of data with processing. In this mode, the better part of actions is performed at 1C:Enterprise server.

When restructuring from the configurator the used restructuring mechanism is determined by the parameter `UpdateDBCfg` in file conf.cfg. The configuration file is analyzed on the computer on which it is opened. The specification of the restructuring mechanism (simple or at the server) determines the presence of the dialog disabling or enabling the restructuring.

When restructuring with the help of the configurator package launch command line (command `/UpdateDBCfg`) the used restructuring mechanism is determined directly in the command line. If the command line contains no direct instruction as to the used mechanism the configuration file conf.cfg will be used for selection.

2.14.3. Background database configuration update

2.14.3.1. General description

NOTE. Available only for CORP licenses.

Updating a database configuration that is associated with a database restructuring that is performed for large infobases can take a long time. During the update, it is not possible to operate with the infobase.
In order to minimize the loss of time on this operation, there is a special mode that updates the database configuration in the background. The background update of the database configuration has the following features:

- Available only in the client/server mode of the infobase;
- Can be executed when the Designer is closed.
- Most of the background database configuration updates are performed without exclusive access to the database (including performing the database restructuring).
- The following operations are not available during a background update:
  - Configuration Editing.
  - Debugging an application.
  - Background database configuration update.
  - You cannot use such methods as SetAggregatesMode(), SetAggregatesUsing(), RebuildAggregatesUsing().
  - Change the contents of the chart of accounts or the chart of calculation types if the associated with it accounting register or calculation register participates in a background update operation. Attempting to modify the contents of such chart of accounts or chart of calculation types causes an error.
- Background configuration update can be paused for up to 48 hours. If the pause lasts more than 48 hours - the background update will be canceled.
- Background updates are not supported for configurations that are in compatibility mode with version 8.1.
- Background update of database configuration is not supported when operating in DBMS IBM DB2 9.1.

The background configuration update process consists of several steps:

- Processing phase**:**
  - It is running for a long time.
  - This phase can be started in any way:
    - Interactively or from the Designer;
    - Using the Disgner batch run mode.
  - During the phase, users can operate with the infobase.

    - Restructuring the main amount of data for the following configuration objects:
        - Catalogs
        - Documents
        - Document journals
        - Information registers
        - Accumulation registers
        - Accounting registers
        - Calculation registers
        - Sequences
        - Charts of accounts
        - Business processes
        - Tasks
    - During the processing phase, the application captures all changed data for the above objects, similar to the data exchange mechanisms.
- Updating phase**:**
    - Starts automatically, after the end of the processing phase, at an interval of 1 minute.
    - During the phase, users can operate with the infobase.
    - The phase consists of automatically repeating iterations. Each iteration analyzes the changes that have been accumulated since the previous iteration (or the completion of the processing phase) and restructures the accumulated changes.
    - Iterations are completed at the time of transition to the next phase.
- Changes Acceptance Phase**:**
    - Requires exclusive access to the infobase.
    - During the phase, users **cannot** operate with the infobase.
    - The first step of this phase is to update the data accumulated since the last (i. e. before the current phase) iteration of the updating phase.
    - The data that is not involved in the processing and updating phases is then restructured. These data are not expected to have a large volume of changes, and they are restructured quickly.
    - The next step is to accept all changes made to the database.
    - The database configuration update is then completed.

If the background update is running so that you do not need to restructure the database, then all updates are made in the change phase, which can be accessed immediately after the background update starts.

During a background update, you have the option to stop the server or to pause the background update process.

After the server is stopped, or after the shutdown of the working process that supports the system background job that performs the update, the first session will take a slightly longer time than usual. This is due to restoring the background update. However, the background update process itself is in a suspended state. To continue, resume the background update. This behavior is implemented to

ensure that no system looping occurs if the actual background update is the cause of the shutdown of the working process that is servicing the background update. After the server is restored, the background update of the database configuration continues as follows:

- If the operation was interrupted during the processing phase, the process continues with the last configuration object that has not been processed.
- If the operation has been interrupted during the update phase- the incomplete iteration begins anew.

TIP. It is recommended that you cancel the running background update of the database configuration when you do not plan to complete the background update of the database configuration (for example, the process is put on an infinite "pause"). The implementation of this recommendation will have a positive impact on the application characteristics, since there will be no registration of changes for the update phase.

When performing a background restructuring operation, consider some peculiarities of the application:

- If the accounting or accumulation register is added to the separator, this register is processed during the changes acceptance phase.
- If you change the type of the independent separator, all objects that are part of this separator are processed during the acceptance phase.
- If you change the type of dimension that is included in the primary selection of the independent information register, the register is processed during the changes acceptance phase.

2.14.3.2. Database Configuration Background Update dialog box

To call the database configuration background operation, select the Configuration - Database Configuration - Background Database Configuration Update menu item.

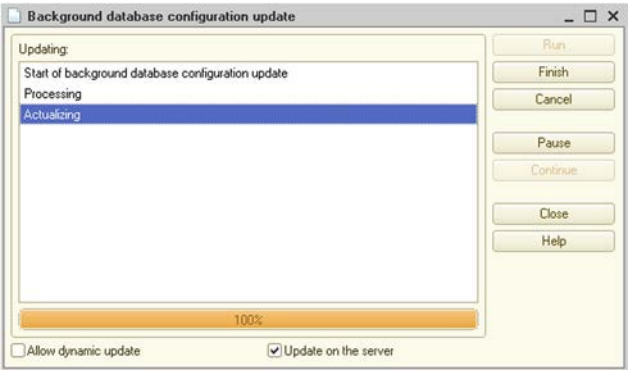


Fig. 24. Database Configuration Background Update dialog box

Clicking Start starts the processing phase. The following text is displayed in the Service Messages window:

```
Start of background update of DB configuration was successful.
Configuration is read-only. Background database configuration update
is in progress.
```

After the background update is started, the configuration is blocked from modification. During any phase, you can pause the background update using the Pause button. To continue the background update, use the Continue button.
After the processing phase is complete, the update phase begins, during which you can move the application to the change phase by clicking the Finish button or by using the Undo button.

TIP. It is recommended that you cancel the running background update of the database configuration when you do not plan to complete the background update of the database configuration (for example, the process is put on an infinite "pause"). The implementation of this recommendation will have a positive impact on the application characteristics, since there will be no registration of changes for the update phase.

The Allow dynamic update check box is used to determine whether you want to try dynamic update when you click Start instead of running the database configuration background upgrade process. If this check box is selected, the option to perform a dynamic update of the configuration will be checked immediately

after you click Start. If the configuration allows this option, a dynamic update will be performed instead of a background update of the database configuration. The Run on server check box is used to determine where to start, complete, and cancel. If this check box is cleared, these actions will be performed on the client application side, and if it is selected- on the server side of 1C:Enterprise. Additionally, if this check box is selected, you can perform a configuration update only if you have the update right in the `UpdateDataBaseConfiguration`, without the installed right `Administration`.

2.14.4. Saving database configuration to file

To save the database configuration to a file on the disk, select Configuration - Database Configuration - Save DB configuration to file. This will open the standard file selection dialog box. Select a directory and specify the name of database configuration file.
The saved database configuration file is required for the configurations compare and merge operation.

2.14.5. Comparison of configuration and database configuration

If during changes to configuration you need to get a report on differences of the database configuration, select Configuration - Database Configuration - Compare, merge with DB Configuration option.
If necessary, you can restore the modified objects.

2.14.6. Cancel configuration changes

To opt out of configuration changes, simply select the Configuration - Database Configuration - Return to the DB configuration.

NOTE. The Save DB Configuration to File... and Return to DB Configuration menu items are available even in case of closed editable configuration. The Return to DB Configuration command is still unavailable when the infobase is connected to the configuration repository.

2.15. Starting 1C:Enterprise

The Designer provides the start of the 1C:Enterprise mode. To do this, select the item Service - 1C:Enterprise. It is often necessary to start 1C:Enterprise in debug mode. To do this, there is the Debug - Start Debugging command.
If the configuration has been modified (changes have been made), the Designer asks the following question: The configuration you are editing is different from the database configuration. Update the database configuration? Select Yes to save your changes.
If you select No, the 1C:Enterprise mode starts without saving the configuration. In case of No the following question is displayed: Database configuration does not match stored configuration. Continue? If you select OK, 1C:Enterprise mode with the previous database configuration starts. If you select Cancel, the 1C:Enterprise mode will not start.

2.16. Unloading and Downoading Configuration Files

The mechanism of unloading and loading of configuration files allows to perform selective unloading/loading of some properties of configuration objects (modules, templates and reference information). To unload the properties, select Configuration - Unload Configuration Files.

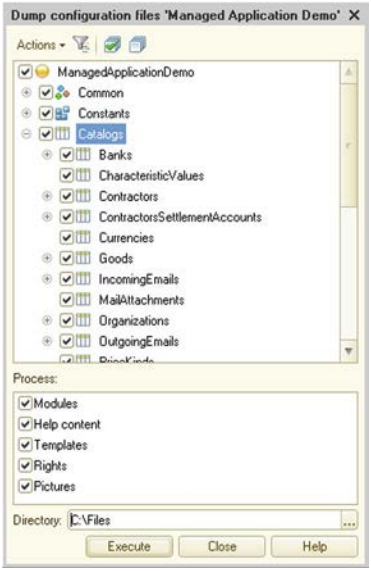


Fig. 25. Unload Configuration Files

In the window that opens, select the configuration objects that need to be unloaded, specify the type of data to be unloaded and specify the directory, to which to unload. To start unloading, click Run. Unloaded data is written to files with names corresponding to the name of the unloaded property. File extension corresponds to the file type:

- htm - for HTML document help and templates;
- txt - for text document modules and templates;
- mxl - for spreadsheet document templates;
- geo - for geographical schema templates;
- grs - for graphical schema templates;
- bin- for binary data templates.

Images are unloaded in the format, in which they are stored in the configuration. Images of all supported formats can be loaded.

To unload/load permissions, select the required roles (or all) and specify the Rights item in the Process list. Access rights are unloaded in XML format. The upload file includes the access rights themselves, data access restrictions, and access restriction templates.
When you unload the Help and templates files of HTML documents, a separate file is generated for each language, for which the emplate is defined. The language code is included in the file name before its extension: for example, the Document. GoodsConsumption.Help.ru.htm means a file with the help information in the Russian language of the GoodsConsumption document.
The data is loaded by selecting the Configuration - Load Configuration Files. The actions in the window are similar to the actions described for the data unload mode.
Loading process starts attempt to load Help files and HTML document templates for each language defined in the application. If only one file is found during loading (without specifying the language code in the name), it is loaded for the configuration current language.

2.17. Dumping configurations to files Restoring configurations from files

2.17.1. General information

This mechanism allows to unload all configuration objects into XML files and restore from them. All configuration objects are unloaded to one or several files. The main file in the unload is the Configuration.xml file, which describes the properties of the configuration itself. Unloading can be generated in either of two formats: linear and hierarchical. It is possible to perform partial loading from files. You can unload the configuration by using the command Configuration- to Unload the Configuration into files. Loading - using the command Configuration - Load configuration from files.
You can unload and load the configuration extension. In this case, you should use the menu commands Actions - Configuration (from the window with the configuration extension) or the corresponding commands in the Configuration Extensions dialog box. The extended or controlled properties will be unloaded.
If there are references to missing configuration objects in the configuration objects, the references are unloaded as a text representation of the identifier and will also be restored to the reference of the missing configuration object (to preserve the identity of the two configurations). For the attributes and elements of the managed form, internal identifiers are unloaded, which are used, in particular, to save and restore settings.

When you unload roles, the values of only those rights that differ from the default values at the time of unloading are unloaded. When defining default rights values, the properties of the Set rights for new objects and Set permissions for default attributes and tabular sections are used. The values of these check boxes are the default values for access rights to configuration objects and access rights to attributes and tabular sections, respectively.

NOTE. Ordinary forms and interfaces of a ordinary application are unloaded in binary form.

If during loading the errors are detected in the files, a window with a list of errors will be generated. If the errors are not critical, click Accept. By selecting the error string - you can open the file that contains the error.

2.17.2. Unload format versions

The Unload files are versioned. The version number of the unload format, depending on the version of the 1C:Enterprise application, is shown in the following table.

Platform version	Format version
8.3.6	1.0
8.3.7	2.0

Explicit specification of the version number in the upload files itself begins with version 2.0 (unloading from the platform version 8.3.7 and later). If the upload files do not explicitly specify the version number, the upload is considered to have the version number 1.0.
The format version is the same for all configuration objects that can be unloaded to files. The format version is written to the root XML files of the configuration objects, as well as to the root XML files of some subordinate configuration objects. Unload files that have a format other than XML do not contain an indication of the version of the unload format.
Unloading is always done in the format of the current version of the 1C:Enterprise platform (the one, from which it is unloaded).
The download is successful only if the version of the downloaded files is less than or equal to the version of the 1C:Enterprise platform, with which the upload files are loaded.

2.17.3. Unloading Formats

2.17.3.1. Hierarchical unload format

When unloading in hierarchical format, all configuration objects are unloaded into a hierarchical file structure that is generated according to some rules. All

files and directories are generated within the directory specified in the upload dialog box.
This directory will be called the **Root unload directory**. It is recommended that you do not specify the path to the upload root directory if it is longer than 30 characters. This will reduce the likelihood of the unload being aborted due to an error associated with exceeding the path to the unloaded file.
All configuration objects are located in subdirectories that correspond to their types. For example, directories will be in the Catalogs subdirectory of the root unload directory, documents - in Documents, and so on. The configuration objects located in the General branch will also be unloaded in their subdirectories of the root unload directory, without an intermediate subdirectory. The properties of the entire configuration are uploaded to the Configuration.xml file, which is placed directly in the root unload directory.
Each object is unloaded as follows:

- The object description is unloaded in an XML file with the name of the object. If an object has subordinate objects (attributes, commands, forms, and so on), a subdirectory with the object name is created in the same directory.
- Subdirectories for all groups of subordinate objects, except attributes and tabular sections, are allocated in the subdirectory of the object. Attributes and tabular sections are located in the xml file with the description of the object itself. Thus, the following subdirectories are displayed:
  - Forms - subdirectory Forms;
  - Commands - subdirectory Commands;
  - Templates - subdirectory Templates;
  - External data source cubes - subdirectory Cubes;
  - External data source tables - subdirectory Tables;
  - Other properties of the configuration object (including the configuration itself) are placed in Ext subdirectories of different levels of subordination.
- If the subordinate object has a hierarchical structure - it is unloaded similar to the parent configuration object.
- The Ext subdirectory can contain:
  - For the configuration itself:
    - Managed application module - ManagedApplicationModule.bsl.
    - Ordinary application module - OrdinaryApplicationModule.bsl.
    - External connection module - ExternalConnectionModule.bsl.
    - Session Module - SessionModule.bsl.
  - For the configuration object:
    - Object Module - ObjectModule.bsl.
    - Command Module - CommandModule.bsl.
    - Object Manager Module - ManagerModule.bsl.
    - Record Set Module - RecordSetModule.bsl.

        - Value Manager Module - ValueManagerModule.bsl.
        - Predefined data - Predefined.xml.
        - Schedule of scheduled job - Schedule.xml.
        - Right of the role - Rights.xml.
        - Template File - Template.xml.
        - Help Information In this case, a general description of the help is in the Help.xml file. Help files named <language code>.html are located in the Help directory. If the Help files contain absolute images - they are located in the _files subdirectory of the Help directory.
        - Graphical schema or business process diagram. In this case, a general description of the schema is in the Flowchart.xml file. The Flowchart directory contains subdirectories with a description of the graphical schema elements that are located in the Items subdirectory. Each element is represented in the form of a subdirectory, inside which is the actual absolute picture.
    - For the subordinate configuration object:
        - Managed Form Module - Module.bsl.
        - Managed Form description - Form.xml.
        - Absolute pictures of the managed form elements.
        - Absolute pictures stored in common pictures (Common- Common pictures).
        - Help Information The subordinate object is stored similar to the reference information for the configuration object.
- The form is unloaded in the following structure:
    - The description of the form is unloaded into an XML file with its name.
    - It also creates a subdirectory with the name of the form that contains the Ext subdirectory.
    - The Ext subdirectory contains:
        - Managed Form description - Form.xml.
        - Form subdirectory.
        - The Form subdirectory contains a file with the form module - Module.bsl.
        - The Ext subdirectory contains the reference information of the unloaded form (described above).
- The template is unloaded in the following structure:
    - The template description is unloaded in an XML file with the name of the template.
    - It also creates a subdirectory with the name of the template that contains the Ext subdirectory.
    - The Ext subdirectory contains the following information:
        - Template.txt - template, placed as a text document;

        - Template.xml - template, placed as an HTML document (with further structure of similar reference information);
        - Template.bin - templates in all other formats.
- External data source cubes are unloaded to the following structure:
    - The external data source itself is unloaded into an XML file with the name of the source;
    - It also creates a subdirectory with the name of the external data source that contains the Cubes subdirectory;
    - Cubes are unloaded as follows:
        - Each cube is unloaded in an XML file with the name of the cube;
        - It also creates a subdirectory with the name of the cube that contains the DimensionTables subdirectory;
        - The DimensionTables subdirectory contains files describing the dimension tables with their own names.
- External data source tables are unloaded to the following structure:
    - The external data source itself is unloaded into an XML file with the name of the source;
    - It also creates a subdirectory with the name of the external data source that contains the Tables subdirectory;
    - Tables are unloaded as follows:
        - Each table is unloaded in an XML file with the name of the table;
        - A subdirectory Tables is also created.
        - Each table is unloaded as a separate configuration object.
- The subsytems are unloaded in the following structure:
    - Subsystem is unloaded as an ordinary configuration object.
    - If the subsystem has subordinate subsystems, then in the directory named subsystem the Subsystems subdirectory will be created, where the subordinate subsystems will be unloaded.
    - It is not recommended to use more than three nested subsystems, including the top-level subsystem, in order not to break the restriction on the full path to files of subsystem objects (255 characters). If the application requires the use of more than three nested subsystems- it is recommended that the subsystems have the shortest names.

The following is the mapping between the configuration object and the directory, to which the object will be unloaded:

Export subfolder name	Object
AccumulationRegisters	Accumulation registers
BusinessProcesses	Business processes
Catalogs	Catalogs
ChartsOfAccounts	Charts of accounts
ChartsOfCalculationTypes	Charts of calculation types
ChartsOfCharacteristicTypes	Charts of characteristic types

Export subfolder name	Object
CommandGroups	Command groups
CommonCommands	Commom commands
CommonForms	Common forms
CommonModules	Common modules
CommonPictures	Common pictures
CommonTemplates	Common templates
Constants	Constants
DataProcessors	Data processors
DefinedTypes	Defined types
DocumentJournals	Documents journals
Documents	Documents
Enums	Enumerations
ExchangePlans	Exchange plans
Ext	Additional data
ExternalDataSources	External data sources
FilterCriteria	Filter criteria
FunctionalOptions	Functional options
FunctionalOptionsParameters	Functional option parameters
HTTPServices	HTTP services
InformationRegisters	Information registers
Languages	Languages
Reports	Reports
Roles	Roles
ScheduledJobs	Scheduled jobs
Sequences	Sequences
SessionParameters	Session parameters
SettingsStorages	Settings storages
StyleItems	Style items
Styles	Styles
Subsystems	Subsystems
Tasks	Tasks
WebServices	Web services
WSReferences	WS references
XDTOPackages	XDTO packages

2.17.3.2. Linear export format

IMPORTANT! The linear export format is obsolete. Using it is not recommended. We recommend to use Hierarchical unload format instead.

The following is unloaded to the separate files:

- Configuration objects- top-level objects, and subordinate objects that are separately captured in the configuration repository.
  Examples of file names:
  - Document.GoodsSales.xml - `Document.GoodsSales` configuration object description.
  - CommonPicture.SalesSubsystem.xml - `CommonPicture.SalesSubsystem` (but not the picture itself) configuration object description.
- Separate properties - tend to have a large volume or complex structure.
  Examples of file names:
  - CommonPicture.SalesSubsystem.Picture.xml - picture description (but not the picture itself).
  - DataProcessor.Guide.Template.MainPage.xml - description of the template, located in the data processor (but not the template itself).
- Separate properties values that are files of known formats (but not XML). These files include form modules, templates, pictures.
  Examples of file names:
  - CommonPicture.SalesSubsystem.Picture.Picture.png - picture saved in Common Pictures with the name `SalesSubsystem`.
  - Catalog.Goods.Form.ItemForm.Form.Module.txt - module of `Goods` catalog item managed form.
  - DataProcessor.AssistantForDataExchangeCreation.Form.Form.Form.SecondInfobasePicture.Picture.png - absolute picture contained in the property `Picture` of the item of the managed form`SecondInfobasePicture`.

Objects that are not development objects are stored in the parent object file, such as the file Dataprocessor. PrintLabelsAndPricetags.Template. PrintBarcodesComponentLinux32.Template.bin means the print add-in for labels and price tags located in the `PrintBarcodesComponentLinux32` binary template.

The following elements are unloaded in a special way:

- The object module - is unloaded as a text file (if it can be obtained) or in binary form.
  An example of file name:
  - DataProcessor.EventLog.ObjectModule.txt - metadata object module `Processing.EventLog`.
- HTML - is unloaded as a set of files: property root file, HTML page files, image files (if any). The root file contains a list of HTML pages included in the document. HTML files are pages included in the document. Each corresponds to a specific language defined in the configuration. Configuration object Help is a special case of HTML unloading.

Examples of file names (for the Help property):

- Catalog.Goods.Help.xml - root file of the external property –metadata object Help `Catalog. Goods`.
- Catalog.Goods.Help.ru.html, Catalog.Goods.Help.en.html – -Help pages for the Russian and English languages, respectively.
- Catalog.Goods.Help.Picture.png - pciture file that is included in the HTML document with Help.

- Vendor configurations - unloaded as a vendor configuration information file in the internal format (non-XML) and configuration files (.cf) for each vendor configuration.
  Examples of file names:
  - Configuration.ParentConfigurations - information file on vendor configurations.
  - Configuration.ParentConfigurations.TradeManagement.cf - vendor configuration `TradeManagement`.

When unloading, all references to metadata objects, field references, type IDs, and so on are unloaded as names (`Catalog.Goods`). All IDs are unloaded in English -reference to the `Goods` catalog will look like `Catalog.Goods`, not `Catalogs.Goods`.

When using linear format, consider the following possible issues:

- the complexity of migrating the Unload directory between different file systems;
- problems with the uniqueness of filenames of unloaded objects in some cases.

2.17.4. Unloading

The configuration can be unloaded in two different formats: linear or hierarchical. Only the hierarchical format is available for unloading from the Designer. Unloading in linear format is obsolete, not recommended and can only be performed using the Designer batch mode.

After you specify the directory and upload format, the actual upload is performed. During the upload process (as mentioned earlier), a Configuration.xml file is generated that contains the properties of the configuration itself. However, in addition to this file, the application generates a version file (ConfigDumpInfo.xml) that contains information about the name of the object and its version. If, in the future, the current configuration attempts to upload the configuration to files in this directory, the Designer will unload only those objects whose version is different from the version, which was during the previous unloading. The "version" in this case refers to a sequence of characters that will be different from the previous version (a sequence of characters) in case the object was edited. This is the only purpose of this information.

If the unload directory does not contain a version file, the configuration will be fully unloaded (with generating this file). The full unload will also be performed if any of the following changes are detected during the upload updating:

- Renaming the configuration itself;
- Renaming or deleting the configuration object;
- Changing properties `PredefinedData` or `BusinessPocessDiagram`. The configuration can have references to the elements of the predefined data and to the business process points, so changes in these properties affect the integrity of the configuration unload.

Remember that the upload format in the directory selected for upload must be the same as the current upload. In other words, if you unload in a hierarchical format and then try to unload the configuration in a linear format in the same directory - you will receive an error and the unload will not occur.

2.17.5. Loading

2.17.5.1. General information

The loading can be divided into two types, significantly different in operating time:

- full loading;
- partial loading.

When performing any loading, remember that the application does not validate the support settings when loading from XML. Accordingly, when performing this operation, the user is responsible for ensuring that the support settings are correct. However, it should be understood that downloading incorrect support settings may cause the configuration update mechanism to fail, as well as the configuration itself to fail.

2.17.5.2. Full loading

Full load replaces the main configuration with the data that is located in the downloaded files. Full loading of a sufficiently large configuration can take a significant time.

When you perform a full download of the configuration from the files, the following features are available:

- a configuration that does not support and is not connected to the configuration repository can be downloaded without restrictions;
- a configuration that is connected to the configuration repository cannot be loaded (the corresponding menu item is not available).
- a supported configuration can only be downloaded if the following conditions are met:
  - the possibility of changes is enabled;
  - all configuration objects are editable.

2.17.5.3. Partial loading

2.17.5.3.1. General information

Full configuration loading from XML files takes a significant time. However, the changes do not always affect most configuration files. Often, only some configuration objects require loading. To speed up loading, in such cases there is a partial loading of configuration from XML files.

Partial loading can be performed on both the configuration object itself and the subordinate objects (forms, Help, modules, etc.). If an object is loaded, it is loaded completely, along with all subordinate objects. Objects from files with configuration objects are mapped by their full name and internal ID. For subordinate objects, the parent object is defined by the file name (linear format) or directory hierarchy (hierarchical format).

Partial loading can only be performed by using the `/LoadConfFromFiles` key of the Designer batch-run and is not supported interactively. Also, partial loading is not supported when operating with multiple extensions.

The general procedure for partial loading is as follows:

1. The files that will be loaded into the configuration are prepared. The file format and location fully corresponds to the upload files in the selected format: linear and hierarchical.
2. The Designer is run in batch startup mode, with the list of changed files and the format of the files to be downloaded.
3. If the loading error is not detected - the configuration record will be performed. If errors are detected during loading, the last saved main configuration will be reverted to. If there were changes and unsaved data in the configuration you attempt to download - they will be lost.

When you perform a partial load, the `/Out` command-line key can help you get a file that contains the diagnostic messages that are generated during the partial loading.

When you perform a partial download of the configuration from the files, the following features are available:

- loading is possible only if all loaded objects are captured in the configuration repository;
- loading is possible if all loaded objects are enabled for modification (for a supported configuration).
- In case of configuration root element partial download, the configuration is not longed supported if the external properties file ParentConfigurations.bin (keeps the information on the support settings) is absent.

The following details describe different situations of partial load.

2.17.5.3.2. Adding Objects

The object is added by loading two objects: the actual added object and its parent. When you try to load a new object, an error may be detected:

- If the name of the loaded object is not unique;
- If the object has a reference to a subordinate object, and no files are provided for the subordinate object.

If there are no files for the subordinate objects and there are no references to these subordinate objects - these objects will accept the default values.

2.17.5.3.3. Object Deletion

To delete an object, load a parent object file that does not contain information about the object being deleted.

2.17.5.3.4. Object Renaming

To rename an object, load the object itself (with all the subordinates) and its parent (with all subordinates). The parent must contain references to the object with the new name. Remember that if you are downloading files with subordinate objects - these objects will accept the default values (if they are not referenced) or a download error will be generated if there are any.

2.17.5.3.5. Object Changing

If you need to change the object in any way, you need to load the information about the modified object and all its subordinate objects. Remember that if you are downloading files with subordinate objects - these objects will accept the default values (if they are not referenced) or a download error will be generated if there are any.

2.17.5.3.6. Subordinate Objects Loading

When loading subordinate objects, the loaded objects should be described in exactly the same way as when unloading these objects. For a linear unload, this means that the file names must match the same name when it is unloaded, and the directory hierarchy must be the same for the hierarchical unload - in addition to file names, the directory hierarchy must be the same.

For example, if you want to load the reference information for the Goods directory, you need to specify the following download command:

```
1cv8 DESIGNER /IBName "IBName" /LoadConfigFromFiles D:\Source  files
D:\Source\Catalogs\Goods\Ext\Help.xml
```

In this case, on hard drive D: the following directory structure must be created (for the case of hierarchical unloading):

```
\Source
  \Catalogs
    \Goods
      \Ext
        Help.xml
        \Help
          ru.html
          en.html
```

When loading, only the name of the loaded object is specified (in this case it is reference information - Help.xml), subordinate objects (Help files in the appropriate languages) must be present on the hard drive and will be used automatically.

The command to load the module of the Counterparties directory object will look similar. Loading Command:

```
1cv8 DESIGNER /IBName "IBName" /LoadConfigFromFiles D:\Source  files
D:\Source\Catalogs\Counterparties\Ext\ObjectModule.bsl
```

In this case, on hard drive D: the following directory structure must be created (for the case of hierarchical unloading):

```
\Source
  \Catalogs
    \Counterparties
      \Ext
        ObjectModule.bsl
```

2.18. Configuration extensions

The list of connected configuration extensions is opened by using the Configuration - Extension Configuration command in the main menu. In the window that opens, you can create new extensions and manage connected ones.

2.19. Mobile Application Operation

2.20. Mobile Client Operation

2.21. Configuration Report

Designer can display information about all configuration objects in text or tabular form. To do this, select Configuration - Configuration Report.

This opens the following dialog box. Select the type (text or table) and the name of the file, in which you want to save the description of the configuration structure.

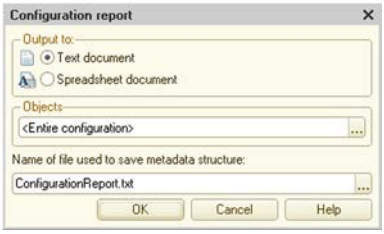


Fig. 26. Configuration Report

Creating a description of configuration objects for complex configurations can take a long time.
If you want to get a report on individual configuration objects, in the Objects field press the selection button and select the check boxes for the desired objects in the object selection dialog box.

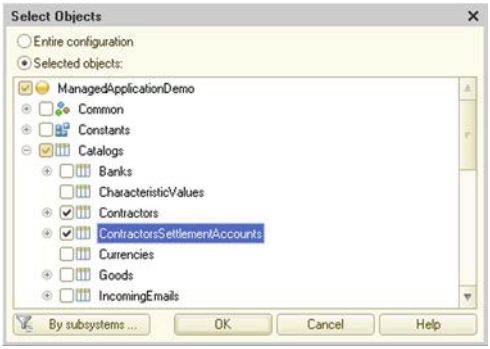


Fig. 27. Selecting objects for the configuration report

Subsystem filter is available.
After the configuration description is completed, a window will open with the description in the selected format (text or tabular).

2.22. Using Global Search and Replacement

The global search and replacement mode is designed to find a specific string in all modules, dialog boxes, spreadsheet documents, configuration descriptions,

and external files (external reports and processing, spreadsheet documents). The found text can be replaced by another one. This mode can be used, for example, to find all calls to a certain global procedure or to refer to any details in different modules.

The search mode is called by selecting Edit - Global Search, and replacement mode - by selecting Edit - Global Replacement.

These modes use the same dialog box. If the search mode is selected, the replacement mode details become unavailable. Therefore, for brevity, we consider the procedure of global replacement, and then specify the features of the search mode.

This opens the replacement dialog box.

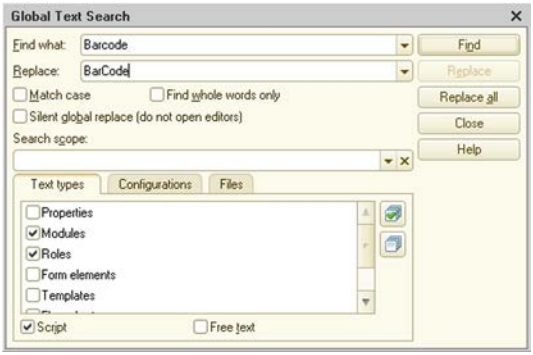


Fig. 28. Global Replacement Window

In the Search field for this dialog box, enter a search pattern or select one of the samples that were previously used in the search operations from the history list.

In the Replace field, enter the text that you want to replace the found text with, or select one of the samples that were used earlier in the replace operations from the history list.

To make the search case-sensitive, select the Match case check box. To find only the whole words, select the Match whole word only check box.

If you do not want to open editors during group substitution (by clicking the Replace All button), select the Do not open editors for group substitution check box. In any state of the check box, the editor will open when you press Search or Replace.

Below is a panel, on which tabs indicate where to search for the specified pattern. On the Text Types tab, the types of objects that will be searched are marked. If you are editing the configuration for the Managed application startup mode, the user interfaces will be excluded from the list of objects.

On the Configurations tab, you can specify the configuration sections to be searched with precision to the object.

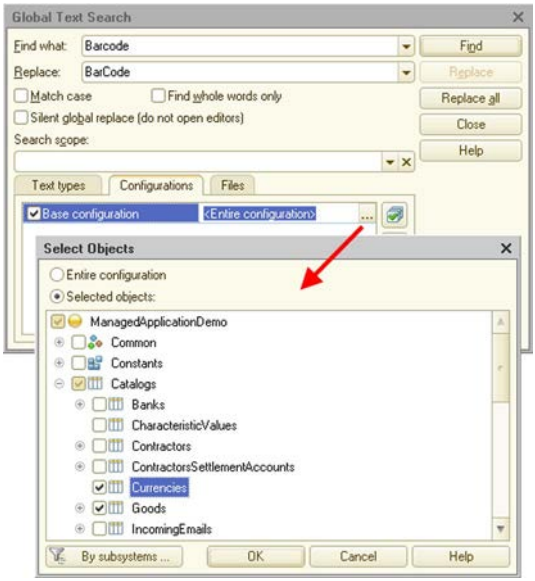


Fig. 29. Global Search

In addition to the main configuration list, the configurations of the database, repository configurations, extension configurations (if they are open) and configurations of extensions stored in the database (for open extensions) are included. Repository configurations must be open before the search or replace mode is called. Database configurations (main and extension) are available only when using global search.

To specify a set of objects, select the Selected Objects radio button and mark the objects to be searched. The first time you start the list, all objects are marked by default. To remove the marks, clear the check box on the configuration name line. You can then specify specific objects to search.

On the Files tab, you can specify the directory and file types in which the search should be performed. The following types of files can be viewed: configurations located in files (saved, delivery files), external reports and processings, text and spreadsheet documents. If the directory is not specified (the Catalog attribute is not filled), the search in files is not performed. The search can also be done in open documents of the same types. To do this, select the Search in open documents check box.

You can save the selected settings composition for further use. To do this, specify the name of the setting in the Search scope field. To switch to a previously saved assortment of settings, select a name in the drop-down list. The following settings are saved: settings on the Text Types tab, the components of objects for the main configuration only on the Configurations tab and settings on the Files tab.

If you started the search mode, you need to click the Search button to start the search.

In global search mode, you can interrupt the process by pressing Ctrl + Break.

The Search Results window displays a list of found occurrences of the source text.

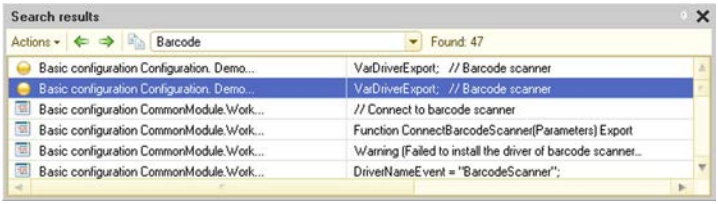


Fig. 30. Global Search Result

If any module has access restriction, the application prompts for the access password before searching for source text in this module. Enter the correct password or refuse to enter a password. If the password is not entered, then viewing in this module is not performed.

You can view the search result, and you can navigate to each value found if you select the required line and press Enter. To view the next or previous found value, you can use the Actions - Next position and Actions - Previous position.

The search result (the entire list) can be memorized to the clipboard by using the Copy command on the context menu of the window, or by using the corresponding toolbar button on the Search Results window, and output to a spreadsheet or text document.

Column width can be changed - by using the mouse pointer while pressing the Ctrl key.

If you have started the Replacement mode, the To text field shows the sample text to replace the source text that you specified in the Replace with field.

If you want to see the source text before you replace it, click Search to start the search. The result of the first found source is displayed. If you click Search again, the current text is skipped and the next occurrence of the source text in the current window or another window containing the source text is displayed.

Group replacement (without confirmation of each replacement) will be performed by clicking the Replace All button. If you do not want to open the objects

where the source code occurrence is found, select the Do not open editors during group replacement check box.

IMPORTANT! You cannot change the search criteria while viewing the search results.

The selection structure of the search scope (text types, list of configuration objects, files, and open documents) is remembered and restored when you open dialog box next time. If you want to save multiple scopes, name each area in the Search Scope attribute. If you reopen the search window in the list of scopes, select the desired one and perform a search.

2.23. Customizing the work area of Designer

To create and edit a configuration, you need to use multiple windows at the same time. For example, you view, select, add, and delete configuration objects in the Configuration window; editing of properties of objects and their components is made in the Property Palette; receiving background information on the 1C:Enterprise language - in the syntax-assistant window; messages and search results are displayed in the message window. In addition, each configuration object can generally consist of different parts, each of which is edited in a separate window. Simultaneous opening of various service windows noticeably narrows the work area intended for editing of applied objects (forms, modules and templates) and editing of common configuration objects (application module and common modules, templates, styles, interfaces, etc.).
To extend the work area and usability, you can use some guidelines for customizing different Designer panels, the presence and behavior of various service windows, and the use of window display modes.

2.23.1. Panel customization

The Designer is designed to maximize the work area by automatically selecting the desired toolbars for each window view. For example, when you edit a spreadsheet document, the Designer provides a toolbar to execute commands for editing the this document; when you navigate to a window that contains a form module, the Designer closes the spreadsheet document toolbar and displays the Text Editor toolbar.
The user can customize the components of toolbars and their placement on the screen. If you are configuring a standard command panel to place commands that are not present in this panel by default, it is possible that after the Designer is restarted, the command bar order will be changed. In this case, hide the standard panel and instead create an additional command panel, placing the required commands on it in the correct order. The order of commands on it will not change.

In addition to the toolbars, the Window Panel and the Status Bar are located below. Each of them occupies a separate line and cannot be moved to another location. If any panel is not needed, you can hide it using the context menu anywhere in any panel. The panel is displayed if the check box is selected to the left of its name and is hidden if the check box is cleared. To change the display mode of the panel, select the line with the panel name in the context menu.
If you do not want the window bar to always occupy a part of the application work area, you can enable Autohide mode. When you operate, the window panel is hidden. To show it, move the pointer over the window bar location.

2.23.2. Configuration Window

Configuration window can be closed and opened at any time. After selecting all the required windows for operation, the Configuration window can be closed. Closing the window does not end the configuration. You can open it at any time. The initial state of the window is Configuration - Attached. In this state, it "overlaps" all windows that are in the Ordinary state (most windows are shown in this state). To use the area occupied by the Configuration window, you can change its state to Ordinary (in this case, other windows will appear on top of the Configuration window) or Hidden (if the window is not needed, it hides automatically, and when you approach it the mouse pointer expands).

2.23.3. Using Window Display Modes

In the Ordinary window mode, it can be visible only within the free work area of the Designer. Using other modes allows to extend the work area or use it more efficiently. You can also “move” the window out of the work area.
Each window (except the calculator) in Designer can be put into Hiding mode. This mode allows to select the desired window for viewing and editing without unnecessary actions by moving the mouse pointer to the line with the title of the window. When the view is finished, simply move the mouse pointer to any other window or the title of another hiding window to minimize the window. In this mode, it is advisable to use the windows that are of a short-term nature (the Syntax Assistant, the Configuration window, the message and search results window, as well as the applied objects window, spreadsheet and text documents opened mainly for viewing).
The Free window state mode allows to place this window anywhere on the screen, regardless of the size and position of the program window (of Designer). In the Attached mode the window can be attached to another window in this state, or to one of the sides of the window, and be located on top of another attached window (combined windows).

Combine such windows that do not need to be viewed at the same time. For example, the Property Palette window and the Syntax Assistant window or the Scoreboard and Call Stack during debugging.

2.24. Saving to a configuration ID file

The red-tape instruction used for the licensing validation mechanism.

2.25. Operating with configuration using the 1C:Enterprise language

Using the built-in language, you can analyze the configuration metadata that is currently being executed. The global `Metadata` context property is used for this purpose. This property provides access to an object of type `ConfigurationMetadataObject`. The properties of this object allow to get information about the configuration properties, the number and composition of configuration objects.
In addition to the ability to perform basic configuration analysis, the application provides the ability to perform similar analysis for any other configuration or extension of the configuration stored in the file. In order to perform this analysis, create a `ConfigurationMetadataObject` object based on a configuration file (or a configuration extension).

```
Configuration = New ConfigurationMetadataObject(New
BinaryData(PathToConfiguration));
```

As a result, the `Configuration` variable will have an object that is similar to the global `Metadata` context property. The value of the `Configuration` variable will describe the configuration located in the file system by the path specified in the `Configuration` variable.
It is also possible to perform the following action:

```
Extension = New
ConfigurationMetadataObject(ConfigurationExtensions.Retrieve()[0].GetD
ata());
```

This **example** creates an object with the extension metadata that was first added to the current configuration. Obviously, if there are no extensions in the configuration, this **sample** code will result in a runtime error.

If you need to get a minimum set of information from the configuration (or extension): configuration name, configuration version, and provider name, you can use the `ConfigurationDescription` objects. For this do the following:

```
FromConfiguration = New ConfigurationDescription(New
BinaryData(PathToConfiguration));
```

The result of this example is the `ConfigurationDescription` object created from the configuration, the path to which is specified in the `PathToConfiguration` variable.

In addition to getting information about the configuration structure and its basic characteristics, the application allows to get some basic set of information from the configuration update file. This set includes a description of the configuration that will result from the update, as well as an array of configuration versions that can be updated using the specified `.cfu` file.

```
FromUpdate = New ConfigurationUpdateDescription(New
BinaryData(PathToUpdate));
```

As a result of this example, `ConfigurationUpdateDescription` type of object will be placed in the `FromUpdate` variable.

Chapter 3. Application interface

3.1. General information

The application interface is focused on user-friendly operation and is arranged in such a way that, performing the necessary actions in the application, the user operates with different forms. Each form can be opened in different ways, which will be described further.

In order to get access to one or another form, you should find the necessary command in the application interface (open the list, generate a report, enter the document, etc.) and execute it. The process of finding the right command is referred to as the application **Navigation**.

The application structure is presented to the user as a hierarchy, which is generated by subsystems and their metadata objects.

Forms that the user operates with can be displayed in different ways:

- on the Home page,
- in a separate window,
- in a separate tab,
- in single window mode - the Taxi interface.

The mode of opening forms (in separate windows or tabs) is defined by a separate configuration of the client application and becomes effective after the client application restarts.

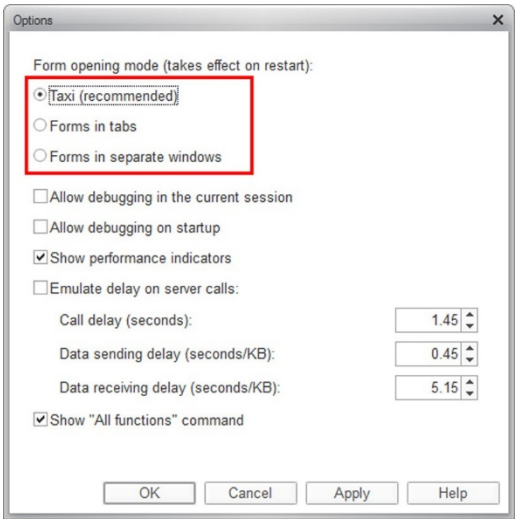


Fig. 31. Setting the Appearance of the Application

This setting is performed in the client application Settings dialog box, which is available using Main menu- Service- Parameters or Service and Settings- Parameters. After selecting the desired appearance parameter, the application prompts you to restart it.

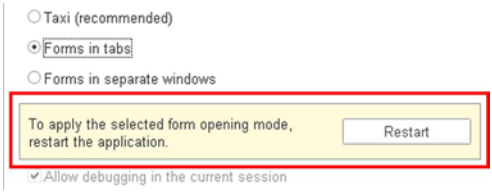


Fig. 32. Suggest restarting when changing appearance

Clicking Restart will restart the application and open it in the selected appearance.

There are two types of windows in 1C:Enterprise:

- Main Window,
- Auxiliary Window.

Each type of windows in 1C:Enterprise is designed for certain tasks. The main application window is used for navigation and command call. Auxiliary windows are used to operate with specific infobase objects (such as documents or catalogs

items), generate reports, or run data processors. The auxiliary window role can be performed by a tab displayed in the main window work area.
The main menu of the main and auxiliary windows of 1C:Enterprise contains only general commands that have no application specifics and cannot be changed by the configuration developer.

3.2. Opening forms in separate windows

3.2.1. General information

In this mode, forms are opened in auxiliary windows. Each 1C:Enterprise window appears in the taskbar of the operating system and in the shift switcher by pressing Alt + Tab.

IMPORTANT! This mode is not supported when using the Web client.

3.2.2. Application Main Window

3.2.2.1. Application Main Window structure

Normally, a 1C:Enterprise application has a single main application window. The main window layout is optimized for efficient navigation through the application, that is, quick finding of the required sections and commands.

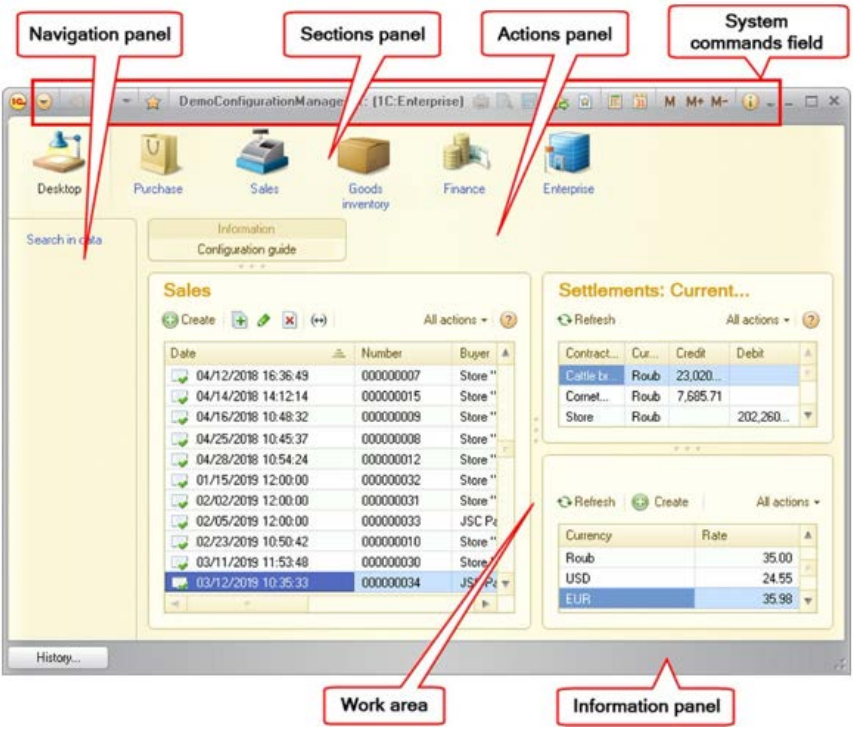


Fig. 33. Auxiliary Window structure

Each state of the main window can be considered as a kind of workplace. For example, if you go to the Inventory section, you will see a set of commands to perform the most common activities related to inventory management.

3.2.2.2. Sections panel

The sections panel displays top-level subsystems list and provides the option to switch between them. Each section represents an application subsystem (for example, Sales, Purchases, or Inventory). For better visualization you can set a meaningful picture for each subsystem. If the picture is not set the subsystem is still displayed in the sections panel. The first section is always Desktop. The

desktop contains the most necessary and frequently used tools of the application. The other sections represent top-level subsystems.
The sections panel provides horizontal scrolling, but you can use access rights to limit the partition set so that scrolling is not required.

NOTE. If there are no sections in the sections panel (they are unavailable or hidden by the user), the sections panel is automatically hidden and the application main window automatically switches to the configuration desktop.

3.2.2.3. Navigation panel

The Navigation panel displays the configuration structure according to the section selected in the Section panel. If the subsystem has subordinate subsystems, they will be displayed as collapsible groups. Further, the subsystems subordinate to the top-level subsystems will be called subsections.
Clicking hyperlinks in this panel usually opens the list forms. In this case, the forms are opened directly in the main window, replacing each other.
Clicking the hyperlink opens the form in the work areaof the main application window.
The Navigation panel can contain commands of three standard groups: Important, Ordinary, and See Also. Commands of the Important group are shown in **bold**. If some group is absent, it is skipped. If the configuration designer has defined the native groups of commands from the Navigation Panel category, they will be located directly in front of the See Also group (which is always the last group in the Navigation Panel).
The following rules apply when the See Also group is displayed:

- See Also commands from the displayed section and all its subsections are visually placed in the same list See Also;
- commands are displayed in the order of sections and subsections specified during configuration;
- inside sections and subsections, commands are displayed in the order specified during configuration.

The order of the sections (top-level subsystems), subsections (subsystems), and commands in the See Also the navigation panel can be changed in the command interface fragment editor.

3.2.2.4. Actions Panel

Actions Panel contains commands that correspond to the current section selected in the Section panel. The Actions panel can contain commands of the following standard groups: Create, Reports, Service and groups created by the developer.
The Create group contains actions that create infobase objects, such as documents or catalog items. This group contains the Create commands for all objects that are part of the subsystem, but by default the visibility of these commands is disabled. The developer should enable visibility for the most commonly used

commands for creating new objects. Reports group contains commands for opening reports, and the Service group - command for opening data processors. If a group contains no commands, the group is not shown. If the configuration designer has defined native command groups that belong to the Actions Panel category, they are located after the Service group.

The following rules apply when the Actions Panel groups is displayed:

- similar groups commands from the displayed section and all its subsections are visually placed in the same list;
- commands are displayed in the order of sections and subsections specified during configuration;
- inside sections and subsections, commands are displayed in the order specified during configuration.

The order of the sections (top-level subsystems), subsections (subordinate systems), and commands in the Actions Panel groups can be changed in the command interface editor.

The default height of the Actions Panel depends on the number of commands in the groups and does not exceed three rows when the panel height is automatically determined. However, the user can arbitrarily change the height by using the separator underneath the panel. If at the current height of the Actions Panel it does not fit all the commands of any group, then in the lower right corner of the group a icon will be placed, clicking on which will open menu that contains all the commands of the group.

3.2.2.5. Information Panel

The information Panel is designed to refer to the latest data, edited by the user, as well as to display the list of recent alerts (information about the actions performed in the application).

3.2.2.6. System command bar

The system command bar allows to perform a number of actions which are independent of the application-specific configuration, but can help in system management. In addition, this area contains various commands to assist in the operation with window.

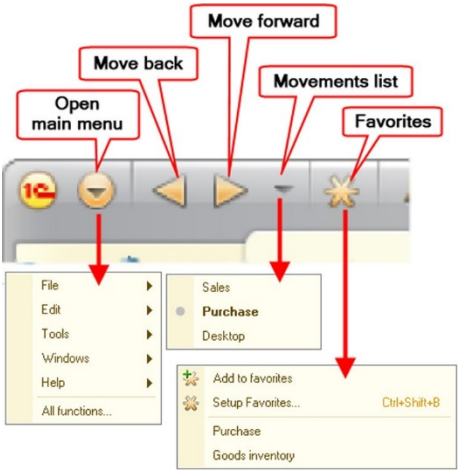


Fig. 34. The left part of the system command bar

On the left side of the system command bar is a button to call the main menu, as well as commands to navigate through the navigation points and user favorites menu. **Navigation point** - a fragment of the interface of the main application window, which can be navigated to using commands located in the main application window.

In 1C:Enterprise, favorites - is a user-created list of links used for quick access to specific configuration sections, navigation points, infobase object forms, reports forms, or data processors.

You can programmatically manage the Favorites list. The `UserWorkFavorites` object is used for this. This object can be retrieved from the application settings repository.

NOTE. When working with favorites, remember that the Favorites list is single-level.

You cannot add a reference to a standard function in the Favorites list (called by the command Main Menu- All Functions - Standard).

In auxiliary windows, the system command bar looks different.

The right side of the system command bar contains various helper commands that help you operate with the window. The list of commands can be configured using the appropriate Setup menu.

3.2.3. Auxiliary windows

Auxiliary window is a window that opens to perform an action, not to navigate the entire application. Such windows, for example, open forms of documents or elements of catalogs, forms of reports and processings.

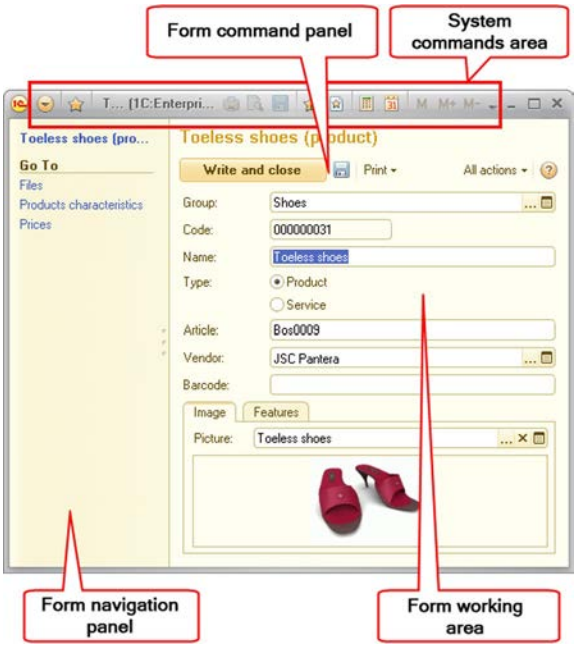


Fig. 35. The auxiliary window structure

Closing the auxiliary window does not close the entire application.
There are a number of opened auxiliary windows is not limited, but for each infobase object (for example, a document or a directory item) only one auxiliary window can be opened. Thus, you can't open several windows to view the same document, unless this is specifically designed in the configuration.
The auxiliary window can contain the **main form** - which is displayed by default when you open the auxiliary window and which opens when you click the first item of the form navigation panel. Clicking the other links in the form navigation panel open the **auxiliary forms**.

3.2.3.1. Navigation panel of the form

The auxiliary window navigation panel (form navigation panel) allows to view various details logically related to the data that are displayed in the main form of the window.

The navigation panel can contain commands of three groups: Important, Go To, and See Also. Commands related to groups Important and Go To, will be combined under the main heading Go To, at this the commands of the group Important are first, bold and separated from the rest of the commands located under the heading Go To with a small indent. If some group is absent, it is skipped. If the configuration designer has defined the native groups of commands from the Form Navigation Panel category, they will be located directly in front of the See Also group (which is always the last group in the Form Navigation Panel).
There is a possibility to return to the main form. To do this, click the hyperlink at the top of the navigation panel (highlighted in **bold**).

3.2.3.2. Form command bar

A form command bar includes the commands that are directly related to the object displayed in the main form. There are standard groups Important and Generate; in addition, the developer can place on the panel the custom groups. Each command from the Important group is displayed with its own button on the panel. The Generate group is presented in the form of the panel submenu. Also all the groups created by the developer are displayed in the form of a submenu. The form commands, such as Save and close, are placed before the Important group commands.
If a group contains no commands, the group is not shown. If the configuration developer adds custom groups to this area, they are placed after the Generate group.

3.2.3.3. System command bar

The left side of the system command bar of the auxiliary window contains only the command to open the main menu and the Favorites command. The right side contains various auxiliary commands that help you to operate with the window (for example, commands to get and navigate by link, call Calculator, calendar, etc.). The right side can be configured using the appropriate Setup menu.

3.3. Opening forms in tabs

3.3.1. General information

In this mode, the form opens as a tab for the main window work area. Some forms still open in separate windows (for example, selection windows), but these windows do not appear in the taskbar.

IMPORTANT! This mode is not supported when using the Web client.

The form opened in the tab can be opened in the auxiliary window, if required by the logic of operation. In this case, the auxiliary window will be displayed in

the taskbar of the operating system and will be available when the windows are switched by Alt + Tab.

3.3.2. Application Main Window

3.3.2.1. Application Main Window structure

Normally, a 1C:Enterprise application has a single main application window. The main window layout is optimized for efficient navigation through the application, that is, quick finding of the required sections and commands.

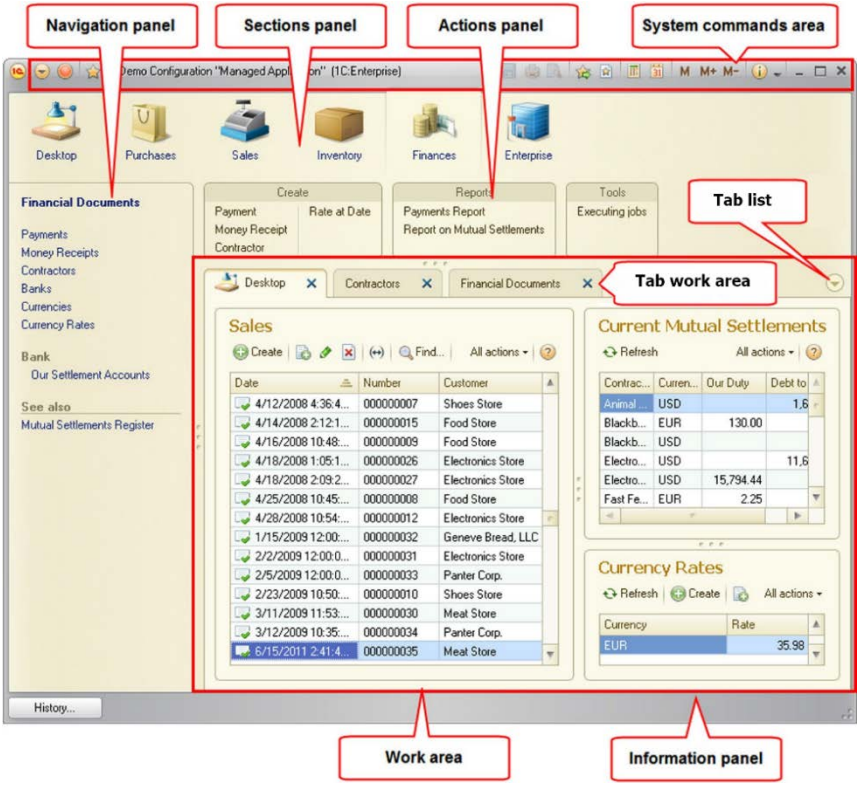


Fig. 36. Main window with tabs

Each state of the main window can be considered as a kind of workplace. For example, if you go to the Finances section, you will see a set of commands to perform the most common activities related to financial transactions.

Each tab displayed in the work area of the main window can be closed (including the Desktop tab). You can do this either by using the context menu (Close command) or by clicking the blue picture on the right side of each tab.

NOTE. The Desktop tab is always the first tab.

The tab title changes size based on the number of tabs in the main window. As the number of tabs increases, the size of the tab header decreases and the title may not be readable. In this case, you can use the special button located to the right of the tabs to get a list of open tabs.

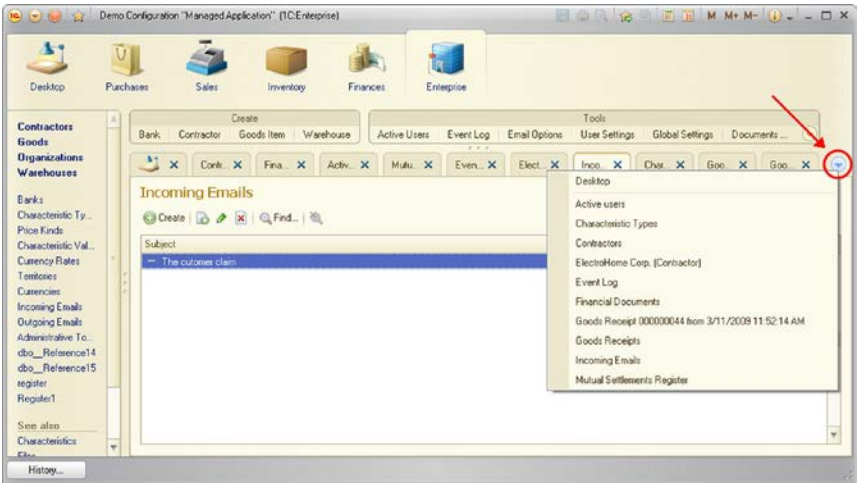


Fig. 37. Bookmark list

If a locking or module form (for example, selection form) is opened in a form opened in the tab, then this form is not displayed as a separate window and is not present in the list of windows on the taskbar. This form blocks the contents only of the tab, in which the locking form was opened. It is still possible to switch between tabs. If you switch to a tab with an open locking form, it will be displayed on the screen.

3.3.2.2. Sections panel

The sections panel displays top-level subsystems list and provides the option to switch between them. Each section represents an application subsystem (for example, Sales, Purchases, or Inventory). For better visualization you can set a meaningful picture for each subsystem. If the picture is not set the subsystem is still displayed in the sections panel. The first section is always the Desktop. The desktop contains the most necessary and frequently used tools of the application. The other sections represent top-level subsystems.

The sections panel provides horizontal scrolling, but you can use access rights to limit the partition set so that scrolling is not required.

Navigating through sections does not change the information displayed in the work area, but changes the contents of the navigation panel and the actions panel.

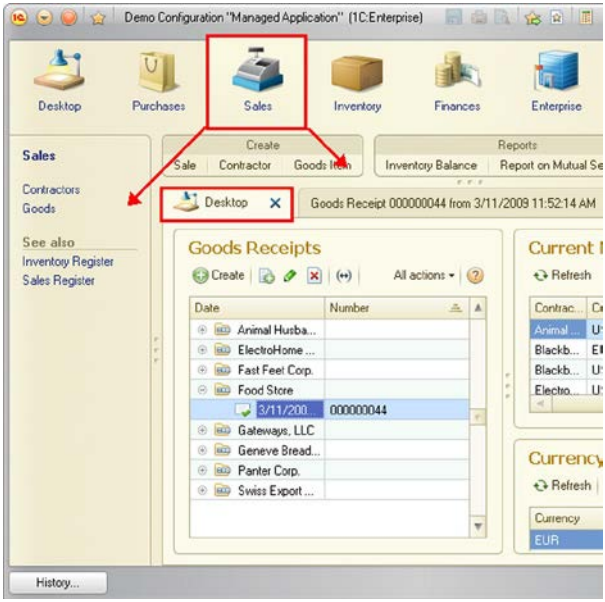


Fig. 38. Switching Subsystems and Work Area

NOTE. If there are no sections in the sections panel (they are unavailable or hidden by the user), the sections panel is automatically hidden and the application main window automatically switches to the configuration desktop.

3.3.2.3. Navigation panel

The Navigation panel displays the configuration structure according to the section selected in the Section panel. If the subsystem has subordinate subsystems, they will be displayed as collapsible groups. Further, the subsystems subordinate to the top-level subsystems will be called subsections.

Clicking hyperlinks in this panel usually opens the list forms. In this case, the forms are opened directly in the main window, replacing each other.

Clicking the hyperlink opens the form in the work areaof the main application window.

The Navigation panel can contain commands of three standard groups: Important, Ordinary, and See Also. Commands of the Important group are shown in **bold**. If some group is absent, it is skipped. If the configuration designer has defined the native groups of commands from the Navigation Panel category, they will be located directly in front of the See Also group (which is always the last group in the Navigation Panel).

The following rules apply when the See Also group is displayed:

- See Also commands from the displayed section and all its subsections are visually placed in the same list See Also;
- commands are displayed in the order of sections and subsections specified during configuration;
- inside sections and subsections, commands are displayed in the order specified during configuration.

The order of the sections (top-level subsystems), subsections (subsystems), and commands in the See Also the navigation panel can be changed in the command interface fragment editor.

3.3.2.4. Actions Panel

Actions Panel contains commands that correspond to the current section selected in the Section panel. The Actions panel can contain commands of the following standard groups: Create, Reports, Service and groups created by the developer. The Create group contains actions that create infobase objects, such as documents or catalog items. This group contains the Create commands for all objects that are part of the subsystem, but by default the visibility of these commands is disabled. The developer should enable visibility for the most commonly used commands for creating new objects. Reports group contains commands for opening reports, and the Service group - command for opening data processors. If a group contains no commands, the group is not shown. If the configuration designer has defined native command groups that belong to the Actions Panel category, they are located after the Service group.

The following rules apply when the Actions Panel groups is displayed:

- similar groups commands from the displayed section and all its subsections are visually placed in the same list;
- commands are displayed in the order of sections and subsections specified during configuration;
- inside sections and subsections, commands are displayed in the order specified during configuration.

The order of the sections (top-level subsystems), subsections (subordinate systems), and commands in the Actions Panel groups can be changed in the command interface editor.

The default height of the Actions Panel depends on the number of commands in the groups and does not exceed three rows when the panel height is automatically determined. However, the user can arbitrarily change the height by using the separator underneath the panel. If at the current height of the Actions Panel it does not fit all the commands of any group, then in the lower right corner of the group a icon will be placed, clicking on which will open menu that contains all the commands of the group.

3.3.2.5. Information Panel

The information Panel is designed to refer to the latest data, edited by the user, as well as to display the list of recent alerts (information about the actions performed in the application).

3.3.2.6. System command bar

The system command bar allows to perform a number of actions which are independent of the application-specific configuration, but can help in system management. In addition, this area contains various commands to assist in the operation with window.

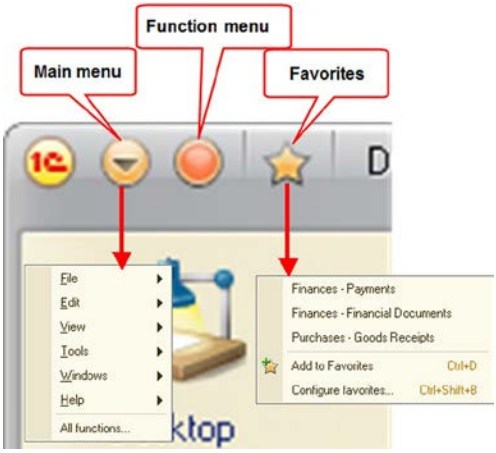


Fig. 39. The left part of the system command bar

On the left side of the system command bar is a button to call the main menu, as well as commands to call functions menu and user favorites menu.
The main menu contains the View menu, which allows to customize the view of the application window.
In particular, this menu allows to turn off the visibility of the sections panel, navigation panel, and actions panel. Hiding panels leaves more space for forms and reduces the need for scrolling. Also pay attention to the command Hide all command panels. After executing the command, the sections panel, the navigation panel, and the actions panel are hidden.

Repeating the command returns the hidden panels. Consider the following feature: the Hide all command panels command does nothing if the sections panel, the navigation panel, and the actions panel are hidden using other commands of the View menu.

To speed up and simplify access to commands of a particular subsystem a special function is used - Functions Menu. To call this mode, press the special button in the System commands bar.

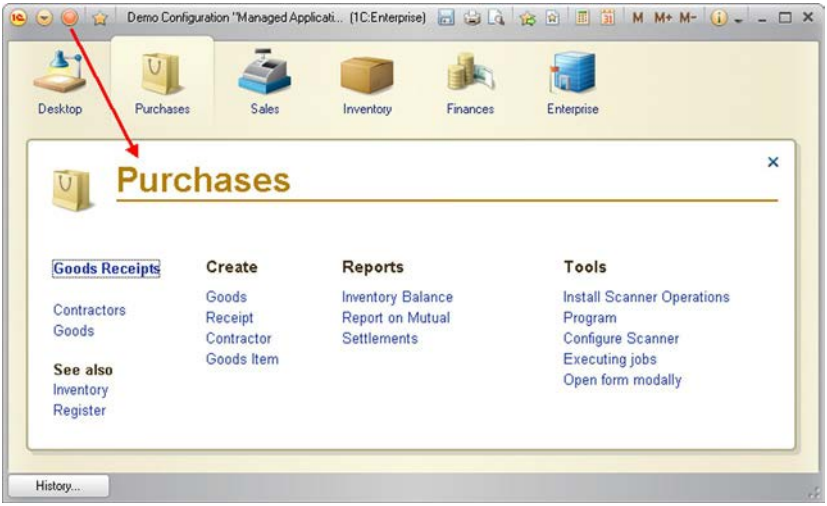


Fig. 40. Functions menu

The functions menu is a sections panel and a special form that contains all the commands of the selected section. The list of sections and command composition correspond to the current user settings. The first column of the form corresponds to the navigation panel, the remaining columns - to the actions panel groups of the selected subsystem. The display of the sections panel does not depend on the state of the Main menu - View - Sections panel. After selecting the required command, the main window returns to the view that was before the function menu was called. The composition of the commands is updated if you select a different partition in the function window.

In 1C:Enterprise, **Favorites** - is a user-created list of links used for quick access to specific configuration sections, navigation points, infobase object forms, reports forms, or data processors.

You can programmatically manage the Favorites list. The `UserWorkFavorites` object is used for this. This object can be retrieved from the application settings repository.

NOTE. When working with favorites, remember that the Favorites list is single-level.

You cannot add a reference to a standard function in the Favorites list (called by the command Main Menu- All Functions - Standard).

In auxiliary windows, the system command bar looks different.

The Window menu displays the main window, window tabs, and auxiliary windows. The check box indicates:

- active auxiliary window;
- active tab of the main window;
- the main window itself, in no tabs are yet opened.

The right side of the system command bar contains various helper commands that help you operate with the window. The list of commands can be configured using the appropriate Setup menu. The availability and composition of the commands depend on the current form in the active tab.

3.3.3. Auxiliary windows

3.4. Taxi interface

3.4.1. General information

In this mode, the forms are usually opened in the main window work area. Some forms still open in separate windows (for example, selection windows), but these windows do not appear in the taskbar. You cannot switch between windows using Alt + Tab. You can switch between open forms using the special buttons in the header of all forms that are open in the work area of the main application window.

3.4.2. Application Main Window

3.4.2.1. Application Main Window structure

Normally, a 1C:Enterprise application has a single main application window. The main window layout is optimized for efficient navigation through the application, that is, quick finding of the required sections and commands. The main application window does not open forms (except the work area).

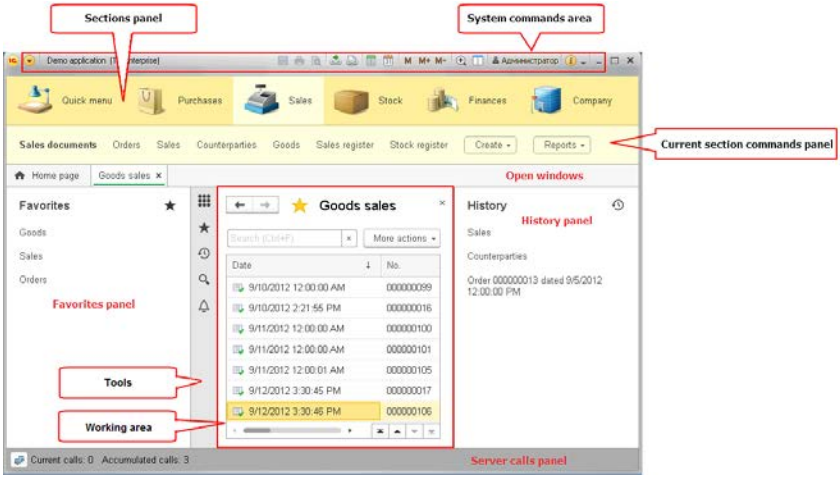


Fig. 41. Main Window in the Taxi interface

The following elements of the main window are numbered on fig. 41:

1. Current section functions menu (and current section functions panel).
2. Global search field.
3. Access to the notification center.
4. Current user.
5. Service and settings.
6. Sections panel.
7. Open items panel
8. History.
9. Main window work area.
10. User work favorites.
11. Server calls display panel.

The application main window can be displayed in two variants:

1. Without the operating system window title (fig. 41). This variant is used if the client application operates under Windows OS management, not in the terminal mode, and not in the remote desktop mode. The main panel is the window title in this case.

2. With the operating system title. This variant is used when operating in Linux and macOS operating systems, with the use of web-client, and under the management of Windows operating system in the terminal mode or in the remote desktop mode. In this case, the main panel is in the OS title.

The application main window representation variants are switched automatically depending on the OS and environment in which the client application is started. In addition, in the case of the client application operation in Windows OS, the OS title representation can be managed with the help of the methods `GetOSTitleRepresentation()`/`SetOSTitleRepresentation()` of the object in the `ClientApplication`. The OS window title representation can be set only in the event handler `BeforeSystemStartup`.

Main panel contains the main means of navigation in the application. Main panel contains the following elements: menu functions opening button, application title, global search field, open buttons, notification center, history, favorites, current user name, and the button to open Service and Settings.

Use Service and Settings - Show M, M+, M- to manage buttons with similar names on the main panel. These buttons are displayed to the left of the global search field.

When operating in the Taxi interface, several different panels can be arbitrarily located in the main window of the application. The work area of the main window of the application displays all forms with which the user is working. The exceptions are locking forms that open in separate windows. If a locking form (for example, selection form) is opened in a form opened in the work area, then this form is displayed as a separate window and is not present in the list of windows on the taskbar. Depending on the opening mode of the window, operations with either the owner form or the entire system interface are locked. If the auxiliary window is open in the owner window lock mode, it is still possible to open other forms and use the command interface of the application. If you switch to the form from which the form locking interaction with the owner is open, the locking form is displayed.

Let us consider the components of the main application window in more detail.

3.4.2.2. Home page

To start use the application open its home page.

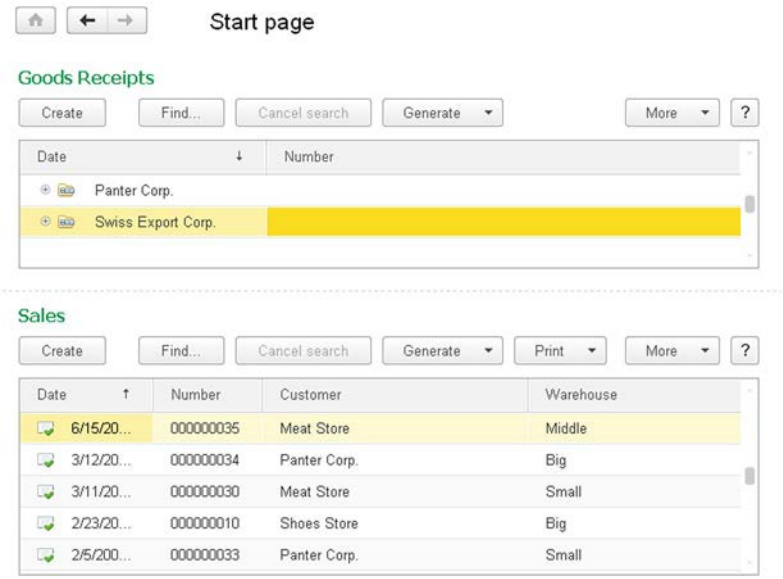


Fig. 42. Home page

This page is used by the application developer to display forms which are most commonly used in this application (as envisioned by the developer). The availability of forms on this page is determined by the access rights and functional options. The home page can't be closed.

The start page interface can be configured using the configuration properties. The user can also form the personal composition of the home page from the list of available forms that the application developer allows to place on the home page. To do this, use the special settings dialog of the home page: Service and Settings- Settings- Home Page Settings.

3.4.2.3. Sections panel

The section panel shows a list of top-level subsystems and allows you to quickly select desired set of functions. Each section represents an application subsystem (for example, Sales, Purchases, or Inventory). For better visualization you can set a meaningful picture for each subsystem. If the picture is not set the subsystem is still displayed in the sections panel. The first section is always Quick

menu. This section contains the most necessary and frequently used tools of the application. The other sections represent top-level subsystems.

Fig. 43. Sections panel

Actually the sections panel element groups system commands logically associated within some activity within the application. For example, it may be activity related to purchase management (Purchases section), inventory management (Inventory section), etc. The sections panel is not a tool to switch workplace.
The sections panel has scroll function. The composition of the sections panel for a specific user is determined by his or her access rights and the installed functional options. The visibility of the available sections can be customized by the user using Service and Settings - Settings- Sections Panel Setup.
When you click a section, the result depends on whether the current section functions panel is displayed:

- The panel is not displayed. Clicking a section displays the functions menu of the section. In this case, the current section is not displayed (when the functions menu is closed).

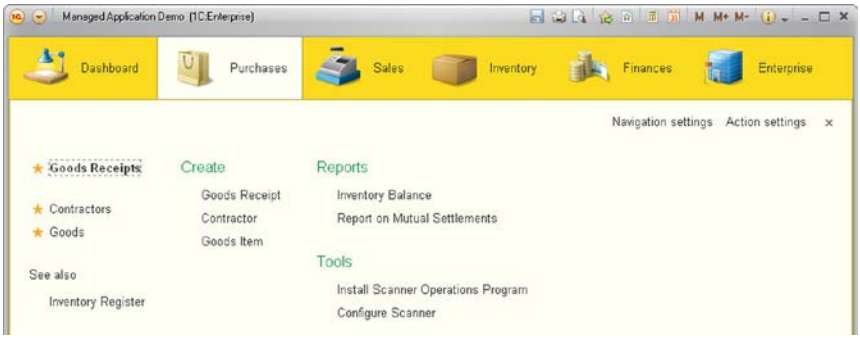


Fig. 44. The functions menu when a section is selected

- The panel is displayed. In this case, when you select a section, the functions menu is not opened, but the contents of the command panel of the current section is changed. If the displayed information is not enough, you can open the functions menu by re-selecting the section. In addition in this case the current section is always displayed on the sections panel.

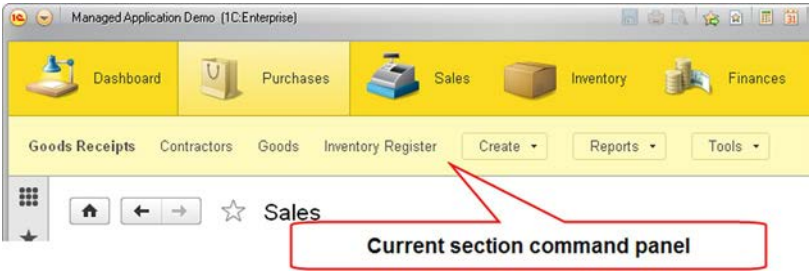


Fig. 45. The current section functions panel

The functions menu contains navigation panel and action panel commands. From the functions menu you can add any command to your favorites. To do this, simply click the star to the left of the command name. To remove a command from the favorites, click the star icon again.

In the upper right corner of the function menu there is the input field to search and the settings menu. The menu contains commands to set up the navigation panel (the Navigation settings command) and the actions panel (the Actions settings command).

To start the search, press Ctrl + F or start typing. After entering text on the left side of the functions menu panel, the text Functions search appears and animation is displayed (during the search). The search is performed for commands of all sections, not just in the current section commands. The search resutl is displayed as follows:

The search result includes the commands with presentation containing any of the words in the search string in any position. Found occurencies are highlighted. Commands of the Important category are not marked in bold. The found commands are grouped into sections (displaying the section image) and are placed in the same functions menu columns as when the search was disabled. The search is performed not only by the commands presentations, but also by the commands groups presentations (except the group See also). If one or several words are found in the name of the group (section), then the search in the command presentations is performed without taking the word(s) into account.

To exit the search mode, you can clear the search field, click the clear search field button or press Esc.

There are several ways to open the functions menu:

- Selecting a section;
- Using the main panel command.
- Using keyboard:
  - Pressing F10;
  - Using keyboard shortcut Ctrl + Shift + T;

- Using keyboard shortcut Alt + 2. In this case, the cursor is placed on the first command in the navigation panel;
- Using keyboard shortcut Alt + 3. In this case, the cursor is placed on the first command in the actions panel.

NOTE. If there are no sections in the sections panel (they are unavailable or hidden by the user), then the sections panel is automatically hidden.

3.4.2.4. The current section functions panel

Current section functions panel contains commands from the current section.

Fig. 46. The current section functions panel

Current section functions panel:

- Navigation panel commands are located at the beginning of the panel. Commands of the Important group are shown in bold.
  If the panel is displayed horizontally, the maximum possible number of commands (at least one) is displayed. If all the commands do not fit into the panel - the More submenu containing the remaining commands is displayed.
  If the panel is displayed vertically, at least one and no more than 10 commands of the navigation panel are displayed. If there is submenu More in the panel - it is located as the 11th element of the command panel.
- The navigation panel commands are followed by submenus for all non-empty groups of navigation panel commands and actions panel of the selected section.

3.4.2.5. Open items panel

This panel contains the list of forms opened in the current session with the application.

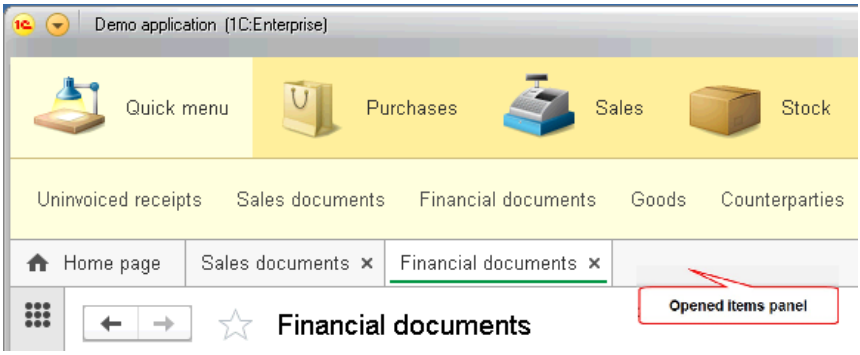


Fig. 47. Open items panel

The first open form is always the home page (if it is configured in the application). Forms of the work area of the home page are not displayed as separate forms in the open items panel. Clicking on the “cross” icon in the upper right corner of the panel element closes the form corresponding to this element.

If a form is locked by a form opened with the Lock Owner Window flag (the form property Window Opening Mode), then the locked form can't be closed using the “cross” icon in the open items panel.

The display of the open items panel depends on the orientation of the panel:

When the panel has horizontal orientation:

- The text in the panel elements is displayed in one line.
- A docked window is displayed as an icon without text.
- As the number of elements increases - the existing elements begin to shrink, reducing the amount of text displayed.
- If some elements in the open panel contain truncated text, then:
  - On the right side of the panel, a button is displayed to open the list of all windows.
  - The element for the home page is displayed as an icon without text.
- Two side-by-side windows are shown side by side as a single element.
- On the elements of the panel, except for the home page and the docked window, a closing “cross” icon is displayed.
- If the title of the document form or the business process contains the presentation of the document (business process) number and the number in the presentation partially or entirely does not fit in the corresponding element of the opened panel, then the number is displayed in the right-hand part of the panel element. The system tries to display all the number.

When the panel has vertical orientation:

- The text in the panel elements is displayed without limit on the number of lines.
- Vertical scrolling is used if necessary, and the home page is not included into the scrollable items.
- The element for the home page is displayed as an icon with text.
- Two side-by-side windows are shown one under another as a single element.
- The closing “cross” icon is displayed on the panel elements when you move the mouse cursor over them (except the home page).

If you configure to display the open items panel in the main application window, then the home page button is hidden in the work area of the home page.

In the open items panel you can change the order of the elements by dragging. In addition you can close the forms by clicking the middle mouse button (wheel) on the open item panel element.

3.4.2.6. History

1C:Enterprise stores the history of your work. You can use the history for quick access to infobase objects that you recently created or updated (such as documents or catalog items). The history contains links to the elements of the application, ordered by time of use. History can be displayed as history form or as history panel.

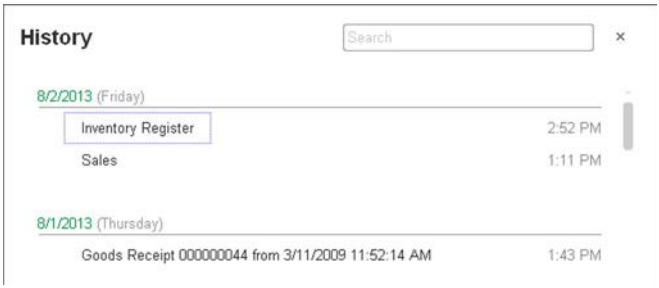


Fig. 48. The history form

List items are grouped by date when a form of application was used. Within a date, the items are ordered by the usage time that is displayed to the right of the item presentation.

The user history is stored in the infobase. If a single infobase object was opened multiple times on different days, each day contains a link to that object. If an

object was opened several times within a single day, only the latest object opening is displayed in that day's history. A maximum of 400 records is stored per user. If the number of records exceeds 400, each new record replaces the oldest one.

If the list of history is large enough, you can use search function for convenience. You can enter a search query in the special field in the upper part of the form. It can be accessed by using the keyboard shortcut Ctrl + F or simply typing the required text.

If the history is displayed in the main window panel, then the list has the following features:

- No grouping by dates and the time of use of the application object is not displayed.
- When the panel is horizontal, it displays the number of elements fitting in the space provided.
- When the panel is vertical, it displays 10 first elements maximum.
- Clicking on the panel header (History) opens the history form.

You can open the history form using the tool panel command or the Ctrl + Shift + H shortcut.

Information is added to the user history in several ways:

- When performing manual operations (open, create, save);
- When displaying a user notification with a link to system object (the object of the link is added to the history). The details for the notification functionality see on page 459.
- When programmatically update the user history using the global context property `UserWorkHistory`.

If an object is saved or opened programmatically, the object information is not added to the history.

3.4.2.7. Favorites

Favorites contains a list of links to the various elements of the application that user has marked for frequent use. Favorites can be displayed as favorites form or as favorites panel.

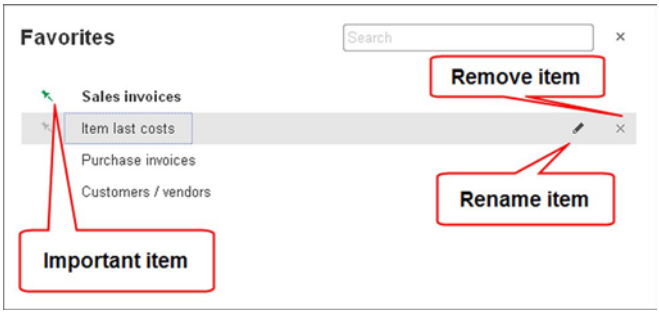


Fig. 49. The favorites form

The following elements are numbered on fig. 49:

1. Attribute of an important element of favorites.
2. Favorites element representation edit button.
3. Favorites element deletion button.

Among the favorites items, you can select the elements that you use most often. To do this, click the button icon to the left of the item. Such an element is marked in bold and the next time you open your favorites, it is located at the top of the panel and do not change its position. You can also rename an item or remove it from the list.

If the list of favorites is large enough, you can use the search function to find desired item. You can enter a search query in the special field in the upper part of the form. It can be accessed by using the keyboard shortcut Ctrl + F or simply typing the required text.

If Favorites are displayed in the main window panel, then the list has the following features:

- You can't search, rename, delete and mark an important element of the favorites.
- When the panel is horizontal, it displays the number of elements fitting in the space provided.
- When the panel is vertical, it displays 10 first elements maximum.
- Clicking on the panel header (Favorites) opens the favorites form.

You can open the favorites form using the tool panel command or the Ctrl + Shift + B shortcut.

Favorites are unavailable if the user has no right `SaveUserData`.

3.4.2.8. Notification center

The notification center provides access to unprocessed user notifications with the status `Important`.

If the user has unprocessed notifications, then a special indicator is displayed in a special way.
Clicking on the notification center icon opens the notifications list.

Clicking the notification is completely the same as clicking on the notification window - the associated action is performed (if it is set) and the notification is removed from the list. You can also delete all notifications at once (command Clear Alerts) but in this case the actions set for the alerts are not performed.
If an interaction system message is deleted from the notification center the relative conversation will be marked as read. The conversation is marked as read irrespective of the way of the message deletion: with the help of the hyperlink Clear notifications or "X" button of a certain message.

3.4.2.9. Global search field

Global search field -is the general interface to the following functions: information search, calculator, tools to follow links (navigational and "usual"). This interface functions can be modified with the help of embedded language. The global search field is inaccessible if a mode different from usual (Usual in configuration properties Client application main page mode) is set for the application main window.
The search is performed in the following sections by default:

- Favorites.
- History.
- Functions menu.
- Messages from non-context conversations (including "face-to-face" conversations) and interaction system non-context conversations titles.
- In the information base data (fulltext search).
- Application solution and platform help.

To access the global search field use the key combination Ctrl + Shift + F.

The search results are shown in a drop-down list. If a link to a website is specified in the global search field -it will be opened in the browser by default.
The search history is saved in user settings. History is saved when the session is terminated and every 20 minutes in case of any changes in it. The search string is placed in history if the user selects one of the results of the search by this line. History contains not more than 7 elements.
If the search returns no results the message of the kind <search string> not found in the drop-down list.
The line contained in the exchange buffer is added to the list as the first element in the drop-down list in the thin client:

- The line is added only if the drop-down list shows the results of the preceding search or if the search string contains an empty line.
- The line is added only if it was placed in the exchange buffer not earlier than 20 seconds ago.
- The line is added only if it is at least 260 characters long and contains not less than 20 words.
- The line is added only if it differs from the current line in the search field.

3.4.2.10. Multiple forms view

The 1C:Enterprise allows to simultaneously display multiple forms:

- It is possible to “join” the display of two forms so that both forms are displayed simultaneously when switching to any of the combined forms.
- It is possible to dock one form on the screen so that the docked form is always displayed in the work area of the main application window.

Combining and docking forms is not intended to organize permanent workplace; these features don't replace the home page customization. After the session end all the combined and docked forms (if any) are “forgotten”.

There several options to control the simultaneous display of two forms:

1. The form header context menu. This menu allows it to dock/undock windows and combine them with other windows.

2. Context menu of the item in the open items panel. There are more features to control the form display than in the previous option.

   The main difference is the commands Show side-by-side with current window..., which are displayed if the context menu is opened for the item of the inactive form. The menu will not contain these items for an active form.

If the menu is opened using the button of the system command bar or by right-clicking the form header, then only commands to combine with the second form are available (the commands Show side-by-side with another window...). If the menu is opened by clicking the right mouse button on an item of the open panel, then the menu includes also the commands to display the form together with the current form (commands Show side-by-side with current window...). Then the commands to dock the form to any edge of the work area follows (Dock...). Next is the command to disable docking or side-by-side display.

If the Show side-by-side with another window... command is used, then the current form and the form selected from the menu are displayed together. If the Show side-by-side with current window... command is used, then the current form and the form for which presentation the context menu command is used are displayed together. To simplify the joining process, it is recommended to use the context menu command of the open item panel.

For combining and side-by-side view the general rule apply: if the form on which the combine or dock command is performed is already combined or docked, then the first a command to separate or undock is performed. -Then the selected command follows.
The example of windows displayed side-by-side is a comparison of the results of report for different periods of time (or with different settings). To display two forms side by side, do the following:

1. Open the analyzed report form.
2. Open the same report in another window (using the commands of the report form or the context menu of the item of the functions menu).
3. Set up and run both reports.
4. Combine the forms using any suitable option.

When combining forms consider the following features of the command:

- Commands of the button Show side-by-side/Dock are unavailable for the home page and for the form locking the owner's window.
- Command Show side-by-side (vertically) places the form selected from the submenu to the right. If the command is called from the context menu of the form other than current form, then this form becomes active.
- Command Show side-by-side (horisontally) places the form selected from the submenu below the current form. If the command is called from the context menu of the form other than current form, then this form becomes active.
- A separator is placed between the combined forms so that both forms are displayed of equal width (height).
- Activation of any of the forms displayed side-by-side displays both forms simultaneously.
- A new form is opened from any of the form displayed side-by-side just as from usual form.
- The Home page, Forward, and Back buttons always appear at the left (top) form of the combined form.
- The home page can't be displayed in any other form (except the docked form).

Another option to display forms simultaneously is to dock a form so that it is always accessible to the user and is not overlapped by the usual forms. Such a display option can be used, for example, in cases where there is some data source that must be constantly displayed (report or document journal), and other forms to work in the application should not overlap this source.

A form can be docked to any side of the main application window.
To dock any form select the command Dock... (using any suitable option). The docked form occupies 40% of the work area (by default) of the main application window. A separator is placed between the docked form and the rest part of the

working area. It allows to change the size of the docked window. The position of the separator (for each side of the main application window) is stored in the settings of the main application window in the system settings storage.

3.4.2.11. Application form scaling

The client application provides the ability to scale the displayed forms on the screen. The user can change the scale of one or all forms of the application in the range from 50 to 400% relative to the scale specified in Designer. To display a dialog box intended to modify scale, use Modify Scale in Service and Settings. The opened dialog box allows to change the scale using the slider, and also to set the specified scale for all forms of the application (the Set for all dialog box button). The scale dialog box can be moved to any place on the screen.

When changing the scale, please consider that the scale set in the client application is added up the scale specified in Designer for each form (the property `ClientApplicationForm.Scale`). Thus, if in Designer the scale is set at 10%, and in the client application the scale is set at 50%, then the resulting scale is 5% of the actual size of the form.

The scale feature is unavailable if the compact display mode is set for the entire application (using the `ClientSettings` object).

NOTE. The scaling is possible for managed forms only.

3.4.2.12. Panel customization

3.4.2.12.1. Manual

The Taxi interface allows to customize the display and layout of various panels in the main application window. To do this, use the Panel editor. For that purpose, select Service and Settings- Settings - Panels Setup....

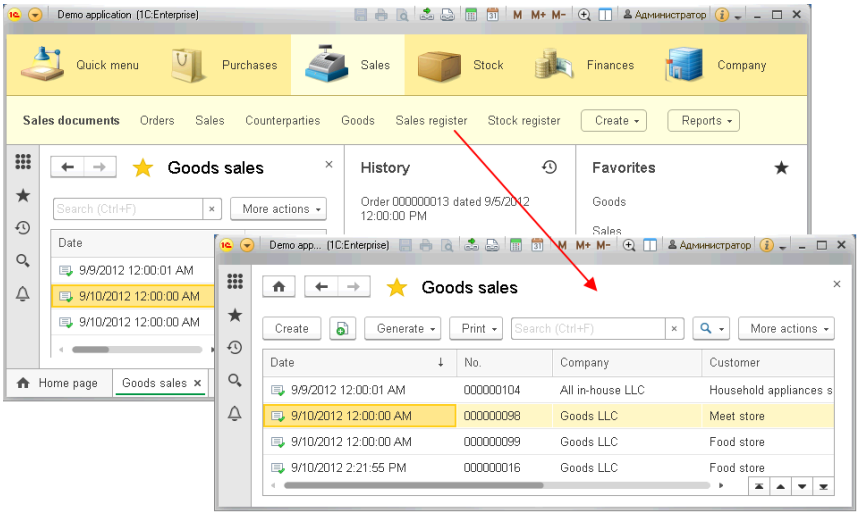


Fig. 50. Panel editor

In opened editor the user can use mouse to move the corresponding element of the main application window (the lower part of the form, shown with a gray background). Items should be dragged inside the light rectangles marked with a gray line. In this case, the panels are displayed in different rows or columns. You can drag an item onto an already existing item. In this case, the items are displayed in the same row or column. The center of the editor displays the works area of the home page. It is impossible to place any panel into the area.

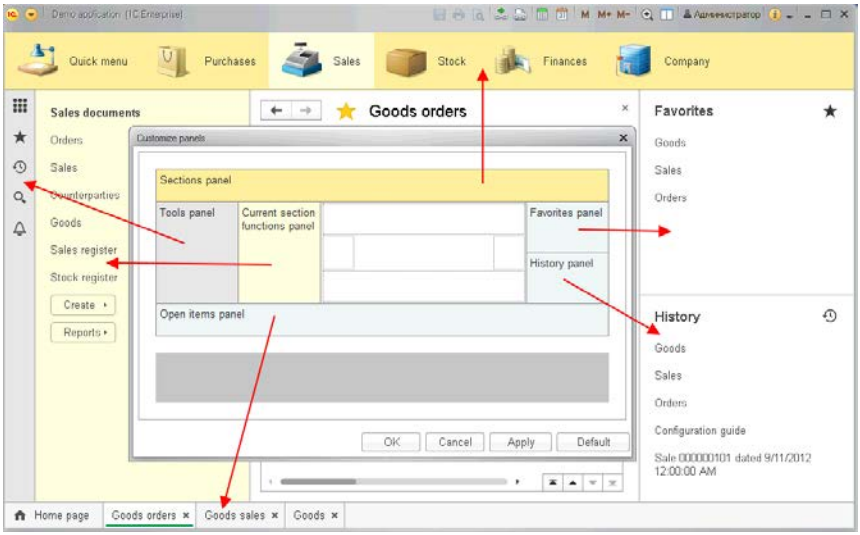


Fig. 51. Panel layout example

To delete an item you need to drag it from the layout back to the item with gray background.

All items can be removed from the main application window except the working area of the main application window. For example, you can create a minimalistic interface of the application, when the main window contains only the work area and the open items panel.

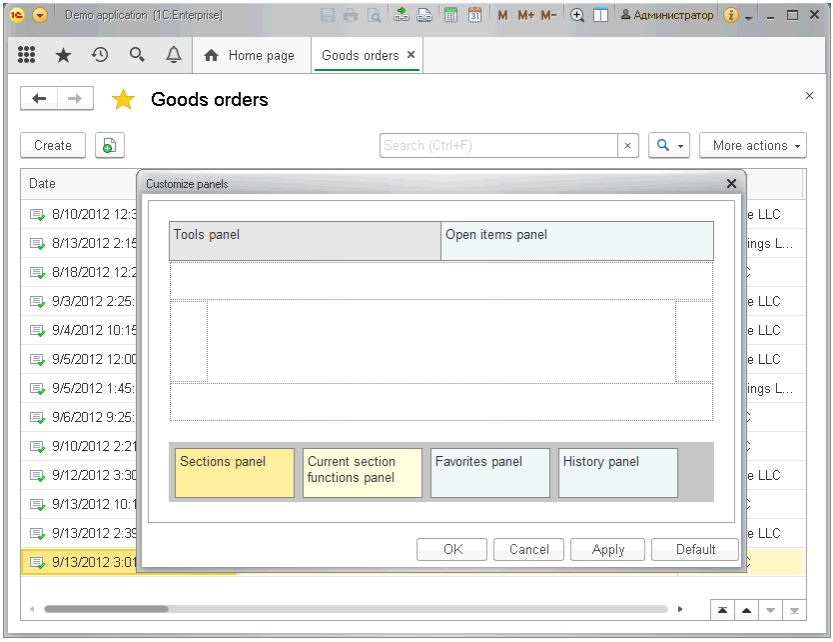


Fig. 52. Minimalistic interface

Clicking the Default button returns the interface to the state specified by the application developer in the configuration property Client Application Interface.

3.4.2.12.2. Software

You can also customize the display and layout of various panels in the main application window using 1C:Enterprise language. To perform the customization use methods `GetContent()` and `SetContent()` of the object `ClientApplicationInterfaceSettings`. Thus, it is possible to get the current settings of the client application interface, modify them as necessary and then set them back.

Client application settings are stored in the standard system settings storage. The settings key is - Common/ClientApplicationInterfaceSettings. To change the settings, first read them from the repository:

```
Settings =
SystemSettingsStorage.Load("Common/ClientApplicationInterfaceSettings"
);
```

Then the settings can be modified and written back into the storage:

```
SystemSettingsStorage.Save("Common/ClientApplicationInterfaceSettings"
, "", Settings);
```

After saving the new configuration variant, rebuild the client application interface by calling the `RefreshInterface()` method.

Let us consider an example where it is necessary at each system start to force the installation of the simplified version of the client application interface, which consists of only one panel (open items panel) located along the top side of the main application window. To show an example, find a common module with Server and Server Call check boxes selected. Let the name of the common module is `SetupInterface`. The common module should include the following procedure:

```
Procedure MinimalInterface() Export

  Settings = New ClientApplicationInterfaceSettings;
  ContentSettings = New ClientApplicationInterfaceContentSettings;
  ContentSettings.Up.Add(ClientApplicationInterfaceContentSettingsItem
("OpenItemsPanel"));
  Settings.SetContent(ContentSettings);
  SystemSettingsStorage.Save("Common/ClientApplicationInterfaceSetting
s", "", Settings);

EndProcedure
```

The managed application module should include the event handler `OnStart`:

```
Procedure OnStart()
  SetupInterface.MinimalInterface();
  RefreshInterface();
EndProcedure
```

As a result, regardless of settings made by the user in the previous session, each new session will start with a minimal interface.

3.4.2.13. Closing the application

You can close the application using the next options:

1. Select Exit in Service and Settings - File.
2. With the help of a hyperlink with the user name in the main panel of the application main window. This hyperlink is not displayed if no list of users is kept in the information base.

Clicking the hyperlink with the user name in the main panel displays a special menu that contains the value of Full name property for the current user (it is also displayed in the main panel), Name property for the current user, and Log out

hyperlink. At the application exit (by way of selecting Service and Settings or Log out) the following actions are performed:

- OpenID authentication is cancelled (OpenID logout) - if the authentication was performed using OpenID.
- client application exit.

On the web client exit you can specify which page (specifying the URL) is opened in the web browser after the log out. To do this, specify the URL to open after exiting the web client parameter when publishing the web client. The exit behavior of the web client depends on whether this parameter is specified:

- The URL is specified - system navigates to the specified address. If no scheme is specified, it is assumed that the HTTP scheme is used.
- The URL not specified - system navigates to a special page that allows to quickly log in the application again.

The navigation to the address (URL) specified during publication is performed in the following cases:

- exit using the commands Close, Exit, Log out;
- executing the global context method `Exit()` with the `RestartOnCompletion` option set to `False`.

In other cases, the system goes to address about:blank or the web client restarts.

3.4.3. Auxiliary windows

3.4.3.1. General information

Auxiliary window -is the application window in which the forms created in the application are opened. The auxiliary window can be opened in the work area of the main application window or in a separate window. The opening mode of the auxiliary application window depends on the value of the form property Window Opening Mode.

Closing the auxiliary window does not close the entire application. There are a number of opened auxiliary windows is not limited, but for each infobase object (for example, a document or a directory item) only one auxiliary window can be opened. Thus, you can't open several windows to view the same document, unless this is specifically designed in the application.

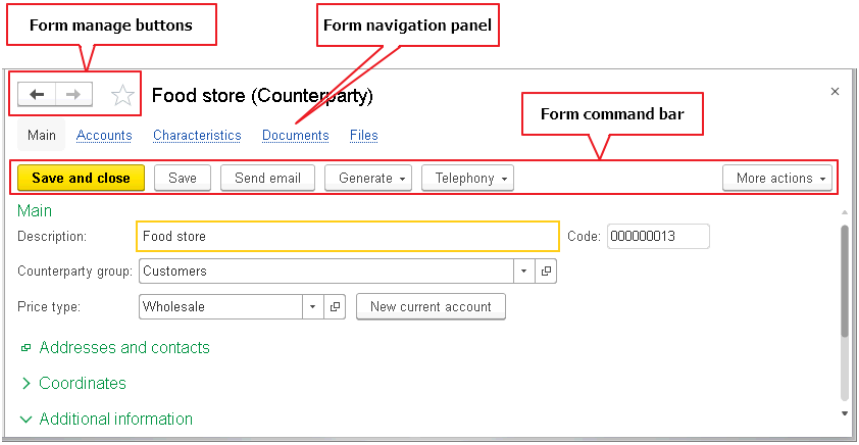


Fig. 53. The auxiliary window structure

The following elements are numbered on fig. 53:

1. Form management buttons. Go to home page button (the first button on the left) is displayed in the form if the open items panel display is disabled in the panel settings. The buttons for navigation to the previous/next form allow to navigate through the list of open windows. The button to navigate to the next form is activated only after the button to navigate to the previous window is pressed at least once.
2. The button to add the form to the favorites list. If the form is not in the favorites list -then the star color corresponds to the form background. Otherwise the star is colored yellow. The button is unavailable if the user has no right `SaveUserData`.
3. Form navigation panel.
4. Form command panel.
5. Get navigation link button of this form.
6. Form close button.
7. Form menu. The home page always displays one form menu (in the top right corner of the home page). The form menu is unavailable in dialog boxes which are displayed using `Question()`, `ShowQuestion()`, `Warning()` or `ShowWarning()` methods.
8. Form work area.

The auxiliary window can contain the **main form**, -which is displayed by default when you open the auxiliary window and which opens when you click the Main

item of the form navigation panel. Clicking the other links in the form navigation panel open the **auxiliary forms**.

3.4.3.2. Navigation panel of the form

The auxiliary window navigation panel (form navigation panel) allows to view various details logically related to the data that are displayed in the main form of the window.

The navigation panel can contain commands of three groups: Important, Go To, and See Also. The general layout of the commands is as follows:

- The first command is always the Main command (highlighted in **bold**). It opens the main form of the auxiliary application window.
- Then the commands of the Important group follow.
- Then the commands of the Go To group follow.
- The latter are the commands of the See Also group. If the configuration developer defines custom groups of commands from the category Form Navigation Panel, then commands of these groups are placed directly before the commands of the See Also group (commands of this group are always at the very end of the form navigation panel).

If the commands do not fit in the navigation bar, then the last command of the panel is More submenu, which contains all the commands that don't fit the displayed part of the navigation panel.

The form navigation panel is accessible via the Alt + 4 keyboard shortcut. In the form navigation bar the following keyboard shortcuts are available:

- Move to the next/previous panel item - Tab/Shift + Tab or Right Arrow/Left Arrow. When the first/last item of the panel is reached, the move to the last/first item of the panel is not performed;
- Move to the first item of the panel - Home;
- Move to the last item of the panel - End;
- Exit the panel navigation mode - Esc .

3.4.3.3. Form command bar

A form command bar includes the commands that are directly related to the object displayed in the main form. There are standard groups Important and Generate; in addition, the developer can place on the panel the custom groups. Each command from the Important group is displayed with its own button on the panel. The Generate group is presented in the form of the panel submenu. Also all the groups created by the developer are displayed in the form of a submenu. The form commands, such as Save and close, are placed before the Important group commands.

If a group contains no commands, the group is not shown. If the configuration developer adds custom groups to this area, they are placed after the Generate group.

If all commands of the command bar can't fit the form, they are placed in a special submenu More, which is located after the last command of the panel. The exception is the help button, which is located to the right of the submenu More, if the help is provided for the form.

3.4.4. Nonvisual access to interface

3.4.4.1. General information

Screen readers can operate the following parts of the form element: name, tooltip, value of the form element, and some others depending on the screen reader. When an element receives a focus, the screen reader reads its name and value. The parts of form elements may be read depending on the screen reader used. Additionally, the current state of the control is read: collapsed/expanded (for example, for the drop-down list), set/cleared (for example, for the check box field). The elements tooltips are read. The exception to read is tooltips for buttons and hyperlinks: if the Tooltip display property is set to a value other than Pop Up, then this tooltip can't be recognized by the screen reader.

3.4.4.2. Description of the behavior of form elements and interface elements

Sections panel

The name of the section panel is read as Section panel in the current localization language. The tooltip is set from the Details property of the corresponding subsystem. The current active section is read (when switching sections as well).

Functions menu

The name of the panel is read as Functions menu in the current localization language. Menu items are grouped. The search results are grouped by subsystems.

Navigation panel of the form

The name of the panel is read as Form navigation panel in the current localization language. The element names are read according to the links in the form navigation panel.

Button and hyperlink

The name of the control is set to its header (automatic or overridden). The tooltip is set from the Tooltip property of the associated command or from the extended tooltip of the button itself. if the Tooltip display property is set to a value other than Pop Up, then this tooltip can't be recognized by the screen reader.
If the keyboard shortcut is specified for the associated command of the button, it is recognized.

Calendar

Names for the month cells are set to the full the date (including month and day). If the calendar displays more than 1 month, then all cells of one month are grouped under the name of the month and year.
The panel of months (if present) is read as a radio button group. After a month is selected the selected month is read.
Hyperlink today … is read as a word today and the current date.

Check box field

The name of the control is set to its header (automatic or overridden). The tooltip is set from a standard or extended tooltip, regardless of the display method.

Switch field

The name of the control is set to the header of each radio button item (automatic or overridden). For the first element of the radio button group the name is preceded by the header of the attribute edited by the radio button.

Command bar

The command bar is not read (other than Service and Settings). For the buttons of the command panel the names, tooltips and hot keys are read. If the button opens a drop-down list, then the list state is read: expanded or collapsed.
If the panel opens a menu, and the menu items can be checked (the Check property), then for such buttons the check state is read.

Input fields

The name of the control is set to its header (automatic or overridden). If there is a button to open a drop-down list, then the list state is read: expanded or collapsed. The tooltip is set from a standard or extended tooltip, regardless of the display method. If the Password property is set for a field, then the value of such a field is not read.

Picture

The element name is set to the tooltip without a hot key. The tooltip is set from a standard or extended tooltip, regardless of the display method. If the keyboard shortcut is specified for the picture, it is recognized.

Collapsible group

The name of the control is set to its header (automatic or overridden). The group state is read: collapsed/expanded.

Slider field

The name of the control is set to its header (automatic or overridden). The tooltip is set from a standard or extended tooltip, regardless of the display method.

Switch field

If the control corresponds to a group of radio button fields, the name of the group on the managed forms is set to the header (automatic or redefined). The names of the radio button fields correspond to their labels. The active element is read with the word selected. The tooltip is set from a standard or extended tooltip, regardless of the display method.

Tabs

The current tab is read (as well when switching tabs).

Table

Only columns that can be focused are processed. If there are more than one such column, then this form element is considered as a table, otherwise -it is considered as a list. If a form element is considered as a list, then its name is set from the column name (if it presents).
Table cells are read as CellContent ColumnName. If the reader processes a dynamic list and the current item is marked for deletion, then after the contents of the cell the text Marked for deletion is additionally read in the current localization language.
If the line is a grouping (if the table displays a tree), then for all its cells the current state of the grouping is additionally read: expanded or collapsed.

Spreadsheet documents

The spreadsheet document cells are read in the format CellContent Coordinates, where the coordinates fully correspond to the Microsoft Excel numbering:

- The A1 coordinate corresponds to the cell in the first column of the first row.
- Columns are coded by letters, rows -are coded by numbers.

If the line is a grouping (in the grouping mode), then for all its cells the current state of the grouping is additionally read: expanded or collapsed.

3.4.4.3. System unmanaged forms

The following unmanaged forms are read:

- Dialog box to adjust the precision of the calculator,
- Font choose dialog box,
- Color choose dialog box,
- Dialog box to follow a link,
- Dialog box to get a link,
- Dialog box to search in a spreadsheet document,
- Dialog box to search in a text document,
- Dialog box to access an infobase in thin client,
- Dialog box to start in thick client,
- Dialog box to start in thin client,

- Dialog box to set up the dialog box to start in thick client,

3.4.4.4. Restrictions

- Group headers are not read.
- Decoration (picture) headers are not read.
- Input hints (within input field) are not read.
- The focus can move to inactive items of the command bars, which are not read in this case.

Chapter 4. 1C:Enterprise language

1C:Enterprise is a flexible customizable system to solve a wide range of tasks in the field of business automation. Specific configuration algorithms are described in the 1C:Enterprise Designer and in software modules containing texts in the 1C:Enterprise language.

4.1. Purpose ans brief description of the 1C:Enterprise language

The 1C:Enterprise language is designed to describe (at the stage of configuration development) algorithms for an application task.
The 1C:Enterprise language (hereinafter - 'language') is a domain-specific programming language, specially designed to be used not only by professional programmers. In particular, all language operators have both Russian and English variants, which can be used simultaneously in one source text. The primary language described in this book is -Russian, however, for each operator of the language its English synonym is given.
Being relative simple, the language has some object-oriented features, for example, the rules for accessing the properties and methods of specialized data types (documents, directories, etc.) are similar to the properties and methods of objects used in other object-oriented languages. However, these specialized data types cannot be determined by the means of the language itself, but should be set in WYSIWYG mode.
The variables typing in the language is not strong, that is, the type of a variable is determined by its value. Variables do not require explicit declaration. The implicit definition of a variable is its first appearance in the left side of an assignment operator. You can also explicitly declare variables using the appropriate operator. You can use arrays, structures, matches and other universal collections of values.

4.2. Format of the modules source

4.2.1. What is module?

The modules in the configuration of the 1C:Enterprise system are not independent programs in common sense, since they are only part of the entire configuration. A module -is a text in the 1C:Enterprise language, which contains the texts of procedures and functions with the necessary algorithms. The modules are called by the system at certain moments of operations. Therefore the module does not have the formal boundaries in its description like: «Begin of module» - «End of module».
The location of a specific module is provided by Designer at points of the configuration that require the description of specific operation algorithms. These algorithms should be written in the form of procedures or functions that are called by the system automatically in predefined situations (for example, when you click a button in a dialog box).
Each individual module is processed by the system as a whole, therefore all procedures and functions of the software module are performed in a single context.

4.2.2. Module execution context

Each module is linked with the rest of the configuration. This link is called the execution **context** of the module.
There are two types of the context:

- global context,
- local context to execute specific module.

4.2.2.1. Global context

Global context is defined by:

- the values of the properties and methods of the global context;
- system enumerations and system value sets (for example, `DialogReturnCode` and `Characters`).

The global context is visible to all program modules and defines a common language environment for the configuration.

4.2.2.2. Local context

The local context of the module is defined by the specific location in the task configuration for which the module is called. The local context is visible only to a specific program module and defines for the module a set of objects, their properties and methods which are directly accessible to the module.

4.2.3. Module types

1C:Enterprise supports several software module types. They differ in their location and available context.

4.2.3.1. Managed application module

A managed application module is a module that is automatically executed when the configuration is loaded at the 1C:Enterprise startup in the following modes:

- Thin client
- web client,
- thick client in managed application mode.

The managed application module is designed to perform actions related to the end-user session (first of all the session start and session end processing). The managed application module is not available for procedures running on the server. It is recommended to implement there only the corresponding event handlers. You should remember that an unhandled exception in any event handler of the managed application module (`BeforeStart()`, `OnStart()`, `BeforeExit()`, `OnExit()`) can cause the system to crash.

For example, in the following example, the system doesn't crash:

```
Procedure OnStart()
  Try
    a=1/0;
  Exception
    Report("Divide by 0");
  EndTry;
EndProcedure
```

But the following example results in crash:

```
Procedure OnStart()
  a=1/0;
EndProcedure
```

The procedures and functions of the managed application module, as well as the variables with the `Export` keyword in the header, are available:

- in non-global client common modules,
- client procedures and command module functions,
- client procedures and functions of the managed form module.

In the context of the managed application module the following elements are available:

- the part of the global context that can be executed in the managed application;
- exported procedures and functions of any client common modules;
- exported procedures and functions of server non-global common modules that have the Server Call property set.

4.2.3.2. External connection module

The external connection module is located, like the application module, in the root section of the configuration. It contains event handler procedures that are initialized at application startup and exit in the external connection (COM connection) mode.

In the external connection module you can declare variables and also declare and define the procedures and functions that will be available to an external application.

The following 1C:Enterprise objects are available from outside through COM connections:

- Exported procedures and functions of the external connection module;
- Exported procedures and functions of common modules;
  - Including and excluding entire common modules is performed in their properties;
  - Including and excluding certain parts of common modules is performed using preprocessor instructions;
- Global 1C:Enterprise context

The module is available only in external connection sessions. No user interface is available in this mode.

4.2.3.3. Session module

The session module is a module that is executed automatically at 1C:Enterprise startup, when a configuration is being loaded.

The session module is intended for initializing session parameters and performing session-related operations. This common module is always executed in the privileged mode of 1C:Enterprise server. The session parameters are set in the event handler `SessionParametersSetting`.

A session module can contain only definitions of procedures and functions, can call procedures from general configuration modules, and does not contain exported procedures and functions.

The handler `SessionParametersSetting()` is called before call of the event handler `BeforeStart()` (or `OnStart()` for the external connection module).

When executing the event handler `SessionParametersSetting`, it may be necessary to determine which session starts: background job or another. This is the case, when the scope of background job data being initialized is materially different from that of other session data being initialized. You can determine the type of session using the `GetCurrentInfoBaseSession()` or `GetBackgroundJob()` methods. The first method returns the description of the current session - the `InfoBaseSession` object. Calling the `GetBackgroundJob()` method of the received object allows to clearly understand whether a background job session or some other session starts.

4.2.3.4. Common modules

Common modules are located in a dedicated branch of the metadata tree. Common modules are primarily designed to keep general-purpose configuration algorithms accessible from other modules. Common modules do not include the variable definition section or the main program section, thus they only have the procedures and functions section (see "Module structure").
You can declare and define procedures and functions that you want to be accessible from any configuration module in any common module.

4.2.3.5. Applied object modules

Each set of applied objects has a module. These are objects of the following types:

- Constant value managers,
- Catalogs
- Documents
- Reports
- Data processors
- Charts of characteristic types
- Charts of accounts
- Charts of calculation types
- Exchange plans
- Business processes
- Tasks
- Registers

The modules are located in the configuration branches that store such objects; they are actually properties of such objects. Each object has a module. These modules allow the declaration of variables, procedures, and functions that are accessible from 1C:Enterprise language outside of the object and supplement the object context.
The applied object context provides access to the attributes and tabular sections of the object, as well as to its methods and events.

4.2.3.6. Object manager modules

Each applied object has a manager intended for managing this object as a configuration object. The manager allows you to create objects, and work with their forms and templates. The manager module extends the manager functionality by introducing 1C:Enterprise language procedures and functions. You can describe methods of a configuration object (for example, a catalog) that are related to the configuration object itself rather than to an object instance in the database.
The context of a manager module is comprised of:

- the properties and methods of the global context;
- exported procedures and functions of global common modules (if these modules are compiled on the server);

- exported procedures and functions of non-global common modules (if these modules are compiled on the server);
- the local context of this module.

A manager module cannot have variables or module body.

When functions or procedures of a manager module are declared as exported ones, they can be accessed through the object manager.

For example, let us declare a function in the directory manager module `Counterparties`:

```
Function GetDebtorsList() Export
  …
EndFunction
```

Then the call of this function from the application code should have the following form:

```
Debtors = Catalogs.Counterparties.GetDebtorsList();
```

4.2.3.7. Form modules

These modules are stored in configuration forms. Each form has a module. These modules allow the declaration of variables, procedures, and functions that are accessible from 1C:Enterprise language outside of the object and supplement the object context.

The context of a form is comprised of:

- the local context of this form module, that is the attributes of the form to which the module "belongs";
- the properties and methods of the `ClientApplicationForm` object of the 1C:Enterprise language;
- properties and methods of the form expansion, determined by the type of the object which data are contained in the main attribute of the form;
- global context, including non-global common modules and exported functions and procedures of global common modules, however the procedure description in the form module (`&OnClient`, `&OnServer`, etc.) should be consistent with the properties set for the common module (Client (managed application), Server, etc.);
- exported variables, procedures, and functions of the managed application module.

4.2.3.8. Command modules

The command module is designed to describe the actions that the system should perform at the command call using the 1C:Enterprise language. The command module can only contain definitions of procedures and functions. The command module cannot have variables and module body.

The handler `CommandProcessing()` must be preceded by the preprocessor command `&OnClient`, since this is a place where the execution of the command begins.
The context of the custom procedures of the command module is comprised of:

- global context, including non-global common modules and exported functions and procedures of global common modules, however the procedure description in the form module (`&OnClient`, `&OnServer`, etc.) should be consistent with the properties set for the common module (Client (managed application), Server, etc.);
- the local context of this command module.

The context of the server procedures of the command module is comprised of:

- the properties and methods of the global context;
- exported procedures and functions of global common modules (if these modules are compiled on the server);
- non-global common modules if these modules are compiled on the server; exported methods of such modules are available;
- server methods of the command module.

In the command module you can describe methods with the keyword `Export`. However, they cannot be used outside of this module. 1C:Enterprise language doesn't provide access to commands and, therefore, to their context as well.

4.2.4. Module format

The structure of the module can be divided into the following sections:

- variable definition section,
- procedures and functions section,
- main program section.

In a specific module any of the sections can be missing.
The variable definition section is placed from the beginning of the module text to the first operator `Procedure`, or `Function`, or any executable statement. This section can contain only variable declaration statements `Var`.
The section of procedures and functions is placed from the first statement `Procedure` or `Function` to any executable statement outside the body of a description of procedures and functions.
The main program section is placed from the first executable statement outside the body of the last procedure or function to the end of the module. This section can contain only executable statements. The main program section is executed when the module is initialized. Usually, the main program section should contain the variable initialization statements specifying necessary values which must be assigned before the first call of any procedure or function of the module.
The source code of the module may consist of statements and comments.

4.2.4.1. Comments

The comments can be used to place all kinds of explanations of the module operation directly in the source code of the module. Best practice is to put into the source text a comment fully describing the algorithm. When a program is run, comments are skipped. In the module text comments begin with a pair of characters `//` and ends with the end of the line. This means that you can start a comment from the beginning of the line or write it after a statement on the same line. You cannot write a statement on the same line after the start of a comment; you should end the comment with the end of the line.

```
A=B; // This is a comment
// This is a comment, too
```

A comment preceding a procedure or function may be used by the context tooltip of Designer..

4.2.4.2. Format of statements

Statements have the form of a standard procedure call, with the exception of the assignment (`A = B;`) and the syntax constructs of the 1C:Enterprise language (for example, such as `For`, `While`, `If`). Statements must be separated by a semicolon. The end of the line does not end an statement, i.e., statements can freely pass through the lines and continue on another line. A line can contain arbitrary number of statements separated by a semicolon.
The language statements in a module can be divided into two categories: variable declaration statements and executable statements.
Variable declaration statements create variable names that can be used in executable statements.
Any executable statement can have a label which can be used as a transition point in the `Goto` statement.
In general, the format of the language statement is as follows:

```
~label:Statement[(parameters)] [ДобКлючевоеСлово];
```

Labels are special identifiers starting with a tilde sign following by a sequence of letters, numbers and underscore characters. To mark an statement, place a label in front of it and the following colon.

```
~label:A=B;
```

4.2.4.3. Names of variables, procedures, and functions

The name of a variable, a declared procedure, or a function can be any sequence of letters, numbers, and underscores, beginning with a letter or underscore. Newly created names can't match the reserved words of the language or the names of the properties directly accessible in the current context. The names of variables, procedures and functions are not case-sensitive.

4.2.4.4. Modules language

The 1C:Enterprise language is bilingual. Almost all reserved words, the names of the types of values, properties, methods, events have two names: Russian and English. The exceptions are words that have no Russian analogues. In the text of modules these names can be freely mixed using Russian and English names without any restrictions.

4.2.4.5. Registers of letters in modules code

Names of variables, properties, methods, procedures, functions, as well as functions of the 1C:Enterprise language are not case-sensitive (you can use upper case and lower case characters).

4.2.4.6. Reserved words

The following keywords are reserved and cannot be used in user created variable names, configuration object attributes, and declared procedures and functions. In this version of the language, each of the keywords has two forms, -Russian and English.

Russian	English
If	If
Then	Then
ElseIf	ElsIf
Else	Else
EndIf	EndIf
For	For
Each	Each
From	In
To	To
While	While
Do	Do
EndDo	EndDo
Procedure	Procedure
Function	Function
EndProcedure	EndProcedure
EndFunction	EndFunction
Перем	Var
Перейти	Goto
Return	Return
Continue	Continue
Abort	Break
AND	And
Или	Or
Not	Not

Russian	English
`Try`	`Try`
`Exception`	`Except`
`Raise`	`Raise`
`EndTry`	`EndTry`
`New`	`New`
`Execute`	`Execute`

NOTE. The keywords are not case-sensitive.

4.2.5. Special characters used in source code

Character	Description
//	A double-slash begins a comment. A comment is all the text from this character to the end of the current line.
|	It is used only in string constants at the beginning of a line and means that this line is a continuation of the previous one (line break)
~	Starts a statement label
:	Ends a statement label
;	Statements separator
( )	Parentheses enclose a list of parameters of methods, procedures, functions and constructors. In addition they are used in 1C:Enterprise language expressions
[ ]	The square brackets statement calls the properties of the object using a string representation of the property name. It is also possible to call the elements of collections by index or other parameter.
,	Separates parameters in the parameter list in methods, procedures, functions, and constructors
" "	Encloses string literals
' '	Encloses date literals
.	Decimal point in numeric literals. The separator to refer the properties and methods of 1C:Enterprise language objects.
+	Addition. Strings concatenation.
-	Subtraction
*	Multiplication
/	Division

Character	Description
%	Remainder of division. Dividend and divisor can be fractional values.
>	Logic operation Greater
>=	Logic operation Greater or equal
<	Logic operation Less
<=	Logic operation Less or equal
=	Assignment. Logic operation Equal
<>	Logic operation Not Equal

4.3. Primitive data types

The 1C:Enterprise language supports a set of primitive data types. Most primitive data types allow to use literals in the module code, that is, to specify the value of the corresponding type directly in the module.

```
// Example of a String type literal
A = "My string";

// Example of a Boolean type literal
B = True;

// Example of a Number type literal
B = 12345.6789;
```

NULL

Description:

Values of this type are used only to specify the missing value in a database operations, for example, when joining tables.

Literals:

`NULL`

Boolean

Description:

Variables of this type have two values, -`True` and `False`, given by the corresponding literals. Values of this type are returned as a result of evaluating logical expressions.

Note.

The 1C:Enterprise language comparison operations use logical expressions. This means that comparison expression should not necessary has the form:

```
If MyVariable = True Then

EndIf;
```

The following is enough:

```
If MyVariable Then

EndIf;
```

Literals:

- `True`
- `False`

Date

Description:

Values of this type contain an AD date (starting from January, 1 0001) and a time with 0.1 millisecond accuracy.

Literals:

A string of numbers enclosed in single quotes of the form `'YYYYMMDDhhmmss'`, where:

- `YYYY` - a 4-digit number of year
- `MM` - a 2-digit number of month
- `DD` - a 2-digit number of day
- `hh` - a number of hour (in 24-hour format)
- `mm` - a 2-digit number of minute
- `ss` - a 2-digit number of second

A literal of the `Date` type must contain year, month, and day. You can omit seconds or the entire time. If seconds are not specified, they are set to 0. A starting date is `'00010101'`. All chars other than digits are ignored in the date literal. You can place separators in the date literal to simplify its presentation.

Example:

```
Date('2017\03\23 10~45~25') = 23.03.2017 10:45:25
Date('2017\03\23 10~45~25') = 23.03.2017 10:45:25
Date('2017.03.23 10:45:25') = 23.03.2017 10:45:25
Date('20170323104525') = 23.03.2017 10:45:25
```

Number

Description:

The numeric type can represent any decimal number. The language defines the following basic arithmetic operations on data of numeric type: addition, subtraction, multiplication and division.

IMPORTANT! The maximum width of the number -is 32 characters.

Literals:

A set of numbers directly in the code of a module of the form:
`[+|-]{0|1|2|3|4|5|6|7|8|9}[.{0|1|2|3|4|5|6|7|8|9}]`
Point is used as a separator for the integer and fractional parts.

Example:

```
A = 15;
B =  968.612;
```

String

Description:

Values of this type contain a string of arbitrary length in Unicode format. The string is encoded in UTF-16 format. If you need to specify a surrogate pair in the string, you should use two calls of the `Char()` function specifying the value of each code unit from the surrogate pair in turn.

Literals:

String literals are a set of characters enclosed in quotes. For the quote character, you need to type two quotes in a row ("").
In addition, you can use "multiline" string constants. In the source code, the multiline constants can be specified in two ways:

- Between fragments representing individual lines of a multiline string, no characters except for spaces, line breaks and comment lines can be inserted.
- Each individual fragment is not closed with quotes, and each next line should start with hyphenation character “|” (vertical bar). In this variant, comments are allowed if the line starts with the comment character “//”.

Example:

```
// String example
MyString = "This is a valid string";

// Example 1: Multiline string
MyMultilineString = "This is
     |a valid
     |multiline
     |string";

// Example 2: Multiline string
MyMultilineString = "This is" // This is a comment
     "a valid"
     "multiline"
     "string";

// Example 3: String containing quotes
CompanyName = """Cornflower"" LLC"

// Example 4: Surrogate pair
TextWithSurrogatePair = "Hieroglyph: " + Char(55401) + Char(56997);
```

`CompanyName` from Example 3 is displayed and printed as follows:

```
"Cornflower" LLC
```

Undefined

Description:

The value of this type is used when it is necessary to use an empty value that does not belong to any other type. For example, attributes of a composite type initially have such values. There is one single value of this type, given by literal.

Literals:

`Undefined`

Type

Description:

Values of this type are used to identify types of values. This is necessary to define and compare types. This type has no literals and can be returned by the 1C:Enterprise language functions `TypeOf` and `Type` (see below).

4.4. Assignment statement

Assignment statement (=)

Description:

The assignment statement (the "`=`" character) means the assignment of the value `<Source>` to a variable `<Destination>`.

Assigning a value to an undeclared variable that matches a property name available in the current context may result to runtime error or configuration check error. To avoid errors, it is recommended to explicitly declare variables with such names using the operator `Var`.

Syntax:

```
<Destination> = <Source>;
```

Parameters:

<Destination>

The destination can be a variable or a property of 1C:Enterprise language object that can be written.

<Source>

The expression giving the value to assign.

Example:

```
A = B;
Page1 = "777";
DocumentDate = '20020717';
```

4.5. Language expressions

An **expression** -is a mathematical, logical, or string formula, consisting of the corresponding operations to calculate the value. Mathematical and logical expression can appear to the right of the equal sign in the assignment statements, and to be a parameter of procedures or functions. A logical expression can also be a condition in the `If`, `While`, and `For` control structures. Expressions consist of constants, variables, and functions joined by symbols of logical and/or arithmetic operations.

4.5.1. Arithmetic operations

The language defines the following types of arithmetic operations.

Name	Expression
Addition	(Op1 + Op2)
Subtraction	(Op1 - Op2)
Multiplication	(Op1 * Op2)
Division	(Op1 / Op2)
Remainder	(Op1 % Op2)
Unary minus	(-Op1)

Arithmetic operations have one or two operands, and their types determine the operation semantics. The semantic variant of the operation is determined by the

first operand. If the type of the second operand mismatch the required type, then its value is converted to the required type in accordance with the type conversion rules. If the type of the first operand does not correspond to any of the valid types, then depending on the situation, a type conversion may be performed or an execution error condition may be raised.

Operation	Action description
Addition is defined for the following types of operands.	`Number` + `Number` `Date` + `Number` (the number of seconds is added to the date)
Subtraction is defined for the following types of operands.	`Number` - `Number` `Date` - `Number` (the number of seconds is subtracted from the date) `Date` - `Date` (the result is the number of seconds between the dates)
Multiplication	`Number` * `Number`
Division	`Number` / `Number`
Remainder	`Number` % `Number`

When performing arithmetic operations with the `Date` type, you should remember that the precision of the result is 4 decimal places (the time precision -is hundreds of microseconds). However, when you write such value to database (in the attribute of the corresponding type), the fractional part of the number is discarded. Thus it is recommended to round up the value of the `Number` type to integer in arithmetic operations with the value of the `Date` type.

4.5.2. Concatenation

The concatenation ("+") is used to append one string to another. The length of the resulting string is equal to the sum of the lengths of the joined strings. In case if the data type of the second or subsequent operands mismatch the string type, their value is converted to the string type in accordance with the type conversion rules.

```
FullName = Surname + " " + FirstName + " " + MiddleName;
```

4.5.3. Logical operations

The logical operation compares the operands and generates a `Boolean` value: `True` or `False`. There are two types of logical operations: comparison operations and Boolean operations. In comparison operations two values are compared. Boolean operations are performed on values of type `Boolean` implementing the Boolean algebra. Symbols of Boolean operations can be combined to form compound operations.

Comparison operations:

The language defines the following types of comparison operations.

Operation	**Expression**
Greater than	Op1 > Op2
Greater or equal	Op1 >= Op2
Equal to	Op1 = Op2
Not equal	Op1 <> Op2
Less than	Op1 < Op2
Less or equal	Op1 <= Op2

Comparison operations are defined for the following types of operands.

Operation	**Expression**

Greater than	`Number` > `Number`
	`Number` > `Boolean`
	`String` > `String`
	`Date` > `Date`

Greater or equal	`Number` >= `Number`
	`Number` >= `Boolean`
	`String` >= `String`
	`Date` >= `Date`

Less than	`Number` < `Number`
	`Number` < `Boolean`
	`String` < `String`
	`Date` < `Date`

Less or equal	`Number` <= `Number`
	`Number` <= `Boolean`
	`String` <= `String`
	`Date` <= `Date`

Equal to	Any type = Any type
Not equal	Any type <> Any type

If the values of the type `Number` and `Boolean` are compared then the value of the type `Boolean` is converted to the type `Number` according to the following rules:

- the value `True` is converted to 1;
- the value `False` is converted to 0.

After that, the values of `Number` types are compared.

Boolean operations:

The language defines the following types of Boolean operations.

`And`	conjunction (Boolean `And`)

`Or`	disjunction (Boolean `Or`)
`Not`	logical negation (Boolean negation `Not`)

Logical expressions are evaluated from left to right. Use parentheses to avoid ambiguity and to control the sequence of operands.

Levels of priority for logical operations:

Level 1	operands in parentheses
Level 2	`Not`
Level 3	`AND`
Level 4	`Or`

NOTE. When evaluating a logical expression, only the necessary parts of the expression are calculated. For example, in the expression `(Price > 0) And CheckAmount()`, if `Price <= 0` the `CheckAmount()` function is not called.

4.5.4. Priorities for evaluating 1C:Enterprise language expressions

The following table lists the 1C:Enterprise language statements in order of increasing priority. Statements with the equal priority are calculated from left to right.

Operation	Description
`Or`	Logical addition
`And`	Logical multiplication
`Not`	Logical negation
`<`, `<=`, `=`, `>=`, `>`, `<>`	Comparison
`+`, `-`	Addition, subtraction
`*`, `/`, `%`	Multiplication, division
`-`,`+`	Unary plus and minus
`.`, `()`	Dereferencing, method call, etc.

Use parentheses to avoid ambiguity and to control the sequence of operands.

4.6. Statements and syntax structures

? (calculate the expression by the condition)

Description:

Allows to calculate one of two specified expressions depending on the result of the evaluation of a logical expression.

Syntax:

`?(<Logical expression>, <Expression 1>, <Expression 2>)`

Parameters:

`<Logical expression>`

A logical expression which result determines the resulting expression to be calculated. If the result of its calculation is `True`, then `<Expression 1>` is calculated. If the result is `False`, then `<Expression 2>` is calculated.

`<Expression 1>`

The result expression to be evaluated if the result of the logical expression is `True`.

`<Expression 2>`

The result expression to be evaluated if the result of the logical expression is `False`.

`Returns:`

The result of evaluating the result expression.

Example:

```
Status = ? (GetDiscount() > 10, "Special customer", "Regular
customer");
Warning(Status);
```

Raise

Description:

Using this form of operator rises new exception.

Syntax:

`Raise <Expression>`

Parameters:

<Expression>

The result of the evaluation of the expression is converted to a string, and this string is used as a description of the exception.

Example:

```
Raise "The document can not be posted."
```

See also:

Description of the `Try` operator.

Execute

Description:

Executes a code fragment that is passed to it as a string value.

IMPORTANT! Implementation of important part of the functionality of application solutions using this method is not recommended.
NOTE. The executable code cannot contain additional procedures or functions, since the code itself is executed within the procedure or function in which this operator is placed. In addition it can't contain explicit variable declarations.

Syntax:

```
Execute(<String>)
```

Parameters:

<String>

A string with the executable code.

Example:

```
// Displays the current date in the message window
Execute("Report(CurrentDate())");
```

AddHandler

Description:

Adds event handler.
When an event handler is added, the matching of the number of event parameters with the number of method parameters assigned as the handler is checked.

Syntax:

```
AddHandler <Event>, <EventHandler>;
```

Parameters:

<Event>

The event to which the handler is added.
The event is specified in the form <Expression>.<EventName>, where:

- <Expression> - arbitrarty 1C:Enterprise language expression. It should return the object to which event the handler is added.
- <EventName> - the event ID (name).

<EventHandler>

Procedure/function to handle the event.
An event handler can be a method of 1C:Enterprise language object. In this case <EventHandler> is specified as <Expression>.<HandlerName>, where:

- <Expression> - arbitrarty 1C:Enterprise language expression. It should return the object which method handles the event.
- <HandlerName> -name of the method to handle the event.

In addition, a procedure/function in the scope can be specified as an event handler. In this case, the event handler is specified as the name of the procedure/function.
It is possible to subscribe to events (in COM objects) of the same name, but with a different number of parameters. To do this, you need to create several handlers in the 1C:Enterprise language (each with a unique name and a required number of parameters), and the subscription mechanism choose the right handler for the given subscription.

Example:

```
Processing = DataProcessors.Document Control.Create();
Invoice = Documents.Invoice.CreateDocument();
AddHandler Invoice.OnWrite, Processing.OnDocumentWrite;

msword = New COMObject("Word.Application");
AddHandler msword.DocumentChange, OnDocumentChange;

Procedure OnDocumentChange()

  Report("Document is changed");

EndProcedure
```

Example of cases with different parameters:

```
// Handler without parameters
Procedure EventHandling()

EndProcedure

// Handler with one parameter
Procedure EventHandling2(Parameter)

EndProcedure

// The object can generate events with
// and without parameters
Object = New COMObject("Test.Events");
AddHandler Object.TestEvent, EventHandling
AddHandler Object.TestEvent, EventHandling2
```

For

Description:

Loop statement `For` is designed for repetition of statements inside the `Do` - `EndDo` structure. Before the loop start the value of `<Expression 1>` is assigned to the variable `<Variable name>`. The value of `<Variable name>` is automatically increased with each loop pass. The increment of the counter for each loop pass is 1. The loop is executed until the value of the variable `<Variable name>` is less than or equal to the value of `<Expression 2>`. The loop condition is always checked first, before the execution of the loop.

Syntax:

```
For <Variable name> = <Expression 1> To <Expression 2>
Do
// Statements
[Break;]
// Statements
[Continue;]
// Statements
```

```
EndDo;
```

```
For <Variable name> = <Expression 1> To <Expression 2>
Do
// Statements
```

Parameters:

<Variable name>

The variable ID (loop counter) which value is automatically increased by 1 each time the loop is repeated. So called loop counter.

<Expression 1>

A numeric expression that specifies the initial value assigned to the loop counter on the first pass of the loop.

To

Syntax link for the parameter <Expression 2>.

<Expression 2>

The maximum value of the loop counter. When the variable `<Variable name>` becomes greater than `<Expression 2>`, the execution of the `For` loop statement terminates.

Do

The statements following the `Do` keyword are executed while the value of the variable `<Variable name>` is less than or equal to the value of `<Expression 2>`.

// Statements

One or several executable statements.

Break

Allows to interrupt the loop at any point. After executing this statement the control is transferred to the statement following the `EndDo` keyword.

Continue

Immediately transfers control to the beginning of the cycle, where the conditions for the execution of the cycle are evaluated and checked. The rest statements in the loop body are not executed at this loop iteration.

EndDo

The final keyword in the structure of the loop statement.

Example:

```
// Loop on the current month days
LastDayOfMonth = Day(EndOfMonth(CurrentDate()));

For CurrDay = 1 To LastDayOfMonth Do

  Status("Now processing day: "+ CurrDay);

  // Statements to process the next day of the month
  . . .

EndDo;
```

For each

Description:

The loop `For each` allows to cyclically pass collections of values. At each iteration of the loop, a new collection element is returned. The processing is carried out until all elements of the collection are processed, or it can be terminated when the `Break` statement is executed.

Syntax:

```
For each <Variable name 1> In <Variable name 2> Do
// Statements
[Break;]
// Statements
[Continue;]
// Statements
EndDo
```

Parameters:

<Variable name 1>

A variable that is assigned the value of the next element of the collection on each loop iteration.

From

Syntax link for the parameter `<Variable name 2>`.

<Variable name 2>

The variable or expression representing the collection. Elements of this collection are assigned to the `<Variable name 1>` parameter.

Do

The statements following the `Do` keyword are executed until all elements of the collection are processed.

// Statements

One or several executable statements.

Break

Allows to interrupt the loop at any point. After executing this statement the control is transferred to the statement following the `EndDo` keyword.

Continue

Immediately transfers control to the beginning of the cycle, where the conditions for the execution of the cycle are evaluated and checked. The rest statements in the loop body are not executed at this loop iteration.

EndDo

The final keyword in the structure of the loop statement.

Example:

```
// Iteration over the tabular section of the document.
Document = Documents.Invoice.FindByCode(12345);

// Check if the document is found
If Not Document.Empty() Then

  For Each ContentString In Document.Content Do

    Status("String: " + Document.Content.Index(ContentString)+1);
    // Processing the next tabular section line
    ...

  EndDo;

EndIf;
```

If

Description:

The `If` operator controls program execution based on the result of one or several logical expressions. The operator can contain any number of operator groups provided in `ElsIf - Then` pairs.

Syntax:

```
If <Logical expression> Then
// Statemets
[ElsIf <Logical expression> Then]
// Statemets
[Иначе]
// Statemets
EndIf;
```

```
[Else]
```

Parameters:

`<Logical expression>`

Logical expression.

Then

Operators following keyword `Then` are executed if the result of the logical expression is `True`.

// Statements

One or several executable statements.

ElseIf

The logical expression following keyword `ElsIf` is only calculated when all of the conditions in the `If` clause and all preceding `ElsIf` clauses are `False`. Operators following `ElsIf` - `Then` are only executed when the result of the logical expression in this `ElsIf` is `True`.

Else

Operators following `Else` are executed when all of the conditions in the `If` clause and all preceding `ElsIf` clauses are `False`.

EndIf

Denotes the end of the conditional execution structure.

Example:

```
If WeekDay(CurrentDate()) = 6 Then
  Report("Today is Saturday.");
ElsIf WeekDay(CurrentDate()) = 7 Then
  Report("Today is Sunday.");
Else
  Report("Today is working day.");
EndIf;
```

New

Description:

This statement allows to create a value of the specified type. It can be used only for those types for which the creation of new values is allowed. For applied objects, it is necessary to use the functional form of the `New` operator (option 2), since when checking modules in Designer the types for applied objects are not defined yet.

When using the functional form of the `New` operator, you should consider that the dereference operation cannot be applied directly to the result of the operator. You should first assign the result of the `New` operator to a variable, and then use this variable.

Syntax (option 1):

```
New <Type name>[(<Param 1>, ..., <Param N>)]
```

Parameters:

`Type name`

Specifies the name of the type which value is created.

`<Param 1>, ..., <Param N>`

If parameters are defined in the constructors for this type, they can be specified after the type name in parenthesis. The valid number of parameters and their purpose are specified in the description of the object constructors.

Example:

```
// Example of creating an array of three elements.
Array = New Array(3);
```

Syntax (option 2):

`New (<Type>[, <Constructor parameters>])`

Parameters:

`Type`

Type name or value of the type `Type`.

`<Constructor parameters>`

An array of constructor parameters.

Example:

```
ValueType = Type("StringQualifiers");
Parameters = New Array(2);
Parameters[0] = 20;
Parameters[1] = AllowedLength.Variable;
StrQualif = New(ValueType, Parameters);
```

Goto

Description:

Unconditional transfer of control to another program statement. Transfers control from one statement to another.
The scope of the statement is limited to a module, procedure, or function; it cannot transfer control beyond the limits of the module, procedure, or function.

NOTE 1. The label in this statement cannot be a label of the `Procedure` or `Function` statement.
NOTE 2. The unconditional Goto statement cannot be used to transfer control to the statements inside the structures: `While` - `EndDo`, `For` - `EndDo`, `For each` - `EndDo`, `If` - `EndIf`, `Try` - `Exception` - `EndTry` from outside.

Syntax:

`Goto <Label>;`

Example:

```
Goto ~Label1;
...
~Label1: Report("Go to the label is performed");
```

Var

Description:

Explicitly declares a variable.

Syntax:

```
Перем <Имя переменной 1> [Экспорт] [, <Имя переменной 2>, …];

Var <Variable name 1> [Export] [, <Variable name 2>, …];
```

Parameters:

<Variable name 1>[, <Variable name 2>, …]

Specifies name or names of the declared variable(s).

Export

Optional keyword. Indicates that this variable is available when accessing the context of this module from other modules. This keyword must be specified for each declared variable separately. It is meaningless when declaring variables of separate procedures or functions.

Example:

```
// Example of single variable declaration
Var A Export;
Var B;

// Example of several variables declaration in single statement
Var A, B Export;
```

Implicit variable declaration:

The language doesn't require to declare variables explicitly. The implicit definition of a variable is its first appearance in the left side of an assignment operator. The type of a variable is determined by the type of value assigned to it. Variables that were not previously declared explicitly or implicitly can't be used in expressions.

Variable scope:

The scope of variable depends on where it is declared in the configuration. There are three areas where variables can be declared:

- In the section of variables definition of the managed application module. These variables are global.

- In the section of variables definition of a module. These are module variables.
- In procedure or in function. These variables are local.

Global variables declared with the `Export` keyword are available for use in executable statements, expressions, in any procedure or function of any custom module of the configuration.
Module variables are available for use in executable statements, expressions, in any procedure or function of the module where they are declared. If they are declared with the `Export` keyword, they are visible from other modules via the context of the module where they are declared.
Local variables are available within the procedure or function where they are declared.
If a variable is defined as global, then it is visible from all the procedures and functions of any custom module of the configuration. If a variable is defined in a procedure or function, then its scope is the procedure or function.
Thus, if two variables with the same name are used in two different procedures of a module and the names of these variables are not declared as global, they are two different variables that are local to the procedures. If a variable is declared as a global variable, then any use of the name of this variable calls the same variable.
The only way to create a local variable in a procedure with a name that matches the name of a variable defined as global -is to declare the variable explicitly using the `Var` operator.

While

Description:

Loop statement `While` is designed for repetition of statements inside the `Do` - `EndDo` structure. The loop is iterated until its logical expression is `True`. The loop condition is always checked first, before the execution of the loop.

Syntax:

```
While <Logical expression> Do
// Statements
[Break;]
// Statements
[Continue;]
// Statements
EndDo
```

```
While <Logical expression> Do
// Statements
```

Parameters:

<Logical expression>

Logical expression.

Do

The statements following the `Do` keyword are executed until the logical expression gives `True`.

// Statements

One or several executable statements.

Break

Allows to interrupt the loop at any point. After executing this statement the control is transferred to the statement following the `EndDo` keyword.

Continue

Immediately transfers control to the beginning of the cycle, where the conditions for the execution of the cycle are evaluated and checked. The rest statements in the loop body are not executed at this loop iteration.

EndDo

The final keyword in the structure of the loop statement.

Example:

```
DocSelection = Documents.Invoice.Select();

// Loop on all documents
While DocSelection.Next() Do

  // Display the document in the status bar
  Status("Process document #" + DocSelection.Number);

  // Statements to process the document

EndDo;
```

Try

Description:

The `Try` operator manages the code execution based on errors (exceptions) that arise during module execution, and defines how these exceptions are handled. Errors (exceptions) originate from module runtime errors. User-defined exceptions are not supported.

If a runtime error occurs during the execution of an operator sequence, the execution of the operator that caused the error is stopped, and the control is passed to the first operator from the exception handling operator sequence. The control is passed even if the error is caused by an operator within a procedure or function called from a Try operator. If an error occurs in a procedure or function, the procedure or function is terminated and its local variables -are destroyed. This is true for any nesting level of a procedure or function call. Once an exception operator sequence is executed, the control is passed to the operator following the `EndTry` keyword. If the Try operator sequence is executed without errors, the exception operator sequence is skipped, and the control is also passed to the operator following the `EndTry` keyword.

`Try` - `Except` - `EndTry` clauses can be nested. If an exception is raised, the control is passed to the handler that contains the Try operator where the error occurred. If a `Raise` operator is called in the exception operator sequence of this handler, the control is passed to the exception handler one level higher, and so on. If there is no exception handler on a higher level, the exception is handled at the system level and the module execution is stopped.

The built-in functions `ErrorDescription()` and `ErrorInfo()` can help in diagnostics (see the description of the functions of 1C:Enterprise language). In addition you should consider that getting into the nested exception handler causes the information about the "parent" exception to be lost.

The statement `Try - Except - EndTry` can't intercept the following errors:

- Compilation errors of 1C:Enterprise language modules at runtime.
- Run-time errors in the handlers `BeforeClose()` and `OnClose()` of managed forms.

Such errors always are displayed by the 1C:Enterprise and 1C:Enterprise language processing is terminated.

Syntax:

```
Try
// Try statements
Exception
// Except statements
[Rise;]
// Except statements
EndTry;
```

```
Try
// Try statements
Except
// Except statements
[Raise;]
// Except statements
```

```
EndTry;
```

Parameters:

// Try statements

One or several executable statements.

Exception

Operators following the `Except` keyword are executed if a runtime error occurs during the execution of the operator sequence.

// Except statements

An executable operator or a sequence of operators that handle the exception.

Raise

The operator that generates an exception in cases when, even though the exception is handled by the exception operators, it is necessary to stop the module execution and generate a runtime error. This operator is valid only inside the `Except` - `EndTry` keyword pair.
The execution of this operator stops the execution of the exception operator sequence and starts the search for a higher-level exception handler (when the Try operators are nested). If such exception handler is found, the control is passed to its first operator. Otherwise the exception is handled at the system level, a message about the original error is issued, and the module execution is stopped.

EndTry

A keyword that ends the exception handling block.

Example:

```
Procedure BuildInExcel()

  Try

    // Attempt to call MS Excel
    Table = New ComObject("Excel.Application");

  Exception

    DoMessageBox(ErrorDescription());
    Return;

  EndTry;

  // Statements to build the report
  ...

EndProcedure
```

Procedure

Description:

The `Procedure` keyword denotes the beginning of code fragment that can be called from any point of the module using the `ProcedureName()` operator with a list of parameters (parentheses are mandatory, even if no parameters are passed). If an ordinary application module, application module, or common program module contains the `Export` keyword in the procedure declaration, it means that the procedure is accessible from all other configuration modules.

When the `Return` operator is executed, the procedure ends and returns control to the point of call. If procedure text does not contain the `Return` operator, an implicit `Return` operator is executed after the last executable operator. The `EndProcedure` operator denotes the end of the procedure.

The variables declared in the local variable declaration area of the procedure body are local variables and therefore accessible in this procedure only (except when they are used as parameters for calling other procedures, functions, or methods).

NOTE. Keywords `Procedure`, `EndProcedure` are not statements, but statement brackets, so they should not end with a semicolon (this can lead to module execution errors).

Syntax:

```
Процедура <ИмяПроцедуры>([[Знач] <Парам 1>
[=<ДефЗнач>], ... ,[Знач] <Парам N>
[=<ДефЗнач>]])[Экспорт]
// Local variables declaration;
// Statements;
```

```
...
[Return;]
// Statements;
...
EndProcedure
```

```
Procedure <ProcedureName>([[Val] <Param 1>
[=<DefValue>], ... ,[Val] <Param
N>[=<DefValue>]])[Export]
// Local variables declaration;
// Statements;
...
[Return;]
// Statements;
...
EndProcedure
```

Parameters:

<ProcedureName>

Specifies the procedure name.

Val

An optional keyword. The parameter that follows the keyword is passed as a value (changing the value of the formal parameter during the procedure execution does not affect the actual parameter that is passed at the procedure call). If this keyword is not specified, the procedure parameter is passed as a reference (changing the formal parameter value inside the procedure changes the actual parameter value).

<Param 1>, ..., <Param N>

Optional comma-separated list of formal parameters. Formal parameter values must match the values of actual parameters that are passed to the procedure. This list contains the parameter names that are used in the procedure text. The list of formal parameters can be empty.

=<DefValue>

An optional default parameter value. You can include parameters with default values anywhere in the list of formal parameters. For more information, see Passing procedure and function parameters.

Export

An optional keyword that specifies that this procedure is accessible from other modules.

// Local variables declaration

Declarations of local variables that can be referenced in this procedure only (see `Var` operator description).

// Statements

Executable statements of the procedure.

Return

An optional keyword that ends the procedure and returns to the code point where the procedure was called. The use of this operator in the procedure is optional.

EndProcedure

Mandatory keyword that identifies the end of procedure. Returns to the code point where the procedure was called.

Example:

```
Var Glob;

// The procedure definition
Procedure MyProcedure(Par1, Par2, Par3) Export

  Glob = Glob + Par1 + Par2 + Par3;
  Return;

EndProcedure

Glob = 123;
MyProcedure (5, 6, 7); // The call of the procedure
```

RemoveHandler

Description:

Removes event handler.
When an event handler is removed, the matching of the number of event parameters with the number of method parameters assigned as the handler is checked.

Syntax:

```
RemoveHandler <Event>, <EventHandler>;


RemoveHandler <Event>, <EventHandler>;
```

Parameters:

<Event>

The event which handler is removed.
The event is specified in the form `<Expression>.<EventName>`, where:

- `<Expression>` - arbitrarty 1C:Enterprise language expression. It should return an object which event handler is removed.

- `<EventName>` - the event ID (name).

`<EventHandler>`

Procedure/function to handle the event.
An event handler can be a method of 1C:Enterprise language object. In this case `<EventHandler>` is specified as `<Expression>.<HandlerName>`, where:

- `<Expression>` - arbitrarty 1C:Enterprise language expression. It should return the object which method handles the event.
- `<HandlerName>` -name of the method to handle the event.

In addition, a procedure/function in the scope can be specified as an event handler. In this case, the event handler is specified as the name of the procedure/function.

Example:

```
RemoveHandler Invoice.OnWrite, Processing.OnDocumentWrite;
```

Function

Description:

The `Function` keyword denotes the beginning of function code that can be called from any point of the module using the name `FunctionName()` with a list of parameters (parentheses are mandatory, even if no parameters are passed).
If an application module or common program module contains the `Export` keyword in the function declaration, it means that the function is accessible from all other configuration modules.
The execution of the function ends on the statement `Return`. Functions differ from procedures only in that they return `ReturnValue`. The `EndFunction` operator denotes the end of the function.
Any function in the module source code can be called like a procedure, that is, the language allows not to accept a return value from the function.
If the keyword `Return` is not specified in the function body or the module string containing the statement is not executed, then the function returns a value of the `Undefined` type.
The variables declared in the local variable declaration area of the function body are local variables and therefore accessible in this function only (except when they are used as parameters for calling other procedures, functions, or methods).

NOTE. Keywords `Function`, `EndFunction` are not statements, but statement brackets, so they should not end with a semicolon (this can lead to module execution errors).

Syntax:

```
Функция <ИмяФункции>([[Знач] <Парам 1>[=<ДефЗнач>],
... ,[Знач] <Парам N>[=<ДефЗнач>]])[Экспорт]
// Local variables declaration;
// Statements;
...
Return <ReturnValue>;
// Statements;
...
EndFunction

Function <FunctionName>([[Val] <Param 1>[=<DefValue>],
... ,[Val] <Param N>[=<DefValue>]])[Export]
// Local variables declaration;
// Statements;
...
Return <ReturnValue>;
// Statements;
...
EndFunction
```

Parameters:

<FunctionName>

Specifies the function name.

Val

An optional keyword. The parameter that follows the keyword is passed as a value (changing the value of the formal parameter during the function execution does not affect the actual parameter that is passed at the function call). If this keyword is not specified, the function parameter is passed as a reference (changing the formal parameter value inside the function changes the actual parameter value).

<Param 1>, ..., <Param N>

Optional comma-separated list of formal parameters. Formal parameter values must match the values of actual parameters that are passed to the function. This list contains the parameter names that are used in the function source code. The list of formal parameters can be empty.

=<DefValue>

An optional default parameter value. You can include parameters with default values anywhere in the list of formal parameters. For more information, see Passing procedure and function parameters.

Export

An optional keyword that specifies that this function is accessible from other modules.

// Local variables declaration

Declarations of local variables that can be referenced in this function only (see `Var` operator description).

// Statements

Executable statements of the function.

Return <ReturnValue>

The keyword that terminates the function and returns the specified value to the expression from which the function is called.
The return value can be an expression or a variable, the value of which contains the result of the function call.

EndFunction

A keyword to end the source code of a function.

Example:

```
Var Glob;

// The function definition
Function MyFunction(Par1, Par2, Par3) Export

  Glob = Glob + Par1 + Par2 + Par3;
  Return Glob;

EndFunction

Glob = 123;
Res = MyFunction(5, 6, 7); // The function call
```

4.7. Working with queries

4.7.1. Using object properties

In addition to the basic reference through a dot, 1C:Enterprise language provides a mechanism to access the properties of objects specifying string with the property name using the operator `[]` (square brackets).

Object property ([])

Description:

This structure allows to access the properties of objects in the same way as through a dot and the property name.

Syntax:

`<Object>[<Property name>]`

Parameters:

`<Object>`

The object which property is accessed.

`<Property name>`

`String`. The name of the property to access.

Example:

```
Dir = Directories.Products.FindByCode(SearchCode);

// Calling the name of the directory using the sting with the property
name
A = Dir["Name"];

// Calling the name of the directory using the property name
A = Dir.Name;

// Both of these property references are absolutely equivalent
```

4.7.2. Extending the context of objects and forms

The language allows to access objects and forms from the outside, from other program modules, by referring their variables, procedures and functions, in the same way as accessing the properties and methods of the objects and forms themselves. You can refer to variables, procedures, and functions that are declared with the `Export` keyword. In addition you can refer to the form attributes.

Example:

```
// An example of using the procedure for printing documents from the
journal
// of documents. Suppose we have several different documents
// and every document has a procedure Print(). The module of the
// document journal contains the Print button which
// calls the print procedure for current document of the journal.
Procedure PressPrint(Element)

  // Get the current document where the cursor is positioned.
  CurrDoc = Controls.JournalList.CurrentRow;

  // Get the main form of the current document.
  FormCurrDoc = CurrDoc.GetForm();

  // Call the print procedure from
  // the document form module.
  FormCurrDoc.Print();

EndProcedure
```

4.7.3. Passing procedure and function parameters

Parameters can be passed to procedures and functions in two ways. One method is called passing by **reference**. Not specific parameter value, but the memory address (variable reference) where this value is located is passed. Changing a passed value in a called procedure or function changes the passed variable in the calling method.

Another method is called passing by **value**. It is the transfer of a copy of the parameter value. In this case, changing the passed value in the called procedure or function does not change the value of the passed variable in the calling method.

In addition the mechanism for transfer the parameters of procedures and functions depends on the kind of call performed:

- a call without transfer of control between the client and the server (only on the client or only on the server);
- a call with transfer of control between client and server.

Let us consider each call method in details.

4.7.3.1. Call without transfer control from client to server

If the call occurs without transfer control between the client and the server (the call occurs only on the client or only on the server side), then by default the parameters are passed by reference. In this case, a change in the formal parameter changes in the corresponding actual parameter. Using the `Val` modifier before

the name of a formal parameter, you can specify that the parameter is passed by value. In this case, you cannot change the value of the actual parameter by assigning a value to the formal parameter.

```
&OnClient
Procedure MyProcedure()

  A = 100;
  ByReference(A);
  // Variable A = 40, since in the procedure body the value
  // of the parameter Parameter1 is changed to 40.
  // Variable A is changed because the parameter is passed by
reference.

  A = 100;
  ByValue(A);
  // Variable A = 100, despite the fact that in the procedure body
  // the value of the parameter Parameter1 is changed to 40.
  // Variable A is not changed because the parameter is passed by
value.

EndProcedure

&OnClient
Procedure ByReference()

  Parameter1 = 40;

EndProcedure

&OnClient
Procedure ByValue(Val Parameter1)

  Parameter1 = 40;

EndProcedure
```

However, you should consider the following feature: if a parameter passes an aggregate object, then it is impossible to assign a another value to the actual parameter, but it is possible to change the passed object itself. For example, if an array is passed to a procedure by value, then you can clear this array using the `Clear()` method, but the parameter value itself cannot be changed in the calling procedure.

```
&OnClient
Procedure MyProcedure()

  Array = New Array;
  Array.Add(12);
  Array.Add(18);

  // The array contains two elements
  ByValue(Array);

  // The array is empty, but this is still an array, not a Number
EndProcedure

// The parameter is passed by value
&OnClient
Procedure ByValue(Val Parameter)

  // The array contains two elements
  Parameter.Clear();

  // The array contains no values!
  // Change the formal parameter
  Parameter = 14;

  // The value of only the formal parameter is changed
EndProcedure
```

You should also consider how the variables are stored when performing calls:

```
&OnClient
Procedure TestProcedure()

  ValueA = New Array;
  ValueA.Add(1);
  ValueA.Add(2);
  ValueB = ValueA;

  // Using any variable: ValueA or ValueB results to
  // change of the same array

  MyProcedure(ValueA, ValueB);

  // After the procedure call both variables (ValueA and ValueB)
  // refer to the same array.
  // The array contains the following values:
  // ValueA[0] = ValueB[0] = "A"
  // ValueA[1] = ValueB[1] = "B"

EndProcedure

&OnClient
Procedure MyProcedure(Parameter1, Parameter2)

  // Though there are two parameters, actually the code operates with
one array

  Parameter1[0] = 5;
  Parameter1[1] = 6;

  Parameter2[0] = "A";
  Parameter2[1] = "B";

EndProcedure
```

In this example, please note the fact that the assignment operation `ValueB = ValueA;` (in the procedure `TestProcedure()`) does not create a copy of the array stored in the variable `ValueA`. Both variables refer to **the same** array! At the same time, in the `MyProcedure()` procedure the parameters are passed by reference, therefore, changing two different formal parameters changes the same physical array.

4.7.3.2. Call with transfer control from client to server

The feature of the call of procedures and functions with the transfer of control between the client and the server is the fact that, in the general case, the call is performed on the computer differ from the computer where the method is called. The reason is the fact that (in general) the client is running on one computer, and the server -is running on another. Therefore, it is impossible to organize the pass of parameters by reference, since one computer cannot directly access the memory of another computer. When using the file-server option, the client and

the server run on one computer, but this does not change the interaction logic. For this reason, the pass of parameters in the case of client-server interaction is performed in a special way:

- When transfer control from the client to the server (and back), **always copies** of the parameters are passed. When a server procedure or function is called from the client, a copy of the actual parameter is created and this copy is passed to the server side. When control is returned from the server to the client, a copy of the formal parameter (which is changed in the called procedure or function) is also created to pass back to the client.
- If the formal parameter is specified with the `Val` modifier, then the parameter value is passed only when the procedure or function is called and it is not passed back when the control is returned to the client.
- If the same actual value is specified for several formal parameters, then the number of copies of the actual value created is equal to the number of the formal parameters where the value is used.
- If the value of the same variable is specified for several formal parameters, then after returning control from the server, the value of this variable is set to the rightmost formal parameter (without the `Val` modifier), which is changed in the called function.
- If an exception occurs during the execution of the server method, the values of the method parameters passed by reference are set undefined when return to the client side.

```
&OnClient
Procedure TestProcedure(Command)

  ValueA = New Array;
  ValueA.Add(1);
  ValueA.Add(2);
  ValueB = ValueA;

  // Using any variable: ValueA or ValueB results to
  // change of the same array

  // At the procedure call a copy of the array is created for each
formal parameter
  ServerProcedure(ValueA, ValueB);

  // After the procedure call:
  // 1. Variables ValueA and ValueB refer to different arrays
  // 2. The arrays contain different values.
  // ValueA[0] = 5
  // ValueA[1] = 6
  // ValueB[0] = "A"
  // ValueB[1] = "B"

EndProcedure

&OnServer
Procedure ServerProcedure(Parameter1, Parameter2)

  // Since the copies are passed, for each formal parameter on the
server an "own" array is created

  Parameter1[0] = 5;
  Parameter1[1] = 6;

  Parameter2[0] = "A";
  Parameter2[1] = "B";

  // The server returns copies of the variables Parameter1 and
Parameter2

EndProcedure
```

You should always consider that when passing parameters to the server, a copy of the value is created. This allows to avoid "hard to find" errors when operating the system. Consider an example:

```
&OnClient
Procedure TestProcedure(Command)

  ValueA = New Array;
  ValueA.Add("1");
  ValueA.Add("2");
  ValueB = ValueA;
  Structure = New Structure("Key1, Key2", ValueA, ValueA);
  ServerProcedure(Structure);
  ValueA[0] = 9;
  ValueA[1] = 8;
  ValueB[0] = "C";
  ValueB[1] = "D";

EndProcedure

&OnServer
Procedure ServerProcedure(Parameter1)

 Parameter1.Key1[0] = "0";
 Parameter1.Key1[1] = "1";
 Parameter1.Key2[0] = "2";
 Parameter1.Key2[1] = "3";

EndProcedure
```

After the call of the `TestProcedure()` procedure, the state of the variables is as follows:

- The arrays `ValueA` and `ValueB` are identical and contain the values of "`C`" in the first element and the value of "`D`" in the second element, since the variable `ValueB` refers to array of the variable `ValueA` and the values of these variables are not passed to the server.
- The variable `Structure` contains two arrays:
  - With the values "`0`" and "`1`" for the elements of the array stored in the structure element with the key `Key1`.
  - With the values "`2`" and "`3`" for the elements of the array stored in the structure element with the key `Key2`.
  - The reason for this behavior is that on a procedure call not only the parameter itself (the `Structure` variable) is copied, but also all the objects stored in this structure are copied: two arrays that initially refer to the same array with the values of the elements "`1`" and "`2`".

4.7.3.3. General features of specifying parameters

When passing parameters to a method (procedure or function), two situations may occur: when the parameter is **omitted** and when the parameter is **skipped**. The skipped parameter -is a parameter which value is not explicitly specified. If several parameters are skipped, then in the string with the method call each of these skipped parameters must be designated with "its own" comma.

```
Procedure MyProcedure(Parameter1, Parameter2)
…
EndProcedure

// Parameter Parameter1 is skipped
MyProcedure(,1)
```

You can omit only one or several parameters specified on the right in the list of formal parameters in the method definition. Only parameters with default value specified can be omitted. If you omit the parameters without default values, then compilation error for the parameter number mismatch is generated.
The value of the omitted parameter(s) is determined by the default values of the omitted parameter(s). The value of the skipped parameter is equal to the default value (if specified), or to the `Undefined` value, if no default value is specified for this formal parameter.

```
Procedure MyProcedure(Parameter1, Parameter2 = "Default", Parameter3 =
16)
…
EndProcedure

// Parameter Parameter1 is skipped, the parameter value = Undefined
// Parameter Parameter2 is specified, the parameter value = 1
MyProcedure(,1)

// Parameter Parameter1 is skipped, the parameter value = Undefined
// Parameter Parameter2 is skipped, the parameter value = "Default"
// Parameter Parameter3 is omitted, the parameter value = 16
MyProcedure(, ,);

// Error of the parameters number mismatch
MyProcedure();
```

If a method is called without passed parameters (the list of parameters is empty), then the parentheses are mandatory anyway.
When passing string values from client to the server and back you should consider that these values can not contain characters that are not valid in the XML specification version 1.0 (http://www.w3.org/TR/xml/).

TIP. On the server side, such a check can be performed using the `FindDisallowedXMLCharacters()` function.

In constructors of various objects such optional parameters are processed in a special way. If the constructor definition specifies that the parameter is optional, this means that this parameter can be omitted, but this parameter can't be skipped in all cases. The omitted parameter always has the default value, and the skipped parameter has always the `Undefined` value. In this case, the behavior of the constructor depends on its implementation. Some constructors may take the value `Undefined` as an instruction to use some default value, but some of them- take this value as is (with the possible generation of a run-time error). When using optional parameters follow the recomendations below:

- It is not recommended to skip optional parameters.
- If the constructor has optional parameters, -it is recommended to omit all optional parameters, if the parameters don't need explicit specification. Otherwise, it is recommended to explicitly specify all optional parameters preceding the parameter that you want to specify explicitly.

4.7.4. Serialization / deserialization

Serialization -is the process to convert an object into a stream of bytes for storage or transfer to memory, to a database, or to a file. This operation is intended to save the state of the object for later restore if necessary. The reverse process is called **deserialization**. 1C:Enterprise provides several different types of serialization/deserialization: standard serialization, serialization into XML and XDTO serialization. XDTO serialization can generate XML or JSON format. The platform always uses XDTO serialization into XML format for internal needs. Each of these serializations is used for different purposes.
Some objects in 1C:Enterprise don't have implementation for specific types of serialization or can't be serialized at all. Syntax Assistant shows information about types of serialization implemented for given object. If Syntax Assistant does not show information about the available types of serialization, -it means that this object does not support serialization at all.
Let us consider how serialization/deserialization is used. For example, on a server call (and return to the client application side), the process is as follows:

- all parameters of the server method call are serialized;
- the method is called with simultaneous transfer of the serialized presentation of the parameters to the server side;
- the server deserialize the passed parameters;
- the server call itself is executed;
- all return parameters and the result of the call (if any) are serialized;
- the control is returned to the side of the client application with the simultaneous transfer of the serialized data;
- on the client side the call results are deserialized;
- the call is completed.

From this description of the call process one very important feature immediately follows: cyclic object references in server call parameters prevent to make a call because the serialization can't be performed.
Another feature of the serialization/deserialization process is: if the serialized object refers to the same instance of any data from different properties, then after the object is deserialized, the following occurs:

- The object that was assigned to different properties of the serialized object- is no longer associated with the properties of the deserialized object.
- Properties that previously referenced the same object, after deserialization refer to two different objects.

Let us give an example:

```
&OnServerWithoutContext
Procedure TestOnServer(MethodParameter)
  MethodParameter.Property1.Property1 = 5;
  MethodParameter.Property1.Property2 = 6;
  MethodParameter.Property2.Property1 = 7;
  MethodParameter.Property2.Property2 = 8;
EndProcedure

&OnClient
Procedure Test(Command)
  NestedObject = New Structure("Property1, Property2", 1, 2);
  MainObject = New Structure;
  MainObject.Insert("Property1", NestedObject);
  MainObject.Insert("Property2", NestedObject);
  TestOnServer(MainObject);
EndProcedure
```

In this example, `NestedObject` -is the object that will be “multiplicated” after serialization/deserialization. Before call the `TestOnServer()` procedure the properties `MainObject.Property1` and `MainObject.Property2` contain a reference to the same structure `NestedObject`. Moreover, changing any property of the structure `NestedObject` automatically results in that in accessing through two points the same values are fetched (`MainObject.Property1.Property1` and `MainObject.Property2.Property1`).
If you examine the state of the `MainObject` structure after the call, you can see that all 4 properties of the `MainObject` structure are filled with different values. Moreover, after the server method call an attempt to change the structure `NestedObject` does not change the state of the structure `MainObject`.
Similar information is given in the section describing the features of the pass the method parameters on client-server call.

4.7.5. Using value collections

Some objects in 1C:Enterprise language are collections of values. Many collections have a set of similar methods and properties, such as `Number()`, `Index()`, `Add()`, `Remove()`, etc. The properties of the collection, as a rule, are its elements. Collections allow to iterate their elements using loop `For each` - `In` - `Do`. Many collections allow to access their elements using the operator `[<Argument>]` (square brackets). As a rule, the index of the collection element is passed as the argument. Index of collection elements starts from 0. This means that the index of the last element is equal to the number of elements in the collection minus 1.
If during the collection iteration removing or other changes in the composition of the elements occur, then the further behavior of the system is not defined.

For a detailed description of specific collections, their properties and methods, and their use, see the descriptions of specific objects.

4.7.6. Using numbers and indexes

In the 1C:Enterprise language some objects have components that are numbered. Examples of such objects include a string which characters have numbers, a spreadsheet document which rows and columns numbers, etc. Such components of objects usually are referred by **Number**. Numbers start from 1.
Collection elements are referred by **Index**. Index of collection elements starts from 0.

4.7.7. Using system enumerations

1C:Enterprise language use a concept of system enumerations. They allow to define some limited set of predefined values. The system enumerations are referred as properties of the global context using its name. Specific values are specified through a dot after the name of the system enumeration. System enumerations are used typically to set values of parameters of system methods or properties of objects, as well as return values of methods.

4.7.8. Using predefined values

4.7.8.1. Using object manager

You can get a predefined value on the 1C:Enterprise server side using the manager of the corresponding object. The string defining the necessary attribute is as follows:

```
PredefinedValueType.MetadataObjectName.Value
```

Let us consider the components of this string in more detail:

- `PredefinedValueType` - to obtain predefined values you can specify the following data types (in plural form):
  - `Directories`,
  - `ChartsOfCharacteristicTypes`,
  - `ChartsOfAccounts`,
  - `ChartsOfCalculationTypes`,
  - `Enumerations`.
- `MetadataObjectName` - specifies the name of the metadata object as specified in Designer.
- `Value` - may be one of the following:
  - for enumerations the name of the enumeration value is specified;
  - to obtain a predefined value its name is specified as it is defined in Designer;

- `RoutePoints.PointName` - the route point of the business process.

If you need to get a route point of a business process, the string describing the resulting value should has the form:

```
BusinessProcesses.MetadataObjectName.RoutePoints.RoutePointName
```

Example:

```
// Obtain enumeration value.
Type = Enumerations.ProductTypes.Product;

// Get the predefined data of the directory.
Element = Directories.Currency.Rouble;

// Route point of the business process
Point = BusinessProcess.Adjustment.RoutePoints.Approval;
```

4.7.8.2. Using the PredefinedValue() function

Because application objects are not available on the client side, it is impossible to obtain predefined details using object managers. The global context method `PredefinedValue()` is provided to obtain them. The parameter of this method is a string describing what predefined value to get. The syntax to describe the predefined value is the same as the syntax for the operator `Value` of the query language.

The string defining the necessary attribute is as follows:

```
PredefinedValueType.MetadataObjectName.Value
```

Let us consider the components of this string in more detail:

- `PredefinedValueType` - to obtain predefined values you can specify the following data types (in singular form):
  - `Catalog`,
  - `ChartOfCharacteristicTypes`,
  - `ChartOfAccounts`,
  - `ChartOfCalculationTypes`,
  - `Enumeration`,
  - `BusinessProcess`.
- `MetadataObjectName` - specifies the name of the metadata object as specified in Designer.
- `Value` - may be one of the following:
  - for enumerations the name of the enumeration value is specified;
  - to obtain a predefined value its name is specified as it is defined in Designer;
  - `RoutePoint.PointName` - the route point of the business process.
  - `EmptyRef` - to get a blank reference.

To get the value of the system enumeration the method parameter should have the following form: `SystemEnumerationName.SystemEnumerationValue`.

Example:

```
ChartType = PredefinedValue("ChartType.ConcaveSurface");
```

If you need to get a route point of a business process, the string describing the resulting value should has the form:

Example:

```
// Obtain enumeration value.
Type = PredefinedValue("Enumeration.ProductTypes.Product");

// Get the value of the blank reference.
EmptyRef =
  PredefinedValue("Document.Invoice.EmptyRef");

// Get the predefined data of the directory.
Element = PredefinedValue("Directory.Currency.Rouble");

// Route point of the business process
Point =  PredefinedValue("BusinessProcess.Adjustment.
RoutePoint.Approval");
```

4.7.9. Synchronous and asynchronous operations

4.7.9.1. General information

Operating with objects that implement some specific functionality (files operations, blocking windows, cryptography and external components) you can use two different approaches: **synchronous** and **asynchronous**. The difference in these approaches is shown by their names.

In the synchronous approach, the call of any object method stops the execution of the calling code until the call of the object method is completed (with all nested calls). A standard procedure or function call in 1C:Enterprise language is a synchronous call.

In the asynchronous approach the object method is called as usual, but control returns immediately to the calling code and the execution of this code continues. The called method guarantees that after the completion of its operation the method notifies the necessary component about the completion (if such notification is required). The notification is performed by call a special method that is passed to the called object method. All of the above is true only for methods for which support for asynchronous operation is stated.

1C:Enterprise allows to use asynchronous operation for the following components:

1. Blocking windows;
2. File system extension;
3. Crypto Extension;
4. Add-ins.

All options assume asynchronous operations only on the side of the client application. On the server side, synchronous operations are available, with the exception of blocking windows (because they absent on the server side).

In terms of the names of methods of 1C:Enterprise language, the asynchronous methods differ from synchronous methods by their names which (as a rule) begins with the words `Start` or `Show`. This knowledge simplifies the search for analogues of synchronous methods.

Let us consider the organization of an asynchronous call on the example of displaying a warning. For asynchronous warnings you can use the method `ShowMessageBox()`. In order to notify that the warning dialog box is closed, the **warning description** is passed to this method. The warning description contains the name and location of the procedure that is to be called by the system after the dialog box is closed. Accordingly, the method `ShowMessageBox()` differs from the method `Warning()` in that the method `ShowMessageBox()` has another parameter through which the description of the warning is passed. This difference means that the method supports asynchronous operation.

Example:

```
&OnClient
Procedure ExecuteCommand(Command)

  CallBack = New NotifyDescription("CompletitionWarning", ThisObject);
  ShowMessageBox(CallBack, "Form closing is processed separately");

EndProcedure

&OnClient
Procedure CompletitionWarning(AdditionalParameters) Export

  // some processing

EndProcedure
```

To describe the notification, you should use the special `NotifyDescription` object. Creating the object, you should specify the name of the notification procedure, the location of the procedure, and additional parameters, if necessary. The additional parameters can be used by the notification handler for its processing. In the example above, the additional parameters are not specified.

In addition it should be noted that the procedures for handling notifications (callback) can be located only on the side of the client application in the form module, common module, and command module. The callback procedure should be declared with the `Export` keyword.

So, the above example shows that using the asynchronous methods significantly changes the approaches to development. The main change is that a single piece of source code in 1C:Enterprise language (for synchronous methods) is divided into several isolated fragments (when using asynchronous methods).

Thus, if in the above example, after displaying the warning, some processing should be performed (just after the user reaction, not after calling the method!), then these actions should be moved to the `CompletitionWarning()`.

More complex algorithms obviously can lead to more complex refactoring of the source code. It is possible that algorithm itself will require refactoring, not only its implementation, which can become much more complicated. For example if in synchronous mode an algorithm requires some user action, then this action is obtained directly where this action is required. If this algorithm is refactored using asynchronous approach, it is reasonable to develop a separate dialog box where the user answer all questions, and then the result of this “survey” is transferred to the method that implements the required algorithm. In this variant the questions “in place” are not required already, but instead the results of the user's “survey” should be analyzed.

Asynchronous methods changes not only approaches to development, but the approach to error handling to some extent. For example, errors may occur while the 1C:Enterprise language code is not executed and the `Try ... Exception` construction cannot be used. To handle such situations in notification handler you can specify a procedure that the system will call when an error occurs. An example of such an error is an error occurring in the process of deleting files.

Example:

```
&OnClient
Procedure DeleteAllFilesInCatalog(PathToCatalog)
  CallBack = New NotifyDescription("DeleteCompletion", ThisObject,
"DeleteError", ThisObject);
  BeginDeletingFiles(CallBack, PathToDirectory,
GetClientAllFilesMask());
EndProcedure

&OnClient
Procedure DeleteCompletion(AdditionalParameters) Export
  Report("Delete successful");
EndProcedure

&OnClient
Procedure DeleteError(ErrorInfo, StandardProcessing,
AdditionalParameters) Export
  Report("Deletion error found: " + BriefErrorDescription(ErrorInfo));
EndProcedure
```

This implementation has one special feature: if an error occurs during the deletion of files, then along with the error message in the platform message window, the user will also see the system dialog box with an error message. If the system

dialog is not needed - then in the `DeleteError()` you should set the `StandardProcessing` parameter to `False`.
It should also be noted that the indication of the error handler in the notification description is ignored when using asynchronous methods for blocking windows.
Another special feature of using asynchronous calls is their debugging. Currently, the debugger is able to debug actions that are synchronously executed in the object of debugging (in fact, at the initiative of the debugger itself).
In the case of asynchronous calls, an asynchronous call is made at a time other than the actual execution of the line of 1C:Enterprise language code that initiates the call. Therefore, if an error occurs during the actual execution of the asynchronous call, the debugger cannot correctly determine the status of the debugged application. Therefore, the current execution line, call stack, and so on are not available.

4.7.9.2. Using web client

There are some features specific to the web client. These differences are due to the implementation features of web browsers (no support for modal windows and synchronous calls, and some features of the security model). In the web client the synchronous operations (in all options) are not supported. Operations with files and cryptography require special plug-ins installed in your web browser. In Google Chrome web browser before installing the platform plug-ins you need to install 1C:Enterprise extension for the web browser. It can be installed from the Chrome Web Store (if necessary, you will be automatically taken to the installation page). Without this extension installed, you can't use the platform plug-ins.
To use the extensions you need:

- Properly configure the web browser.
- File system extension:
  - Install the extension -using the `BeginInstallFileSystemExtension()` or `InstallFileSystemExtension()` method. This is manual action that must be performed once for each user of the local computer using the extension.
  - Enable the extension -using the `BeginAttachingFileSystemExtension()` or `AttachFileSystemExtension()` method.
- Cryptography extension:
  - Install the extension -using the `BeginInstallCryptoExtension()` or `InstallCryptoExtension()` method. This is manual action that must be performed once for each user of the local computer using the extension.

- Enable the extension -using the `BeginAttachingCryptoExtension()` or `AttachCryptoExtension()` method.
- External components.

4.8. Differences between various system startup options

The 1C:Enterprise system can be used in file version or client-server version, using external connection, as well as Internet services.
Designer allows you to customize the use of procedures and functions of common modules and object modules for each option.

4.8.1. Procedures and functions execution

To enable the use of procedures and functions of various modules you can use preprocessor commands and compilation directives.

4.8.1.1. Differences between preprocessor commands and compilation directives

The preprocessor commands and compilation directives are designed to leave in the compiled module only that content that is really needed in a particular context. The preprocessor commands act on the source code of the module (that is, remove unwanted code from the module), and compilation directives affect the structural units of the source code (methods - procedures or functions, as well as variable declarations).
All software modules in the system can be divided into two large groups of modules:

- Modules that can exist (and run) in only one context, for example, a managed application module can run only on the client side (using thin client or web client).
- Modules that can exist (and run) in multiple contexts. Such modules are the managed form module, the command module and the common module. For example, a managed form module can exist in four instances, and in the process of operation the control can be transferred between different instances of the module:
  - client context module,
  - server context module,
  - client out-of-context module,
  - server out-of-context module.

The command module and the common module cannot have context instances (as opposed to the form module). Therefore, some modules can be compiled several times, depending on the code fragments of the module that can be executed in a particular context.

Let us consider the influence of compilation directives and preprocessor commands on the source code of a module in the process of its conversion into “executable” code.

Consider this on the example of a managed form module.

At the time of form creation, four instances of the form module are created and all the necessary preprocessor commands for each module instance are determined. Then each module instance is processed and the code between the preprocessor commands is excluded.

Then, the resulting program code is compiled in accordance with the context and compilation directives. The result is the executable module.

Suppose, in the source code of a form module, there is following fragment:

```
&OnClient
Procedure FileOperations()

  #If WebClient Then

    // source code 1
    // runs only in web client
  #Else

    // source code 2
    // runs in other client types
  #EndIf

EndProcedure
```

Then the following occurs:

- This procedure will be compiled on any client (according to the compilation directive).
- At the same time, the preprocessor commands will determine which code on which client should be left in the module (and, as a result, be compiled). In this example, only “source code 1” will be available on the web client, and “source code 2” -will be available on other clients.

However, if you add to the form module the following code:

```
#If OnServer Then

&OnClient
Procedure ForClient()
EndProcedure

#EndIf
```

Then the following occurs:

- the source code of our procedure is present on the server, but it will not be compiled, because the compilation directive `&OnClient` prevents the appearance of the procedure on the server side;
- the client even doesn't have the source code of our procedure, since this preprocessor command “cuts out” the code. So this procedure will not be compiled and will not be available for a call.

The described mechanism allows to understand the interaction of the directives and the commands and to use them in combination.

It should be noted that the methods marked in the module with the directives `&OnClientOnServerWithoutContext` and `&OnClientOnServer` will be present simultaneously in different instances of the modules.

4.8.1.2. Preprocessor commands

Preprocessor commands use the following syntax:

Preprocessor command

```
#<Instruction>
```

Instruction

```
<Instruction> = { If <Logical expression> Then | OtherwiseIf <Logical expression> Then | Otherwise | EndIf | Insert | EndInsert | Delete | EndDelete }
```

Logical expression

```
<Logical expression> = [HE] <Preprocessor keyword> [<Boolean operation> [HE] <Preprocessor keyword> [<Boolean operation> [HE] <Preprocessor keyword>]…]
```

Preprocessor keyword

```
<Preprocessor keyword> = { Server | AtServer | Client | AtClient | ThinClient | MobileClient | WebClient | ExternalConnection | ThickClientManagedApplication | ThickClientOrdinaryApplication | MobileAppClient | MobileAppServer | Region | EndRegion }
```

Boolean operation

```
<Boolean operation> = {And | Or}
```

Russian	English
If	If
Then	Then
Else	Else
ElseIf	ElsIf
EndIf	EndIf
Area	Region
EndRegion	EndRegion

Russian	English
Insert	Insert
EndInsert	EndInsert
Delete	Delete
EndDelete	EndDelete
AND	And
Или	Or
Not	Not
Server	Server
AtServer	AtServer
Client	Client
AtClient	AtClient
ThinClient	ThinClient
MobileClient	MobileClient
WebClient	WebClient
ExternalConnection	ExternalConnection
ThickClientManagedApplication	ThickClientManagedApplication
ThickClientOrdinaryApplication	ThickClientOrdinaryApplicatio
MobileAppClient	MobileAppClient
MobileAppServer	MobileAppServer

Below list of preprocessor commands and their brief description are given:

- Server, AtServer - defines a server;
- Client, AtClient - defines ant client;
- ThinClient - defines thin client;
- MobileClient - defines mobile client;
- WebClient - defines web client;
- ExternalConnection - defines external connection;
- ThickClientManagedApplication - defines managed application mode for thick client;
- ThickClientOrdinaryApplication - defines ordinary mode for thick client;
- MobileAppClient - defines client part of mobile application;
- MobileAppServer - defines server part of mobile application;
- Region, EndRegion - are not intended to determine the location of code execution in 1C:Enterprise language. They are used to select a code fragment that can be collapsed (similar to procedures, functions, etc.).
- Insert, EndInsert, Delete, and EndDelete -are used only in the extended configurations with the annotation ChangeAndControl.

The following table shows which preprocessor commands are defined in various 1C:Enterprise operating modes:

Mode	Server, AtServer	Client, AtClient	ThinClient	MobileClient	WebClient	ThickClientManagedApplication	ThickClientOrdinaryApplication	ExternalConnection	MobileAppClient	MobileAppServer
Client/server mode										
On server side	+									
Thick client in ordinary mode		+					+			
Thick client in managed mode		+				+				
Thin client		+	+							
Mobile client		+		+						
Web client		+			+					
External connection								+		
File mode										
Thick client in ordinary mode	+	+					+			
Thick client in managed mode	+	+				+				

Mode	Server, AtServer	Client, AtClient	ThinClient	MobileClient	WebClient	ThickClientManagedApplication	ThickClientOrdinaryApplication	ExternalConnection	MobileAppClient	MobileAppServer
Thin client		+	+							
On server side of thick client	+									
Mobile client		+		+						
On server side of mobile client	+									
Web client	+				+					
External connection								+		
Mobile application client		+							+	
On server side of mobile application	+									+

If you use a non-global common module, which is set to be used on the client (any) and on the server, then the methods included in the condition `#If Server Then #EndIf` are available only if the call to these methods is performed from the server. You can't call such methods from the client side.

In the module of the managed form that is executed in the file mode in the thick client, the commands `Server` and `AtServer`, and `Client` and `AtClient` are different. Thus, the `Server` and `Client` directives are defined both in the

client and the server instances of the form module, while the `AtServer` directive is defined only in the server instances of the form module, and the `Client` directive -is defined only in the client instances of the form module.
In other cases, the `Server` and `AtServer` commands are completely identical and the `Client` and `AtClient` commands are also completely identical.

NOTE. Before transfer the module to a thin client or web client, the server processes the preprocessor commands in the module. During the processing, the code in 1C:Enterprise language, which is not executed on the side of the above clients, is replaced with spaces (actually deleted). At the same time, the characters of line breaks and tabs remain in their places.

4.8.1.3. Compilation directives

Compilation directives have the following syntax:

Compilation directive

```
&<Directive>
<Syntactic structure>
```

Directive

```
<Directive> = { AtClient | AtServer | AtServerNoContext
| AtClientAtServerNoContext | AtClientAtServer }
```

Syntactic structure

```
<Syntactic structure> = { <Variable declaration> |
<Procedure declaration> | <Function declaration> }
```

Russian	English
`AtClient`	`AtClient`
`AtServer`	`AtServer`
`AtServerNoContext`	`AtServerNoContext`
`AtClientAtServerNoContext`	`AtClientAtServerNoContext`
`AtClientAtServer`	`AtClientAtServer`

Below list of compilation directives and their brief description are given:

- `AtClient` - means that the method is executed on the client side in the context of the form.
  The variable exists for the all lifetime of the client part of the form.
  Client variables of the form module are visible from the method.
  Any method calls are allowed.
- `AtServer` - means that the method is executed on the server side in the context of the form.
  The variable exists only during a server call.
  Server variables of the form module are visible from the method.
  You can call the following methods:

- server methods;
- sever out-of-context methods;
- client-server out-of-context methods;
- methods of non-global server common modules.

- `AtServerNoContext` - means that the method is executed on the server outside the context of the form.
  This compilation directive can't precede variables.
  Variables of the form module are not visible from the method.
  You can call the following methods:
  - sever out-of-context methods;
  - client-server out-of-context methods;
  - methods of non-global server common modules.
- `AtClientAtServerNoContext` - means that the method is executed both on the client and on the server, outside the context of the form.
  This compilation directive can't precede variables.
  Variables of the form module are not visible from the method.
  You can call the following methods:
  - sever out-of-context methods;
  - client-server out-of-context methods;
  - methods of non-global server common modules;
  - non-global common module methods with Server and Client (managed application) flags.

  You can't call export procedures with this directive such as the methods of the object `ThisObject`.
- `AtClientAtServer` - means that the methods are executed on the client and on the server. This compilation directive can't precede variables.
  You can call the following methods:
  - sever out-of-context methods;
  - client-server out-of-context methods;
  - methods of non-global server common modules;
  - non-global common module methods with Server and Client (managed application) flags.

The table below shows which compilation directives are available in which modules of the 1C:Enterprise system:

	Form module	**Variables of form module**	**Command module**	**Common module**
`AtClient`	+	+	+	+
`AtServer`	+	+	+	+
`AtServerNoContext`	+			
`AtClientAtServerNoContext`	+			

	Form module	Variables of form module	Command module	Common module
AtClientAtServer			+	

4.8.2. Annotations

Annotation is a special form of syntactic metadata that is added to the source code to ensure operations of the configuration extension functionality. Annotations are used for code analysis and in runtime. You can annotate methods and variables in modules in 1C:Enterprise language. Annotations and compilation directives can be alternated.

Annotation

```
&<Before|After|Around>[([<Name>=<Value>[,
<Name>=<Value>, …]])]
```

The annotation begins with the “&” character (similar to compilation directives), followed by the name of the annotation (the valid annotation names are listed below), then if the annotation requires parameters its parameters follow separated by the character “,”. Annotation parameters are enclosed in parentheses. If an annotation has only one parameter this parameter value can be specified without its name. The following annotations are available:

Russian	English
Before	Before
After	After
Around	Around
ChangeAndValidate	ChangeAndValidate

Remember that the system does not support custom annotations. In other words, the developer cannot create his own applied annotation, which he can use on his own.

4.8.3. Specific features of using objects, their properties and methods

Each object, method or property of the 1C:Enterprise language (later in this section - object) has a certain visibility (see Syntax Assistant), which determines where an object, method, or property can be used. In addition, Syntax Assistant specifies some additional data that can help the developer.

Thin client - specifies that the object is visible in thin client.

Mobile client - specifies that the object is visible in mobile client.

Web client - specifies that the object is visible in web client.
Server - specifies that the object is visible on 1C:Enterprise server.
External connection - specifies that object is available in the external connection mode.
Thick client - specifies that the object is visible in thick client.
Mobile application (client) - specifies that the object is visible in the client part of 1C:Enterprise mobile version.
Mobile application (server)- specifies that the object is visible in the server side of 1C:Enterprise mobile version.

IMPORTANT! If for some object it is indicated that it is not visible for some startup mode, then the properties and methods of this object are also unavailable. Therefore, in descriptions of properties and methods it is not separately mentioned.
NOTE. If an object is not visible on the side of any client application, then the type of this object is also not available in this client application. For example, if the `CatalogObject.Counterparties` object is not available in the thin client, then the `Type("CatalogObject.Counterparties")` is also not available in the thin client.

Serialized. Indicates the ability to save the values of objects (for example, using the methods `SaveValue()` and `ValueToFile()`), to save the parameters of forms of reports and data processors, as well as to store in the `ValueStorage` and in temporary storage.
This object can be serialized to/from XML. Indicates the possibility to read/write data values of the 1C:Enterprise system to/from XML.
Server exchange is possible. Indicates that exchange of values of this type between the client and the server is possible.

NOTE. In managed startup mode, the exchange between the client and the server is possible only for objects that support XDTO serialization.

This object can be serialized to/from XDTO. Indicates that this type can be mapped to the XDTO data model. You need to specify the qualified name of the type(s) (indicated by the URI namespace and the name of the type) to which this type is mapped. For example, for the type `ValueStorage`: {http://v8.1c.ru/8/data/core}ValueStorage.
Syntax Assistant contains information about the caching of the results of some 1C:Enterprise methods calls. For example, for the global context method `PredefinedValue()` the following information is displayed: The result of the execution is cached at the first call before changing the configuration or version of the platform..
In some cases, the system restricts possible actions in event handlers. In these cases, Syntax Assistant specifies the prohibited action or describes the action, which may lead to an invalid action:

1. In the handler for this event, you cannot use the server methods of the form with the compilation directive &AtServer.
2. Changing the property on the client may require the server call.
3. The method call performs the server call as well.. This warning is placed in the description of client methods. It means that the developer cannot call this method in the handler of event for which context server calls are prohibited.

During the application runtime, it is possible to enable diagnostics of a violation of the above restrictions. If the `/EnableCheckServerCalls` key is specified in the client application startup command line, then detection of a violation of the specified rules displays diagnostics in message window. The same diagnostics is shown in the Information for technical support dialog box.

When launching the client application from Designer, this key can be enabled in Designer settings using the check box Display warnings upon server calls in event handlers.

Chapter 5. Configuration objects

This chapter describes using the configuration as a whole and the modes and mechanisms used for all configuration objects.
The creating and configuring basic configuration objects (constants, directories, documents, sequences, journals, enumerations, reports, data processors, registers), and some objects located in the General configuration branch (filter criteria, styles) are described in the documentation on the example of using the object edit window. Similar operations can be performed in the object property palette.

5.1. Configuration properties

The configuration has editable properties. The properties palette opens for the root branch of the configuration tree.
In addition to the basic properties of each configuration object, the configuration has the following properties described below.

5.1.1. Specifying general properties

Default run mode - specifies the default system startup mode (Managed Application or Ordinary Application). The new configuration is set to the Managed application startup mode. Also, the startup mode can be changed for the system user. This property cannot be changed if the Compatibility mode property is set to Version 8.1.
1C:Enterprise language variant - chooses the default programming language (Russian or English). This choice determines what language is used for the syntax structures in modules (for example, when using Syntax Assistant), for property names for objects created by the platform operations, as well as for the names of components (elements, commands, attributes, parameters) of forms created by the platform (both in the 1C:Enterprise mode and in Designer). Regardless of the value of the property, you can use both Russian and English versions of the language structures. When changing the value of the property, the option to type the language structures does not change.
Default roles - are the list of roles to use when the list of application users is empty. In this case, access authorization is not performed at the start of the system operation. The access rights are determined by the set of roles specified in this property. In this case, any user have administrative rights regardless of the

value of Administration right for all roles that are listed as default. If no configuration roles are specified here and the user list is empty, no access restrictions are applied to a user. Roles are specified in the Common -Roles branch of the configuration tree.

The default startup mode specified in the application must comply with the rights for valid applications specified for all roles that are selected in the Default roles property. So, if the Default run mode property is set to Managed application, and the set of primary roles does not provide the Thin client right, the user can't use the application by means of the thin client.

Managed application module - The Open link opens the managed application module editor window.

Session module - The Open link opens the session module editor window.

External connection module - The Open link opens the external connection module editor window.

Use managed forms in an ordinary application - indicates the need to use managed forms in the ordinary thick client mode. When this check box is selected, the rules for selecting a form by a thick client change, and the rules for centralized configuration verification also change.

NOTE. This property is only accessible when the configuration editing mode is set to Managed application and ordinary application.

Use ordinary forms in an ordinary application - indicates the need to use ordinary forms in the managed thick client mode. When this check box is selected, the rules for selecting a form by a thick client change, and the rules for centralized configuration verification also change.

NOTE. This property is only accessible when the configuration editing mode is set to Managed application and ordinary application.

Additional full-text search dictionaries - select common layouts or constants that will act as additional full-text search dictionaries.

Reports user settings storage - this storage stores user settings for reports.

Report option storage - this storage stores report options.

Form data settings storage - this storage stores form data. This storage can be used, for example, to save data processing attributes. At the same time it is possible to select an individual storage for each report and data processor.

User-defined dynamic list settings storage - Stores dynamic list settings.

Common settings storage - this storage stores various application settings. The platform does not write any data to this storage automatically. The developer has to implement the storage specified in the property using 1C:Enterprise language to save/restore applied user settings.

5.1.2. "Presentation" category

5.1.2.1. The interface parameters

Command interface - The Open link opens an editor to set the default visibility of the subsystems on the home page (including with a breakdown by roles).

Home page work area - The Open link opens the settings form to specify the forms to be located on the home page and the layout to form the work area.

Main section command interface - The Open link opens the dialog box for settings of the command interface of the main section.

Main section picture - this property changes the picture of the main section of the section panel.

Client application interface - allows to customize the default panel layout for the Taxi interface. The interface takes this form when the user clicks the Default button in the panel editor.

Main client application window mode - allows to specify the mode to open the main window of the client application:

- Normal - In this case, the application operates in the interface mode determined by the set of client application interface settings (configuration property Configuration compatibility mode and the Appearance parameter in the client application settings).
- Workplace - this mode is only used when running Taxi interface. Application starts operating in workplace mode.

  This mode has the following distinctive features:
  - The system command bar hides Service and Settings and all elements, except the client application logo, window title, hyperlink with the user name and About, Collapse, Expand, and Close buttons.
  - All panels, except the messages panel and the performance indicators panel, are hidden.
  - Form navigation arrows, the Home page title, the Home button, and the Favorites button are hidden.
  - Context menu accesible by right-clicking the form header is blocked.
  - If the properties of the form Show title and Show close button are set to `False`, the entire area with the title containing the form close button is be hidden.
  - The following operations (and their corresponding key combinations) become unavailable:
    - Panel navigation (Alt+1, Alt+2, Alt+3, Alt+`, F10),
    - Form navigation (Ctrl+Tab, Ctrl+Shift+Tab, Ctrl+F6, Ctrl+Shift+F6),
    - Show/hide all panels (Ctrl + Shift + `)

    - Create new document (Ctrl + N)
    - Open document (Ctrl + O)
    - Save document (Ctrl + S),
    - Print document (Ctrl + P)
    - Print document with current settings (Ctrl + Shift + P)
    - Open calculator (Ctrl + F2)
    - Follow the link (Shift + F11),
    - Retreive the link (Ctrl + F11),
    - Add to favourites (Ctrl + D),
    - Open the history panel (Ctrl + Shift + H),
    - Open the Favourites panel (Ctrl + Shift + B),
    - Open the Search panel (Ctrl + Shift + F),
    - Open the configuratin help (F1, Shift + F1, Alt + Shift + F1, Alt + F1, Ctrl + Shift + F1), at the same time forms help is still accessible,
    - operations with numbers in the clipboard (Shift + Num *, Shift + Num +, Shift + Num -).
- Full-screen workplace - this mode is only used when running Taxi interface. Application starts operating in full-screen workplace mode.

  This mode (in addition to features of Workplace mode) has the following features:
  - Application windows opens in full-screen mode. Exceptions are the cases of usage of such browser for web client running, which does not support full-screen mode.
  - In the web client, if your browser supports full-screen mode, a confirmation dialog box is displayed before the full-screen mode is enabled.
  - Press Esc to exit the full-screen view while in the web client.
  - In thin and thick client application, the Collapse , Expand, and Close buttons are hidden in the main application window. The Collapse button is hidden in the auxiliary windows.
  - Context menu accesible by right-clicking the system command bar is blocked.
- Kiosk - this mode is only applicable when running Taxi interface. Application starts operating in kiosk mode.

  This mode has all features of Full-screen workplace mode and the hyperlink with username and the About button are hidden from the system command bar.
- Embedded workplace - this mode can be used, only if a web client is embedded in an external web site. When this mode is specified for any

other client application -, it is ignored and client application main window is opened in Ordinary mode.

This mode has the following distinctive features:

- All features of Workplace mode.
- The system command bar is hidden.
- RUNS 1C (POWERED BY 1C) is displayed in the bottom right corner either in Russian or English. This message cannot be deleted.

You can select the way the application main window is opened in using any of the following three methods (they are listed in the ascending order of priority):

1. in the configuration properties;
2. using `MainWindowMode` command available in the command line when a client application is started;
3. using `SetMainWindowMode()` 1C:Enterprise language method.

5.1.2.2. Configuration parameters

Main style - specifies the style, used for configuration.

Main language - specifies the main configuration language.

Brief information - brief configuration description.

Detailed information - detailed configuration information (multiline text is permitted).

Logo - logo selection. The logo is selected using standard image selection window:

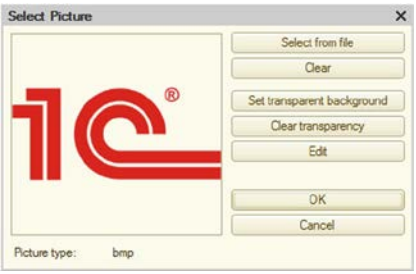


Fig. 54. Picture Selection

NOTE. The logo image is **64x64 pixels or less**. Image type may be any supported by 1C:Enterprise.

Splash - splash screen selection. The splash screen is selected using standard image selection window. The image used as splash screen must be **600x255** pixels. Transparency is not supported. Image type may be any supported by 1C:Enterprise.
In some cases it is possible to completely replace the splash screen displayed during 1C:Enterprise startup. Please address 1C Company to receive such option.
Copyright - configuration copyright notice.
Vendor information address - link to the configuration vendor information. It is specified in Copyright property. May be set with both schema prefix (http://) and without it.
Configuration information address - link to the configuration information. May be set with both schema prefix (http://) and without it. About window displays the following information: Configuration synonim, Configuration information address property, Copyright property, Configuration vendor information address property.
Default constants form - the default form for entering and editing the configuration constants. This form is selected from common forms in Common- Common forms configuration branch.

NOTE. This property is only accessible when the configuration editing mode is set to Managed application and ordinary application.

5.1.2.3. Main forms

Main report form, Main report settings form and Main report option form forms allow to specify the common forms to be used by the reports, which do not have the respective forms specified. For example, it is possible to create the common report form containing some features to be present in all the application reports, e.g., emailing the generated report. To solve this task create a common report form, implement the required commands in it and specify it in the Main report form configuration property. After that all the reports, which do not have the report form specified, will use the created common form.
Main dynamic list settings form property allows to specify the common form, using which the dynamic list settings will be edited in the application. If this property is empty - the automatically generated settings form will be used to edit the dynamic list settings.
Main data history form, Main data version history form and Main data version history differences form properties allow to specify the forms to be used when browsing the data history information. In addition to redefining the data history forms for complete configuration (using the above properties), it is also possible to selectively redefine the data history forms for the selected objects using `FormGetProcessing` event handler of the selected objects manager module.

The property Main form of selection of collaboration system users allows specifying the form to be used for the selection of users in the message, conversation or video conference of the collaboration system. This form will be used if the user clicks the hyperlink Show all in the list of users selection and the form is not overridden by the form selection handler (event `CollaborationSystemUserChoiceFormGetProcessing` of the form module or application module).

5.1.3. "Development" properties category

This properties category specifies the configuration vendor and version data.
Update directory address - contains the address of resource to be used for updating the application.

5.1.4. Properties related to development for mobile devices

Intended Use - refers to application intended use (Platform Application or Mobile Platform Application). The property is available only if the Default run mode property is set to Managed application. Depending on the selected use purpose, the capabilities of Designer change:

- PC application - development of an application that will run in thin, thick, web- and mobile client applications. The developed application will not run on a mobile platform.
- Mobile platform application - development of an application that will run on mobile devices. In this case the developer have access to a limited set of applied objects.
- Whenever both check boxes are checked (Platform Application and Mobile Platform Application) - you can develop applications containing all objects supported by 1C:Enterprise. However, the developed application may not run properly on a mobile version. The reason for this may be, for example, using configuration objects that are not supported by the mobile platform.

You should consider the following:

- If among the use purposes there is no PC application, then:
  - option to change the default run mode is unavailable;
  - option to change the mode of the main window of the client application is unavailable;
  - in the form editor a preview option can't be selected;

- the properties unavailable on the mobile platform are not available in the properties palette for configuration objects;
- only the types available on the mobile platform can be used as attribute types.

- If among the use purposes there is no Mobile platform application, then:
  - module verification uses the setting for the Mobile platform application mode; this mode, by default, is proposed for modification when editing parameters;
  - configuration verification uses separately stored settings of checks performed;
  - Syntax Assistant uses the separately stored settings of the imposed filters.

Required mobile application permissions - describes required permissions of the mobile operating system to run the mobile application. Please note that the composition of the installed permissions is determined by the functions of the 1C:Enterprise mobile version used by the developer. They should be set manually; they are not automatically granted by the analysis of the application code used.

The required permissions set in the configuration are used by the mobile application builder to set the permissions of the mobile application built. The actually set permissions depend on specific mobile operating system the application is built for. If the application uses any functionality requiring a permission for operation that is not specified in the configuration (and, as a result, in the built mobile application), then the behavior on specific device depends on the response of the mobile operating system to an attempt to use "unpermitted" functionality. Please note that the mobile version of 1C:Enterprise for the developer always has all the necessary permissions, regardless of what methods and objects used by the application. In this regard, the absence of set permissions in the configuration properties can be detected only after the ready mobile application is run on the mobile device.

When the mobile application tries to use any permission the mobile operating system will show this permission request dialog to the user. The text describing what the mobile application requires permission for will be shown in this dialog. This permission text may be specified by the configuration developer. The column Permission request message of the dialog Required mobile application permissions is intended for this purpose. If the configuration supports several languages -the message text can be standardized for each language.

Mobile client signature allows to manage the digital signature of the application used in the mobile client. Clicking on Open hyperlink opens dialog box to perform several operations:

1. Check the correspondence between the digital signature and the configuration status.

2. Create a private digital signature key for this configuration (and save it to the specified file).
3. Generate a digital signature for the configuration.
4. Clear the digital signature.

Information on the status of the digital signature is shown at the top of the dialog box.

If the configuration is not signed, the dialog box displays message Configuration is not signed. If the current configuration status does not match the digital signature, the label in the dialog box changes to The configuration signature has expired. In both cases, you should create or update the digital signature using the Create signature button. When you click this button, the system requests the path to the file with the digital signature private key. After the generation of the digital signature, the public key hash value is generated in the Public key hash field in the Base64 format for store in the `MobilePublicKey` parameter of the file with the shared infobase list (.v8i). If the configuration is not planned for use with a mobile client, you don't need to generate the digital signature.

Offline Configuration Content - defines content of offline configuration of a mobile client supporting standalone mode.

Limiting roles of the offline mobile application -define the roles restricting features available to a user operating a mobile device. This set of roles is used as follows:

- All roles assigned to a user are combined "by OR".
- All roles specified in this property are combined "by OR".
- The results obtained for a user and this property are combined "by AND".

Therefore, roles defined in this property are intended to restrict access rights available to a user operating a mobile device. For instance, if you need to restrict creation of `Goods` catalog elements, do the following:

- Create a role named `RestrictedOfflineApplication` (by way of example only).
- Grant all rights to a user with this role assigned. As such, the rights available to other roles will be reserved.
- Disable creation of new elements for `Goods` catalog. Thus, a user is no more able to create further goods on their mobile device.
- Select in a list of roles of Limiting roles of the offline mobile application property `RestrictedOfflineApplication` only.

For reference, note that `RestrictedOfflineApplication` can be used to disable all unwanted access rights on a mobile device without modifying the content of application primary roles.

5.1.5. "Help" properties category

Include in help contents - is the property is specified, the help contents will be included in the general configuration description.
Help information - Open link opens the configuration description editing window.

5.1.6. "Compatibility" properties category

Data lock control mode - option of data lock management in transaction.
Objects automatic numbering mode - determines, whether automatically retrieved object numbers are reused, if these have not been recorded to the database.
`AutoFree` value of this property is used to ensure the numbering mode similar to 1C:Enterprise 8.0. Numbers and codes retrieved automatically will be further used, if the object, which these have been retrieved for, has not been recorded.
`NotAutoFree` value of this property is used to select the numbering mode, when the numbers for the objects requiring continuous numbering are be assigned the numbers when being recorded and when the form is opened.
If the object (document, catalogue item, chart of characteristic types, business process or task) number has been allocated in transaction, it will be reused even in case if the `NotAutoFree` automatic numbering mode is enabled, including if the object has not been recorded during transaction.
`NotAutoFree` automatic numbering mode is only valid for the numbers, assigned outside the transaction. I.e. if the number has been assigned outside the transaction and the code is not used (the object has not been recorded), in `NotAutoFree` mode the number will not be used, and in `AutoFree` mode the number will be reused.
Modality usage mode - specifies if it is possible to use the methods, causing the modal windows to open, in the application. If the property is set to Use - it is possible to use modal windows without any restrictions. If the property is set to Do not use - it is not possible to use the modal windows in this application. The error is returned when attempting to use the methods (including during syntax check in the 1C:Enterprise language). Blocking windows must be used instead of the modal windows. If the property has Use with warning value, errors will not be returned when using modal windows, but a message will be output to message window, that modal windows can not be used in this mode. When using this mode modal windows must be replaced with blocking.
Stating with 1C:Enterprise version 8.3.5 behaviour of thin and thick client applications slightly changes. The behaviour described above (for thick and thin client applications) will be observed only if `/EnableCheckModal` 1C:Enterprise

startup command line parameter is used. If this parameter is not specified, the modal methods remain operable regardless of the Modality usage mode configuration property. In particular, such behaviour allows using external data processors, written without considering Modality usage mode property, in configurations, written with a glance to setting this property to Do not use.
If 1C:Enterprise is started from the Designer, `/EnableCheckModal` startup command line parameter is set mandatory without possibility to configure such behaviour.
Synchronous platform extension and add-ins call use mode - property controls the feature to use the synchronous calls of file system extensions, cryptography and add-ins. If the property is set to Use, the synchronous methods for extensions and add-ins will be available on client side. If the property is set to Do not use, synchronous methods become unavailable in Syntax Assistant, context help when editing modules and the extended configuration check reports errors if client-side synchronous calls are detected. It is recommended to use asynchronous methods in this case. Synchronous methods that perform operations with extensions and add-ins generate exceptions in 1C:Enterprise mode. Use with warnings mode is an intermediate option, where the use of synchronous methods is allowed, however, each such method generates a warning both in 1C:Enterprise mode and during the extended configuration check. Thick client generates actual value of this configuration property. When using thick client all the synchronous methods are executed without exceptions and warnings.
When using web client, running in Google Chrome or Mozilla Firefox, the following behaviour is implemented:

- The property is set to Do not use -, the file system and cryptography extensions, which are represented by synchronous methods, do not load. The extensions, represented by the asynchronous modes, are used.
- In other cases - when connecting the file system and cryptography extensions loading of extensions representing the synchronous methods is attempted. If the attempt is successful, the possibility to use both synchronous and asynchronous file system and cryptography methods is provided. If the "synchronous" file system and cryptography extensions are not operable, the extensions, represented by asynchrounous methods will be used.

Please note, that synchronous methods support in Google Chrome is disabled by default since version 42 (April 2015) and is completely removed since version 45 (September 2015). Mozilla Firefox does not support synchronous calls in web browser since version 43 (December 2015).
At the same time the above behaviour will be observed if the `/EnableCheckExtensionsAndAddInsSyncCalls` client application startup command line parameter is used.
If 1C:Enterprise is started from the Designer, `/EnableCheckExtensionsAndAddInsSyncCalls` client application startup command line parameter is set mandatory without the option to configure

such behaviour. The exception from above is starting the thick client, for which this command line parameter is not set when starting from the Designer.

Interface compatibility mode - this property manages the client application interface mode:

- Version 8.2 - client application runs in version 8.2 interface. Switchin to Taxi interface is not possible. This mode is not supported when using the Web client.
- Version 8.2. Allow Taxi - version 8.2 interface is used by default. It is possible to switch to Taxi interface using parameters dialog box, `ClientSettings` object or command line. This mode is not supported when using the Web client.
- Taxi. Allow Version 8.2 - Taxi interface is used by default. It is possible to switch to version 8.2 interface using parameters dialog box , `ClientSettings` object or command line. This mode is not supported when using the Web client.
- Taxi - client application uses Taxi interface. Switching to 8.2 interface mode is not possible.

When using version 8.2 interface it is possible to switch between the interface in separate windows and interface in tabs.

Only Taxi interface is supported in the wen-client. If another interface compatibility mode is set in the configuration settings with the help of the embedded language or startup command line command -these settings are ignored and Taxi interface will be used.

If Compatibility mode property is set to value, exceeding Version 8.3.2 (Do not use, Version 8.3.3 and higher), and Interface compatibility mode property is set to Version 8.2 or Version 8.2. Allow Taxi, the interface in tabs will be automatically set for the new application users.

`PictureLib.Help` image is displayed in many ways, if Interface compatibility mode property is set to Taxi or Taxi. Allow Version 8.2 and in other cases.

Compatibility mode - the property controls the behaviour of features, which has been changed in the new system version in comparison to the previous versions. This property can accept the following values: Version 8.3.15, Version 8.3.14, Version 8.3.13, Version 8.3.12, Version 8.3.11, Version 8.3.10, Version 8.3.9, Version 8.3.8, Version 8.3.7, Version 8.3.6, Version 8.3.5, Version 8.3.4, Version 8.3.3, Version 8.3.2, Version 8.3.1, Version 8.2.16, Version 8.2.13, Version 8.1 and Do not use.

Using configuration with unknown compatiblity mode is not supported. Unknown compatibility modes are the modes, that do not match the functions, implemented in the following 1C:Enterprise versions. For example, if the compatibility mode is set to Do not use in version 8.3.1, it will be considered unknown when opening the configuration in version 8.2.16. At the same time, if the compatibility mode is set to Version 8.2.16 in version 8.3.1, then when opening the

configuration in version 8.2.16, it will be displayed as Do not use. If the specific version does not introduce a new compatibility mode, the effect of the Do not use mode will be similar to the previous version. If attempting to run or load a configuration with the unknown compatibility mode, an error will be displayed indicating the required version. Importing 1cv8.dt files generated in version 8.3.1 and later into 1C:Enterprise versions prior to 8.3.1 is not allowed. The only exception is when the Compatibility mode configuration property is set to Version 8.2.16 in 1C:Enterprise 8.3.1 and later.
When converting configurations of the 1C:Enterprise 8.1 versions (and earlier), the property assumes the Version 8.1 value. In general case, it may be said that when a configuration is opened with a later version of 1C:Enterprise, the Compatibility mode property will be set to the compatibility with the previous version if such mode is introduced in the new version.
If it is necessary to ensure the operation of the application in several 1C:Enterprise versions at once (including for the ones with a compatibility mode), then it is recommended to first receive the current version of the platform in areas of different code calls and then compare it with the compatibility mode (if necessary). However, it is not recommended to check the compatibility mode for the `DontUse` value.
The mobile platform compatibility mode has several differences:

- Mobile platform compatibility mode is not managed by configuration properties (including the Compatibility mode property). However, this property exists "inside" the mobile application.
- The setting of the compatibility mode value is performed when creating the infobase. The compatibility mode value will correspond to the version of the mobile platform that was used to create the infobase.
- Updating the mobile platform version does not update the compatibility mode value.
- When updating the mobile application infobase configuration, the compatibility mode is set to a value equal to the mobile platform version which was used for configuration update.
- Accordingly, all the methods of 1C:Enterprise language the behavior of which considers the compatibility mode will consider the compatibility mode generated as per the above rules.

5.2. Managed application module

The managed application module, being part of the configuration, is saved only as part of the configuration. Using the File - Save item will result in the saving of the changes made to the entire configuration.

5.3. External connection module

5.4. Session module

5.5. "Common" configuration branch

This section describes some configuration objects that are located in the Common branch of the configuration tree. These objects do not describe the data structure and their processing functions. These are intended for setting user data processing rules, descriptions of auxiliary objects used to generate various forms in the data exchange engine, and also contain common modules and print form templates available from any configuration module.

5.5.1. Subsystems

NOTE. Setting the subsystem name `Desktop` is not recommended.

No restrictions on the number and nesting is applicable to the Subsystem branch objects structure.

To view configuration objects related to a specific set of subsystems, it is possible to configure the objects selection filter in the Configuration window. Select the Actions -By Subsystems item of the Configuration window and specify the required set of subsystems, as well as set additional selection criteria Include objects of subordinate subsystems and Include objects of parent subsystems.

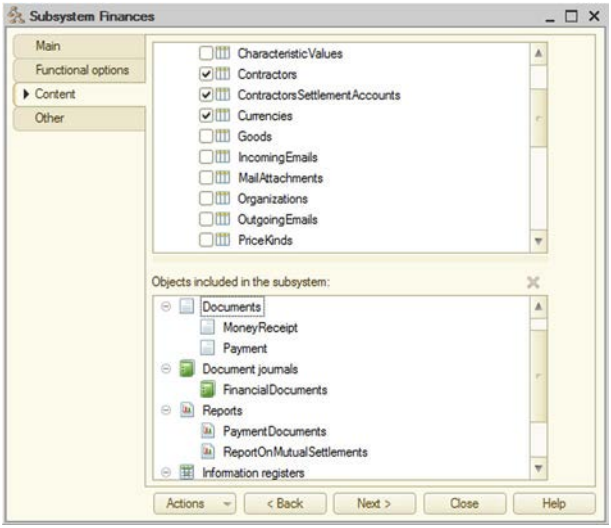


Fig. 55. Selection by subsystems

In the list of subsystems, there is a special item <Not included in subsystems>, using which it is possible to select only the objects that do not belong to any subsystem.

For a quick selection of objects included in a specific subsystem, the special context menu command may be used: Subsystem objects. Execution of this command is equivalent to execution of the Action- By subsystems command with indication of the only one subsystem in the opened dialog box.

NOTE. When selecting by subsystem, the key branches of the configuration object tree, which do not contain selected objects, are not shown.

The belonging of configuration objects to a specific subsystem defines the user interface.

The value of the `Subsystem` configuration object property is available in the program using the 1C:Enterprise language tools. This provides additional options for data selecting.

The Move Subsystem context menu command allows changing the subordination of the subsystem in the hierarchy of subsystems.

In order to “bind” metadata objects to a specific subsystem (subsystems), use the Structure tab of the subsystem editor.

The top section of the window lists all the configuration objects that can be assigned to different subsystems. Ticking an object means that the object starts to be included in the subsystem and appears in the bottom section of the window. The bottom section of the window lists all the objects that are included in the edited subsystem.

The Image property allows to specify an image that will be displayed in the sections panel (for top-level subsystems). When specifying the images of different size, the height of the section panel will be calculated depending on the interface compatibility mode:

- Version 8.2:
  - The section panel height is calculated based on the fact that the vertical size of all images is 48 pixels.
  - Smaller images are proportionally decreased so that the vertical size is 48 pixels.
- Taxi:
  - If there is at least one section image where the vertical size is more than 48 pixels, all such images will be proportionally decreased so that the vertical size is 48 pixels.
  - If all the section panel images have a vertical size of less than 48 pixels, then the height of the section panel is calculated based on the maximum vertical size of the images, while the smaller images are centered in the areas reserved for the images.

If the Include in the help content property is set, then the help content will include a branch that displays help on the subsystem and all the objects included in it. If the property is cleared, then such a branch (describing the subsystem and objects included in it) will not be included in the help content, but the help on the objects included in the subsystem will be available directly in the forms of the objects themselves.

5.5.2. Common modules

Objects located in a of the Common modules configuration tree branch are intended to contain the texts of functions and procedures that can be called from any other configuration module.

IMPORTANT! A common module may only contain the definitions of procedures and functions.

The procedures and functions of the common module, for which the `Export` keyword is shown in the headers, are the ones of the **global context** components. For more information about writing procedures in a common module please see "Program modules source code format" and "Operators" sections of the 1C:Enterprise language help.

To edit the common module it it necessary to click on Open link in the Common modules type object property palette in the Module property of the Configuration window. The text of the common module will be displayed for editing in the 1C:Enterprise text editor in the program module text editing mode.

The common module, being part of the configuration, is saved only as part of the configuration.

The Global property defines whether exported methods of the common module are a part of the global context.

When the Global property is set to `True`, exported methods of the common module are available as global context methods.

When the Global property is set to `False,`, a property is created in the global context with the name matching the name of the common module in the metadata. This property is read only. The value of this property is a `CommonModule` object. Exported methods of this common module are available through this object. Thus, nonglobal common module method calls have the following format: `XXXXX.YYYYY`, where `XXXXX` - is the name of the common module context matching property, and `YYYYY` - is the name of the exported method of the common module.

Example:

```
OperationsWithShopEquipment.ConnectBarCodeScanner();
```

5.5.2.1. Various contexts and common modules

Using the common modules properties and preprocessor commands it is possible to arrange the execution of various methods of common modules in the desired context.

Each property of a common module is responsible for the possibility to compile (and execute) a common module in a particular context.

The following properties, responsible for the context, are available, where the methods of the common module are available:

- Client (ordinary application) - common module methods will be available to the thick client in the ordinary application mode;
- Client (managed application)- common module methods will be available to the thin client, web client, and the thick client in the managed application mode;
- Server - common module methods will available on server;
- External connection - common module methods will be available in the external connection.

If several properties are set at the same time, this means that the common module methods will be available in several contexts.

If the Server property and another property are set for common module, this means that the common module will be available simultaneously on the server and in the selected client. It should be understood that these in fact will be several variants of the compiled code (based on the number of selected clients and for the server itself).

However, if a method located in such a common module is called from the client side, then a client copy of the common module will be used, and if it is called from the server - the copy from the server will be applied. In this case, using preprocessor commands, it is possible to “isolate” the server from the code that cannot be executed on it.

Consider an example. In the general module (which can be executed on the thin client and on the server) there is a method that has slightly different behaviour on the thin client side and on the server side. Let us see how to do this:

```
Procedure CommonModuleMethod() Export

  // Place various important code here
#If ThinClient Then

  // Show a warning
  ShowUserNotification ("In client");
#EndIf

EndProcedure
```

The server-side code will be as follows:

```
Procedure CommonModuleMethod() Export

  // Place various important code here
EndProcedure
```

The thin-client-side code will be as follows:

```
Procedure CommonModuleMethod() Export
  // Place various important code here

  // Show a warning
  ShowUserNotification ("In client");

EndProcedure
```

There are several ways to pass control from client to server:

- call the server common module method;
- call the the method which is preceded by `&OnServer`, `&OnServerWithoutContext` compilation directives in the form or command module.

At the same time, it is impossible to call the methods of client common modules (which do not have the Server property set) and client methods of the form module or the command module from server procedures. The control will return to the client after the most external call of the server method is completed.
The exceptions are the methods of the form module and the command module, which are preceded by compilation directives`&OnClientOnServer`, `&OnClientOnServerWithoutContext`.
The following points should also be mentioned:

- If a common module is available for more than one client, when writing a program code the maximum restrictions that can be imposed by clients should be considered the preprocessor commands to "isolate" code specific to a particular client should be used.
- The preprocessor commands also make sense when one common module has several execution contexts, for example, an external connection and a thin client or (which is much more common) any client and a server. In this case, preprocessor instructions will surroung interactive code that cannot be used on the server, but is possible on the client (see the example above).

The Server call property is intended to manage the possibility to call the exported methods of the server common module from the client code. If the property is set, the exported methods of the server common module are available for calling from the client. If the property is not set, such exported methods can be called only from server methods (both methods of server common modules and server form module methods and command modules).

TIP. It is recommended to set the Server Call property to `False` in cases where the server common module contains methods that are undesirable to call from the client (for example, for security reasons).
NOTE. If the Client (ordinary application), Client (managed application), External connection properties are set at the same time, the Server call property is automatically reset. If the Server Call property is set, the Client (ordinary application), Client (managed application) and External connection properties are automatically reset if these properties are set at the same time.

The Privileged property is designed to disable access control when executing common module methods.

NOTE. If the Privileged property is set, the common module is automatically assigned the Server property and all the other properties (Client (ordinary application), Client (managed application) and External connection) are reset. Privileged common module may only be executed on the server.

5.5.2.2. Return values reuse

If the common module is not global, then the Return values reuse property becomes available. This property can have the following values:

- Do not use - reusing return values for functions of this common module is not allowed.
- During call and During session - the return values reuse determination method is used for the common module. The essence of this method is that during the execution of the code, the 1:C Enterprise caches the parameters and the result of the functions after the first function call. When the function is called again with the same parameters, the cached value is returned (from the first call) without executing the function itself. If the function execution changes its parameters, the return values are not reused.

Storing the function result has some specificities:

- when a function is executed on the server and is called from the server code, the parameter values and the function result are cached for the current session on the server;
- when a function is executed on the thick or thin client, the parameter values and the function results are cached on the client;
- when a function is executed on the server but called from the client code, the parameter values and function result are cached both on the client and on the server (for the current session).

Cached values are deleted:

- if the common module property is set to During call:
  - on the server - when the server returns the control;
  - on the client -when the execution of the top-level 1C:Enterprise language procedure or function (called from the interface rather than from another 1C:Enterprise language procedure or function) is completed;
- if the common module property is set to During session:
  - on the server - upon the session completion;
  - on the client -upon closing of the client application.

The cached values will be deleted:

- on the server, in the thick client, in the external connection, in the thin client and in the web client with the normal connection speed - 20 minutes after calculating the cached value or 6 minutes after the last use;
- in a thin client and in a web client with a low connection speed - 20 minutes after calculating the cached value;
- when there is a shortage of memory in the server working process;
- when a working process is restarted;
- when the client switches to another working process.

After the values are deleted, an exported function call is performed in the same way as for the first call.

This common module property does not affect the execution of procedures - the procedures are always executed.

If the common module is set to reuse the return values, the types of parameters of the exported function are subject to a number of restrictions. Parameters can only be of the following types:

- Primitive types (`Undefined`, `NULL`, `Boolean`, `Number`, `String`, `Data`).
- Any database object references.
- Structures with values of the above type properties. In this case, the identity of the parameters is controlled "by content" of the structures.

If the exported function returns any object, the reference to the object stored in the cache is actually returned. If, after receiving this reference, there is a change in the object condition, a subsequent call of the same function will result in returning the reference to the object that has already been changed without actually executing the function. This behaviour will be observed until the cached value is deleted (for any reason). In other words, - a change in the object condition resulting from a function call from a common module with the return values reuse is not a reason for actually calling a function. It is also worth noting that the cache of the returned objects is insensitive to the privileged mode condition at the time of the function call with return values reuse. This feature may cause the following behaviour:

- The actual execution of the function call with the return values reuse (the first call) was performed with privileged mode enabled.
- When executing the function, an object was received that could not be retrieved with the privileged mode disabled.
- Subsequent function calls were performed without setting the privileged mode on.
- However, until the cache of the returned objects is cleared or the actual call is repeated, the function will return a formally unavailable object.
- The reverse behaviour is also applicable when the first call is performed without setting the privileged mode, and in the privileged mode the object that could have been retrieved in the privileged mode is not returned.

If the Return values reuse common module property is set to During session, the values, returned by the functions of such module, can not use values of `TempTablesManager`, `Query` types and applied objects (e.g., `ProcessingObject` or `DocumentObject`), both directly and within any collections. The exceptions are the reference type data (`DocumentRef`, `CatalogRef`, etc.).

If the common module function with the reuse configured is called from the same common module (for example, with the `CommonModule` name), the following peculiarity should be kept in mind: if the function is called by the `MyFunction()` name, the function will be executed each time the function is called. In order to use the cached values, the function must be called by the full name: `CommonModule.MyFunction()`.

The `RefreshReusableValues()` global context method deleted all the reusable values, both on the server and on the client, regardless of where the method is called from. After executing `RefreshReusableValues()` method the first function call will be executed completely.

5.5.3. Session parameters

Session parameters are mainly intended for using parameter values in queries and data access restriction conditions for the current session.
Using the session parameters reduces data access time by excluding the related tables.
Session parameters are configured in the property palette.
For each session parameter, two access rights are defined - `Retrieving` and `Setting` (see the section below for more details). If the `Setting` right is removed, then the initialization of this session parameter is only possible in the common module with the Privileged property set or in the session module.
Initialization of session parameters can be performed in the session module, in the `SessionParametersSetting()` event handler.
Prior to initialization, the session parameter value is undefined. When attempting to read such parameter, the `SessionParametersSetting()` event handler is called first. If the parameter remains undefined after the call, an exception is thrown.
The managed application module (external connection module) session parameters and global variables application areas should be distinguished. The main differences of session parameters are:

- Session parameters are metadata objects, which allows 1C:Enterprise to exercise enhanced control over their use.
- Session parameters have a type. The set of session parameter types is limited. Their important common feature is the impossibility of changing the internal condition of the objects of these types.
- To set or retrieve the value of a session parameter, the current user must be granted the appropriate rights.
- In the client/server mode of 1C:Enterprise, the values of the session parameters are stored on the server and are accessible both from the server and from the client.
- Session settings are available from both 1C:Enterprise language, for example:

```
SessionParameters.CurrentUser = Username()
```

And from the data access restrictions, for example:

```
Document.Report.User = &CurrentUser
```

In the latter case, the current user requires no respective right to retrieve the value of the session parameter.

NOTE 1. If the session parameter is set to one of the following types: `FixedArray`, `FixedCollection` or `FixedStructure`, the `Undefined` value may be the value of the collection item.
NOTE 2. The amount of data transferred between the client and the server to set one session parameter must not exceed 4 GBytes (in serialized form).

5.5.4. Roles and access rights

5.5.4.1. General information

Each user must have free access to common information, such as common catalogs, constants or enumerations.
On the other hand, it is necessary that each user dealt only with the information that they require for work, and could in no way by their careless actions affect the operations of other users or the performance of 1C:Enterprise as a whole.
The 1C:Enterprise Designer provides developers with advanced administration tools designed to solve these problems.
First of all, during the process of configuration generation, the necessary number of typical roles is created, which describe the rights of various categories of users to access the information processed by the 1C:Enterprise. Roles may be set in a fairly wide range - from the possibility of only viewing a limited number of document types to the full set of rights to enter, view, adjust and delete any types of data.
1C:Enterprise has two types of rights - **basic** and **interactive**. **Basic** are always checked, regardless of the infobase objects access method. **Interactive** are checked when executing the interactive operations (viewing and editing in the form, etc.).
If the `View` right is set (permitted) for the object, which data is represented in the form, but the `Edit` right is not set, this attribute (the control, associated with this object, displays the attribute value) will be displayed in the form, but editing the value will not be available. If the `View` right is removed, then when attempting to open the form, the following warning is displayed: Access rights violation!, and the form will not be opened.
In the list of rights when editing a role, the attention must be paid to the internal rights hierarchy. The hierarchy is represented in the form of "seniority" of the rights. When removing a “senior” right, other rights (“junior rights”), related to the “senior” right, are removed; and vice versa, when setting the “junior” rights, the removed “senior” rights are set. Thus, when removing `View` right the `Edit` right also is removed. Which is quite logical, since it makes no sense to provide the right to edit if it is impossible to display the control related to the data.

In general case, the rights may be set for:

- entire configuration,
- objects,
- object attributes,
- tabular sections,
- tabular section attributes,
- standard attributes,

When creating a new role, the following rights are set for the root configuration object: `ThinClient`, `WebClient`, `SaveUserData`, `Output`.

5.5.4.2. Privileged operation mode

On the 1C:Enterprise server, code fragments can be executed both in normal and in privileged mode. In privileged mode, access control at the record level is not performed, rights are not controlled, and any operations are allowed, which speeds up the execution of modules.

To control the privileged mode, use the `SetPrivilegedMode()` global context method, which allows to enable or disable the privileged mode.

IMPORTANT! In the client/server mode the method call does not affect the usage of the client.

Privileged mode is disabled by default.

The number of privileged mode activations must match the number of shutdowns. However, if a privileged mode (once or more) was enabled within the procedure or function, but was not disabled, the it will be automatically shut down as many times as there were incomplete activations in the procedure or function

If the procedure or function contains more `SetPrivilegedMode(False)` method calls, than `SetPrivilegedMode(True)` method calls, the exception will be thrown.

`PrivilegedMode()` functions returns `True` value, if the privileged mode is enabled, and `False` value, if it is completely disabled. At the same time It does not analyze the number of privileged mode settings in a particular function.

The programmatical setting of the privileged mode may be required in case of massive operations with the infobase data, and when it does not make sense to check the data access rights. For example, there is a user who is entrusted to perform the recalculation of the prices for goods. Then, in data processor, for example, which performs this operation, it is possible to check the current user's right to perform this data processing, and then enable privileged mode and perform all the necessary operations with the database. At the same time, the user may not have rights to read prices. But since this data processor does not output the prices themselves to the user, but only recalculates them, the assigned access restriction tasks will also be solved.

It is also possible to start a privileged session. This is a session where privileged mode is set from the very beginning of the 1C:Enterprise operation. In this case during the operation the `PrivilegedMode()` method will always return `True`, and the possibility to disable the privileged mode is not supported. Only a user with administrative rights (`Administration` right) can start a privileged session. Session start is possible using the `UsePrivilegedMode` client application startup command line option or `prmod` infobase connection string parameter.

5.5.4.3. Safe operation mode

If it is necessary to use “unreliable” program code on the server: external data processors or program code entered by the user for use in the `Run()` and `Calculate()` methods, the safe operation mode may be used.

In safe mode:

- Privileged mode is **cancelled**.
- Swtiching to privileged mode is **ignored**.
- Operations that cause the use of external tools in relation to the 1C:Enterprise platform (including non-blocking analogues of these methods) are **prohibited**:
  - COM mechanisms:
    - `COMObject()`;
    - `GetCOMObject()`;
    - `HTMLDocumentShell.GetCOMObject()`.
  - Loading add-ins:
    - `LoadAddIn()`;
    - `AttachAddIn()`.
  - File system access:
    - `ValueToFile()`;
    - `FileCopy()`;
    - `MergeFiles()`;
    - `MoveFile()`;
    - `SplitFile()`;
    - `CreateDirectory()`;
    - `DeleteFiles()`;
    - `New File`;
    - `New xBase`;
    - `HTMLWriter.OpenFile()`;
    - `HTMLReader.OpenFile()`;
    - `ReadXML.OpenFile()`;
    - `WriteXML.OpenFile()`;
    - `FastInfosetReader.OpenFile()`;
    - `FastInfosetWriter.OpenFile()`;

    - `CanonicalXMLRecord.OpenFile()`;
    - `XSLTransform.LoadFromFile()`;
    - `ZipFileWriter.Open()`;
    - `ZipFileReader.Open()`;
    - `New TextReader()`, if the first parameter - is a string;
    - `TextReader.Open()`, if the first parameter - is a string;
    - `New TextWriter()`, if the first parameter - is a string;
    - `TextWriter.Open()`, if the first parameter - is a string;
    - `New TextExtraction()`;
    - change of `TextExtraction.FileName` property;
    - `TextExtraction.Write()`;
    - `New Image()`, if the first parameter - is a string;
    - `Image.Write()`;
    - `New BinaryData()`;
    - `BinaryData.Write()`;
    - `New DataWriter()`, if the first parameter - is a string;
    - `New DataReader()`, if the first parameter - is a string;
    - all the `FileStreamsManager` object methods;
    - `New FileStream()`;
    - `FormattedDocument.Write()`;
    - `GeographicalSchema.Read()`;
    - `GeographicalSchema.Write()`;
    - `GeographicalSchema.Print()`;
    - `SpreadsheetDocument.Read()`;
    - `SpreadsheetDocument.Write()`;
    - `SpreadsheetDocument.Print()`;
    - `GraphicalSchema.Read()`;
    - `GraphicalSchema.Write()`;
    - `GraphicalSchema.Print()`;
    - `TextDocument.Read()`;
    - `TextDocument.Write()`.
- Internet access:
    - `New InternetConnection`,
    - `New InternetMail`,
    - `New InternetProxy`,
    - `New HTTPConnection`,
    - `New FTPConnection`.

IMPORTANT! When performing prohibited operations at run time the exception is generated.

NOTE. External reports and data processors that are opened using the File - Open menu, are performed in safe mode if the user does not have administrative access rights.

The number of safe mode activations must match the number of shutdowns. However, if a safe mode (once or more) was enabled within the procedure or function, but was not disabled, the it will be automatically shut down as many times as there were incomplete activations in the procedure or function.

If the procedure or function contains more `SetSafeMode(False)` method calls, than `SetSafeMode(True)` method calls, the exception will be thrown.

Programmical setting of the safe mode may be required in the case when the configuration developer assumes the use of third-party (with respect to the configuration) program code, the reliability of which can not be guaranteed by the developer. An example of such code is the execution of the `Run()` and `Calculate()` methods in cases where the executable code is obtained from the outside. In this case, good practice will be to set a safe mode before executing these methods:

```
// Generates a program code to be executed
// It is possible that the code is loaded from external sources
// or is entered manually
ExecutableCode = GetExecutableCodeFromOutsideWorld();

// Enable safe mode
SetSafeMode(True);

// Run the potentially dangerous code
Run(ExecutableCode);

// Disable safe mode
SetSafeMode(False);
```

In some cases, the safe mode settings may conflict with the privileged mode settings. An example of such conflict is the posting of the document for which the Privileged mode during posting property is set, from the code in the 1C:Enterprise language, which is executed in safe mode. In this case, the privileged mode is disabled, and attempts to enable it are ignored. As a result, the code in the 1C:Enterprise language that “counts” on the enabled privileged mode, “encounters” its absence, which leads to errors with unclear reasons of occurrence. To prevent such situation, the 1C:Enterprise automatically disables safe mode for event handlers that are available in the object module or the manager module, provided that the executable code in the 1C:Enterprise language is not located in the configuration extension. Such handlers are marked in a special way in the Syntax Assistant.

It also provides the possibility to disable safe mode from the 1C:Enterprise language (if the program code from which the shutdown is attempted is not located in the configuration extension). To disable safe mode, use the

`SetSafeModeDisabled()` method. Checking if safe mode is currently disabled (automatically or by calling a method) is performed using the `GetSafeModeDisabled()` method.

Within one method, in the 1C:Enterprise language there can be no more than one nesting level of the safe mode setting (by calling the `SetSafeMode()` method) and setting the safe mode disabled (automatically while the event handlers for metadata objects are running or by calling the `SetSafeModeDisabled()` method). When attempting to increase nesting, an exception is generated:

```
// Correct use
Procedure ProcedureName()
  SetSafeModeDisabled(True);
  SetSafeMode(True);
  SetSafeMode(False);
  SetSafeModeDisabled(False);
EndProcedure

// Incorrect use
Procedure ProcedureName()
  SetSafeModeDisabled(True);
  SetSafeMode(True);
  SetSafeModeDisabled(False); // Exception
 EndProcedure

Procedure ProcedureName()
  SetSafeMode(True);
  SetSafeModeDisabled(False); // Exception
EndProcedure
```

5.5.4.4. Data deletion modes

The 1C:Enterprise provides users with the option to delete unnecessary or outdated information in two modes:

- **direct object deletion**, which does not analyze the use of the deleted object in other database objects;
- use of **referential integrity control**, where the check is made whether the objects to be deleted are referenced by other database objects.

If the direct deletion mode is allowed for the user, in this case additional responsibility lies with the user performing the deletion of objects and the developer (system administrator), who determines the user rights and 1C:Enterprise actions in case of unresolved references. The 1C:Enterprise operation without reference integrity control can, for example, be used by specialists in the process of debugging the configuration. If the referential integrity control is not used, then the deletion of objects is performed directly (without a mark for deletion) and the possibility of creating the unresolved references appears.

The most radical method to set the referential integrity control is to disable the rights to directly delete objects in general. This completely excludes the possibility to directly delete any objects within this configuration. Users will only be able to mark objects for deletion.

Note that it is also possible to directly delete objects using the built-in language. Therefore, parts of a configuration can perform direct deletion regardless of the referential integrity monitoring mechanism. In this case, the responsibility for data integrity lies with the specialist, who performs the 1C:Enterprise configuring.

5.5.4.5. Role combination rules

Roles are usually specified for each activity. When adding a new user to the user list they may be assigned a specific role or a set of roles. If several roles are used, the algorithm for granting access to each object and type of access rights (for example, the Mark for deletion) will operate as follows:

- if **at least one** role has **the permission**, the access will be **granted**;
- if **all the** roles have **prohibition**, the access will not be **granted**.

5.5.4.6. Access rights editor

The left pane of the rights editing dialog box displays a tree of configuration objects for all subsystems. The right pane - displays a list of rights for the selected configuration object in the configuration tree. If a check box is set for an action, it is allowed.

So, for example, a user with the `SalesManager` role is allowed to view the `GoodsArrival` document and its interactive addition is prohibited.

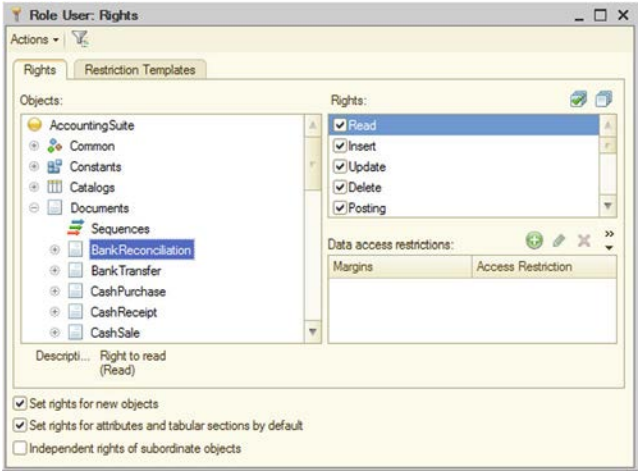


Fig. 56. Role access rights editor

The state of the Set rights for new objects check box determines whether the rights for this role will be set for the newly added configuration objects (removed for the new role by default).

The state of the Set default rights for attributes and tabular sections check box determines whether the rights for this role are set for attributes (including

standard ones) and tabular sections (including standard ones) for the new configuration objects (set by default).
When the state of the Set default rights for attributes and tabular sections check box is changed, the user is proposed to change (set or reset) access rights for all the attributes (including standard ones) and tabular sections (including standard ones) of all the configuration objects. If the proposed action is refused, there are no changes in the existing objects, and the default behavior for new objects is changed only.
When creating a new role, all the rights are set by the Designer as follows:

- no rights for objects are set;
- rights are set for attributes (including standard ones) and tabular sections (including standard ones).

The Independent rights for subordinate objects check box determines how the 1C:Enterprise will take consider the state of rights for the parent object when determining the rights for a subordinate object. If the box is checked, the rights for the parent object are considered. If the check box is unchecked, when determining the right for a subordinate object, the corresponding right for the parent object is analyzed. If the parent object has no right, the subordinate object has no right also, regardless of the state of the right for the subordinate object.
The Independent rights for subordinate objects check box does not lift the requirement for having the same right for the parent object as for the subordinate one. That is, for the attribute or tabular section, the user needs to have the right for the object, and for the attribute of the tabular section - for the tabular section and the object. This property has an effect if the user has several roles assigned, the rights of which are combined “using OR logic”. That is, if the Independent rights for subordinate objects property is set in a role and there is a right for the the attribute only, and another role has a right for an object only, then after combining the rights of roles, the user will have the right to attribute
When assigning access rights for the attributes (tabular sections) of reports/data processors, the following peculiatiry should be kept in mind: if Independent rights for subordinate objects check box is removed for the role and the `Edit` right is set for attribute (tabular section), and the `View` right is not set for report/data processor, it is considered that the `Edit` right is not set for attribute (tabular section).
If a user is assigned several roles, then the rights for the parent object are checked before a combination of rights by roles is performed (to determine the resulting rights to the object).
The Independent rights for subordinate objects check box effect is applied to the following objects:

- attributes (including standard ones);
- tabular sections (including standard ones);
- tabular section attributes (including standard tabular section attributes);
- commands.

To change the access right, select the configuration object in the left list, and change the state of the check box in the required action type row in the right list. If it is required to change access to all the objects of the selected branch at once, specify this branch in the left pane and change the access rights setting.
The description of each role can be displayed in a tabular or text document using the Actions - Display list item.

5.5.4.7. Viewing and editing of all the roles

If several roles are used in the configuration, it is recommended to use the All roles window for ease of viewing and editing the rights. To open it specify the Roles branch in the configuration objects tree of the Configuration window and select the All roles command in the context menu.

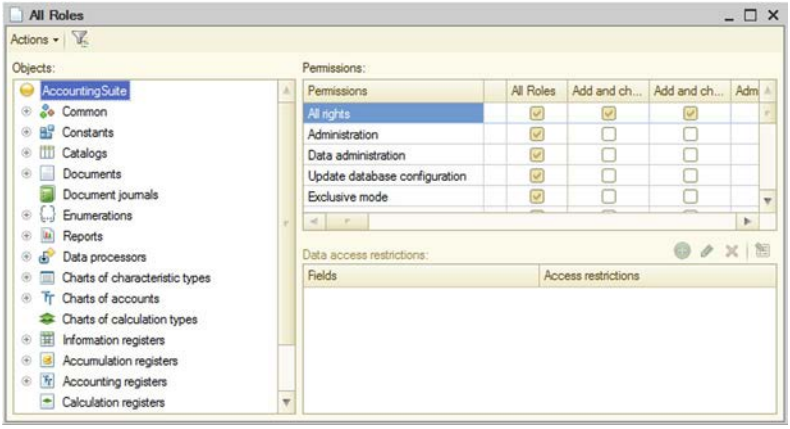


Fig. 57. "All roles" editing window

The window has three tabular fields. In the first (to the left), the desired configuration object is selected. The first column of the second tabular field displays a list of rights for the selected object. Other columns are intended to indicate the use of each right for each existing role.
If it is required to set or remove all rights for any role, it is enough to check or uncheck the permission box in the first row of the tabular field.
If it is required to check or uncheck the permission box in all the roles for some right, it is enough to check or uncheck the permission box in the first column of the tabular field.
The possibility to rearrange columns corresponding to roles is supported.
The third tabular field allows to edit data access conditions at the level of individual fields and records.

5.5.4.8. Data access restriction

5.5.4.8.1. General notes

The data access restriction mechanism (also known as RLS, Row Level Security) allows to manage access rights not only at the level of metadata objects, but also at the level of 1C:Enterprise database objects. To restrict access to data, the following 1C:Enterprise objects can be used:

- roles,
- session parameters,
- functional options,
- privileged common modules,
- `PERMITTED` keyword in query language.

Joint use of these objects allows for maximum flexibility when it is necessary to differentiate data access rights between users who perform various functions.
Data access restrictions may be imposed on the following data operations (access rights): reading (`Read` right), adding (`Adding` right), editing (`Editing` right) and deleting (`Deleting` right). The current user will be able to perform the required operation in the following cases:

- For reading and deletion operations, the object in the database must comply with the data access restriction.
- For the adding operation the object that is to be written to the database must comply with the data access restriction.
- For a change operation, the object both before the change (for the object to be read) and after the change (for the object to be written) must comply the data access restriction.

When imposing data access restrictions, it should be noted that only one condition can be set for changing, adding and deleting operations, and more than one data access restriction can be set for a reading operation. This means that different conditions can be set to read different fields of an object, and when specifying a condition it is possible to set both the name of a specific field and the Other Fields special field. In the first case, the condition will be imposed only if the selection (using which the data is read) has a field for which the restriction is set, and in the second case - the restriction is imposed to all fields of the object, except the fields for which the restrictions are explicitly specified.
When setting a restriction for a specific field, this field will be read if the restriction is met, and when setting a restriction for Other fields, the object data will be read only if the restriction is met for all the fields of the object that fell into the data read request.
For database objects of the following types, various restrictions can be imposed on different types of changes (addition, changing, deletion):

- Exchange plans
- Catalogs

- Documents
- Charts of characteristic types
- Charts of accounts
- Charts of calculation types
- Business processes
- Tasks

For the following types of database objects, it is possible to impose restrictions on reading of not only the entire object, but also its individual fields:

- Exchange plans
- Catalogs
- Documents
- Document journals
- Charts of characteristic types
- Charts of accounts
- Charts of calculation types
- Information registers
- Business processes
- Tasks

IMPORTANT! When accessing the fields of database objects using the properties of applied objects from the 1C:Enterprise language, the entire object is read, and not just the value of the field used. The exception is getting the presentation where only the values of the fields involved in the generation of the presentation are read.

Access restrictions are contained in roles; these can be specified for most metadata objects and are written in a special language, which is a subset of the query language.

5.5.4.8.2. Data access restrictions language

Data access restrictions are described in a special language, which is a subset of the query language. The data access restriction language has the following changes in comparison to the query language:

- In a data access restriction request, there is always one table as a data source. - this is the table of the object on which the restriction is imposed (the main restriction object).
- Brief query description. Data access restriction language uses query language sections `FROM` and `WHERE` only. Thus, the description of the query language is as follows:

<Query description>

```
ВЫБРАТЬ [РАЗРЕШЕННЫЕ] [РАЗЛИЧНЫЕ] [ПЕРВЫЕ <Количество>]
  <Selection fields list>
[FROM <Source list>]
[WHERE <Selection condition>]
[GROUP BY <Grouping fields>]
[HAVING <Selection condition>]
[FOR CHANGING [<Top level tables list>]]
```

While the description of the data access restriction query language is as follows:

<Query description>

```
[Main restriction object table alias]
[FROM <Source list>]
[WHERE <Selection condition>]
```

It should be noted that in the nested queries used in the data access restriction language, the set of possibilities is limited;

- It is possible to specify session parameters and functional options as condition elements;
- It is possible to use templates that simplify writing restrictions at any point of the data access restriction request.

The main section of the restriction is the condition that is calculated for each database table record, on which the data access restriction is imposed. A record is considered available if, a non-empty table (i.e., a table with 1 or more records) is retrieved as a result of a condition application for a single record of the main restriction object table. If, as a result of the condition application, an empty table is retrieved -, the record, for which the condition has been met in this way, is considered unavailable. Moreover, a change in the main restriction object table record is considered valid if the record does not contradict the restriction specified for the right, both before the execution of the change operation and after the execution of this operation.

Table fields

The following may be used in data access restrictions:

- **Object table fields** for which the data access restrictions are described. For example, if a restriction is imposed on the reading of elements of the `Counterparties` catalog, then the restriction may use the fields of the `Counterparties` catalog and its tabular sections. In particular, the simplest restrictions on the reading of the `Counterparties` directory elements may be as follows:

```
WHERE Name = "Brick Factory"
```

Or like that:

```
WHERE Products.Name = "Red brick"
```

Where `Products` -is a tabular section of the `Counterparties` catalog.

- **Fields of object tables accessible by references** in the main restriction object**.**
  For example, if the `MainManager` attiribute of the `Counterparties` catalog has the type of reference to the `Users` catalog, then the access restriction may, for example, be as follows:

```
WHERE MainManager.Code = "Ivanov"
```

Or:

- WHERE MainManager.PhysicalEntity.Name = "Petrovsky"
  Fields of object tables associated with the main restriction object by certain conditions and expressions over them.
  For example, the following restrictions may be imposed on the reading of the `Counterparties` catalog elements:

```
Counterparties
FROM
Catalog.Counterparties AS Counterparties
LEFT CONNECTION Catalog.Users AS Users
BY Counterparties.MainManager.Name = Users.Name
WHERE Users.PhysicalEntity.Name = "Petrovsky"
```

This restriction uses the `Users` catalog element fields associated with this element in the `Counterparties` catalog by the value of the `Name` fields.

Nested queries

Nested queries are used to generate sets of records that can be used for:

- linking to the table of the main restriction object;
- using as an operand in of comparison operations `IN` or `NOT IN`.

Nested queries may use any means of the query language, except for:

- `IN HIERARCHY` operator;
- `TOTALS` expression;
- results of nested queries must not contain tabular sections;
- some virtual tables, in particular `BalancesAndTurnovers`.

In the following example of the restrictions on reading from the `Counterparties` catalog, the nested query is used as a set of records for linking to the main restriction object:

```
Counterparties
FROM
Catalog.Counterparties AS Counterparties
LEFT CONNECTION
    (SELECT
      Users.Name.Users.PhysicalEntity
      FROM
      Catalog.Users AS Users
      WHERE
      Users.Code > "Petechkin") AS Users
BY Counterparties.MainManager.Name = Users.Name
WHERE Users.PhysicalEntity.Name = "Petrovsky"
```

The following example shows the restriction on reading from the `PhysicalEntityPassportDetails` catalog, where the nested query is used as an operand of the comparison operation `IN`:

```
PhysicalEntityPassportDetails
WHERE
PhysicalEntityPassportDetails.PhysicalEntity IN
    (SELECT VARIOUS
      Workers.PhysicalEntity AS PhysicalEntity
      FROM
      InformationRegister.Workers AS Workers)
```

If in a nested query it is necessary to retrieve data from the tabular section, then it is necessary to refer directly to the tabular section in the section `FROM` of the nested query. For example, instead of:

```
SELECT
  Reference AS Reference,
  Products.Name AS ProductsName
FROM Catalog.Couterparties
```

as a query nested in a restriction, the following should be used:

```
SELECT
  Reference AS Reference,
  Name AS ProductsName
FROM Catalog.Counterparties.Products
```

Session parameters

The data access restriction queries may include session parameters. For example, the following access restriction can be set for reading the elements of the `ElectronicMailGroups` catalog:

```
WHERE Owner.AccountAccess.User = &CurrentUser
    AND Owner.AccountAccess.Administration = True
```

Where `CurrentUser` - is a session parameter.

Functional options

The data access restriction queries may include functional options. Parameter-independent functional options can be used only. For example, if the `Products` catalog has a `MainWarehouse` attribute, then the restriction on reading this attribute may be as follows:

```
WHERE &AccountingByWarehouses = True
```

Where `AccountingByWarehouses` - is a fnctional option.

Specifications

Not all the fields of the main restriction data object can be used in the restrictions on database objects of the following types:

- in **accumulation registers** access restrictions may contain main restriction object dimensions only;
- in**accounting registers** the restrictions may use balance dimensions of main restriction object only.

NOTE. If the conditions of access restriction to the turnover accumulation register data use dimensions that are not included in the totals, then when accessing the virtual turnover table, the stored totals are not used and the query is executed completely according to the movements table.

5.5.4.8.3. Access restriction actions

Access restrictions are checked when any relevant operations are performed on database objects (from dialog boxes, from the 1C:Enterprise language, through queries) and may act in one of two methods:

- **All**. The “all” method implies that some operation on the data (from dialog boxes, from the 1C:Enterprise language or through queries) must be performed on all database objects implied by the operation. If during such an operation the database objects, for which the corresponding access restrictions are not met, are to be read or modified, the operation fails due to an access violation.
- **Permitted**. The "permitted" method implies that only those database objects that meet the relevant access restrictions must be read when performing an operation on data. Database objects that do not meet the access restrictions are considered absent when performing such an operation and do not affect the result of the operation.

Data access restrictions are imposed on database objects at the time when 1C:Enterprise addresses the database. In the client/server mode of 1C:Enterprise imposing restrictions is performed on the "1C:Enterprise" server.

The mode of restrictions effect chosen to perform each operation on data is determined by the purpose of this operation and the responsibility degree of its results. In particular, the "permitted" method is used when displaying dynamic lists and some other interactive actions. The “all” method is used when performing any operations with applied objects from the 1C:Enterprise language, including any changes to database objects. Therefore, for example, it may be difficult to build a selection for the `Select()` method of catalog, documents and other managers, and then bypass the result if quite a complex restriction is imposed on the corresponding object, since not all the conditions for access rights restriction can be adequately represented as a selection for the `Select()` method.

The method, using which data access restrictions are effected, can be controlled in queries. For this purpose, the query language provides the `PERMITTED` keyword. If the query does not indicate `PERMITTED`, the restrictions are applied using the “all” method. If the `PERMITTED` word is indicated, then the "permitted" method is selected.

It is important that if the query does not specify the `PERMITTED` keyword, then all the selections specified in this query must not contradict any of the restrictions on reading the database objects used in the query. In addition, if the query uses virtual tables, then the corresponding selections must be imposed on the virtual tables themselves.

Example:

```
SELECT
  ContactDetailsSliceFirst.Presentation
FROM InformationRegister.ContactDetails.SliceLast(, Type = &Type)
   AS ContactDetailsSliceFirst
WHERE
  ContactDetailsSliceFirst.Type = &Type
```

When using object method, data access in the `PERMITTED` mode is not supported. It is assumed that the object method is used for the most important data operations, including for their change. To retrieve all the data using the object method, regardless of the restrictions impossible, the necessary actions can be performed in a privileged module or as user with full rights. There is no means of retrieving only permitted data using the object method.

5.5.4.8.4. Restrictions imposing mechanism

Any operation on data stored in the database in 1C:Enterprise ultimately results in accessing the database with some query to read or modify the data. In the process of executing queries to the database, the internal mechanisms of 1C:Enterprise impose access restrictions. Further:

- A list of rights (read, add, change, delete), a list of database tables and a list of fields used by this query are generated.

- Data access restrictions are selected from all the roles of the current user for all the rights, tables and fields involved in the query. At the same time, if any role does not contain restrictions on access to the data from any table or field, this means that the values of the required fields from any record are available in this table. In other words, the absence of a data access restriction means the presence of `WHERE True` restriction.
- The current values of all the session parameters and functional options involved in the selected restrictions are retrieved.
  To retrieve the value of the session parameter from the current user, the right to retrieve this value is not required. However, if the value of a certain session parameter was not set, an error will occur and the database query will not be executed.
  The retrieving of functional options is affected by the functional option Privileged mode upon retrieval property. If this property is reset, the current user must have read permissions on the object which stores the functional option.
- Restrictions derived from one role are combined by `AND` operation.
- Restrictions derived from different roles are combined by `Or` operation.
- The generated conditions are added to the SQL queries used by 1C:Enterprise to access the DBMS. When accessing data from the access restriction conditions, the rights are not checked (neither to metadata objects, nor to database objects). Moreover, the mechanism for adding the conditions depends on the chosen mode of restrictions: “all” or “permitted”.

"All" method

When imposing restrictions using the “all” method, conditions and fields are added to the SQL queries so that 1C:Enterprise can receive the information on whether the data that was forbidden for this user was used in the process of executing a database query. If the forbidden data has been used, the query is terminated abnormally. The layout of access restrictions imposing using the “all” method is shown in fig. 58:

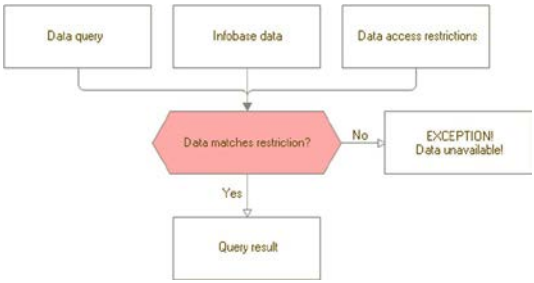


Fig. 58. "All" method

"Permitted" method

When restrictions are imposed using the “permitted” method, such conditions are added to SQL queries so that entries that are forbidden for the current user did not affect the query result. In other words, when restrictions are imposed in the “permitted” mode, records forbidden to this user are considered to be missing, which is schematically shown in fig. 59:

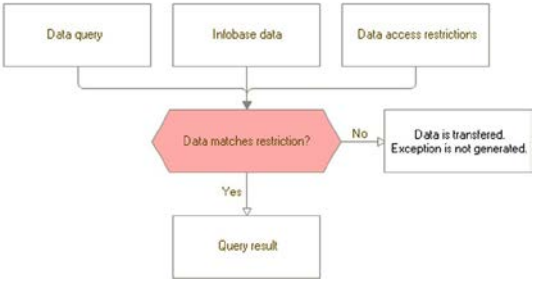


Fig. 59. "Permitted" method

5.5.4.8.5. Other objects related to data access restrictions

When developing configurations using data access restrictions, metadata objects such as session parameters, functional options and common modules with the Privileged check box may be useful.

Session parameters

Session parameters can be used in data access restrictions in the same manner as the query parameters can be used in a query.

Functional options

Parameter-independent functional options can be used in data access restrictions in the same manner as the query parameters can be used in a query.

Privileged common modules

If the Privileged check box is checked for the common module, the execution of the procedures and functions of this module acquires important specific features:

- Only the module that runs on the server can be privileged in the client/server mode of 1C:Enterprise.
- Execution of procedures and functions of the privileged module and everything, that is called from them, is performed when the feature of rights restriction to both metadata objects and data is disabled. Thus, any operation on any objects can be performed from the privileged module, even if the current user does not have the corresponding rights.

Privileged modules are intended to initially set the values of session parameters used in data access restrictions. The common modules can be also used for some integral operations with data by a user with limited rights. For example, if the user function includes entering and posting the documents, but the user must not have access to data that is affected by the posting of the document, the execution of the posting operation can be passed to a privileged module. This will allow the user to post the documents without giving them the rights to other information (e.g., registers).

Privileged mode

It is possible to programmatically set privileged mode when working with data. The programmatical setting of the privileged mode may be required in case of massive operations with the infobase data, and when it does not make sense to check the data access rights.

5.5.4.8.6. Preprocessor use

When editing the data access restriction text, it is possible to use preprocessor commands. The following commands are available:

```
#IF <Expression> #THEN
#ELSIF <Expression> #THEN
#ELSE
#ENDIF
```

`<Expression>` - arbitrary boolean expression in the 1C:Enterprise language, the result of which is of `Boolean` type. The expression may contain:

- comparison operations `<`, `>`, `<=`, `>=` , `=`, `<>`;
- logical operations `AND`, `Or`, `NOT`;
- session parameters - `&Parameter` syntax is used, where `Parameter` - is session parameter name.

If the result of the `#IF` or `#ELSEIF` command expression is `True`, the text following the `#THEN` keyword is included in the resulting access restriction command text. If the result of the expression is `False`, then the text following the `#THEN` keyword is not included in the access restriction command text. If none of the earlier conditions are met, the resulting access restriction command text will include the text following the `#ELSE` command.

NOTE. If the data access restriction text contains preprocessor commands, such restriction does not pass the syntax check during editing and cannot be changed using the wizard.

Example:

```
#IF &CurrentUser <> "Klimova" #THEN
<access restriction text>
#ENDIF
```

Here `CurrentUser` - is a session parameter of `CatalogRef.Users` type. This statement means that the condition for setting access restrictions will be checked for all users from the catalog, except the user Klimova.

5.5.4.8.7. Access restriction text templates

A role can contain a list of access restriction templates, which are described in the Restrictions templates role form tab. Access restriction templates can be also edited in the access restrictions and templates batch editor.
Each access restriction template has a name and a text. Template name must comply with the naming conventions of 1C:Enterprise.
Template text contains text section written in data access restriction language and can contain parameters marked with "#" sign.
The "#" sign can be followed by:

- One of the keywords:
  - `Parameter`, followed by the parameter number in the template in parentheses;
  - `CurrentTable` - indicates the insertion into the text of the full name of the table, for which the restriction is being generated;
  - `CurrentTableName` - indicates the insertion into the text of the full name of the table (as a string value, in quotes), which the instruction is applied to, in the current version of the 1C:Enterprise language;
  - `CurrentAccessRightName` - contains the name of the right, for which the current restriction is met: `ЧТЕНИЕ`/`READ`, `ДОБАВЛЕНИЕ`/`INSERT`, `ИЗМЕНЕНИЕ`/`UPDATE`, `УДАЛЕНИЕ`/`DELETE`;
- template parameter name - indicates insertion into the restriction text of the corresponding template parameter;
- "#" sign - indicates inserting of a single "#" into the text.

The access restriction expression may contain:

- Access restriction template, which is specified in the `#TemplateName("Template parameter value 1", "Template parameter value 2" ...)` format. Each template parameter is enclosed in double quotes. If required to specify a double quote character in the parameter text, two double quotes must be used.
- Function `StrContains(SearchWhere, SearchForWhat)`. The function is intended to search for `SearchWhere` string in

SearchForWhat string. Returns True if the string is found and returns False -otherwise.

- + operator for string concatenation.

For the ease of template text editing click the Set template text button in the role form on the Restriction templates tab. Enter the template text in the dialog box and click OK.

The 1C:Enterprise will check the syntax of template text, template usage and check the macro substitution of the role access restriction template texts in the query text.

Template macro substitution includes the following operations:

- replacing parameters in the template text with parameter values from the template use expression in the restriction text;
- replacing the template use expression in the query text with the resulting template text.

When calling the query wizard for a condition containing access restriction templates, a warning is displayed about replacing all the templates.

Examples of restriction templates are shown below:

Template name	Template
Template body	Total = #Parameter(1)
Usage	Where #Template("10")
Result	Where Total = 10

Template name	Template1(DocumentType)
Template body	DocumentType = #DocumentType
Usage	Where #Template1(""Invoice"")
Result	Where DocumentType = "Invoice"

Template name	Template2
Template body	DocumentType = #Parameter(1) ## #Parameter(2)
Usage	Where #Template2(""""Invoice", "1"""")
Result	Where DocumentType = "Invoice # 1"

Template name	Template3
Template body	DocumentType = #Parameter(3)
Usage	Where #Template3("","",""Invoice"")
Result	Where DocumentType = "Invoice"

5.5.4.8.8. General recommendations on rights restriction

In order to flexibly manage user data access in accordance with functions when setting data access restrictions, it is recommended to comply with the following principles:

- It is necessary to choose a set of information (may be dependent on the current user) for which preliminary preparation is appropriate. The selected information must, on the one hand, simplify data access restrictions as much as possible, and on the other hand, must not be too large. Distribute it to the session parameters.
- Set the values of the session parameters in the `SessionParametersSetting()` session module handler.
- Set access restrictions to the data for which this is justified (the data are secret or most important to preserve the 1C:Enterprise integrity). It must noted that the access restrictions setting can slow down any access to this data. The excessive complexity of the restrictions can cause a performance decrease.
- If it is necessary to ensure that a certain number of operations are performed on data by the user, for whom it is impractical to give full access to this data, pass these operations to privileged modules or explicitly turn privilege mode on and off at the appropriate sections of the program code.
- Data access during various checks performed by 1C:Enterprise when writing objects is executed in privileged mode. This allows not to disable rights restrictions at the record level for the corresponding fields, if the configuration operation with this data is planned **in the managed mode only**:
  - for catalogs when checking the parent, owner and uniqueness of the code;
  - for documents, business processes and tasks when checking the uniqueness of the number;
  - for exchange plans it is disabled when checking the uniqueness of the code;
  - for charts of accounts and carts of characteristic types when checking the parent and the uniqueness of the code.

When creating a data restriction query, some limitations and features should be noted:

- If data access restrictions are set for the object table and the data query uses joining with such table, then the use of object tabular section with the specified access restriction is not permitted in the connection condition (`BY` query section).
- If the query contains a table in which the query does not use any fields, then all the data access restrictions are imposed on this table. For example, `SELECT QUANTITY(*) FROM Catalog.Counterparties` query

will be executed with consideration of all the access restrictions, specified for the `Test` catalog. Restrictions are imposed "using OR logic". This means that all the records that are accessible by at least one condition will be available. If conditions are not specified for any fields, the query will be executed for all the records in the table.
If the query uses a top-level table, then the restrictions specified for the columns of nested tables are not imposed.
If the query uses a nested table, then restrictions are imposed on both nested and top-level tables. For example, `SELECT QUANTITY(*) FROM Catalog.Counterparties.Contracts` query will be executing with consideration of all the restrictions for `Counterparties` catalog and the restrictions relating to `Contracts` tabular section.

- If access to the fields necessary to retrieve the reference metadata object presentation is denied using data access restrictions or access to an object is denied at the level of access rights, then retrieving a presentation of such object does not affect the progress of the current transaction.

5.5.4.8.9. Data access restrictions wizard

To call the wizard go to edit mode and press the select button in the Data access restrictions table filed, in the Access restriction column, and click the Request wizard... button in the opened form.
The wizard form is displayed:

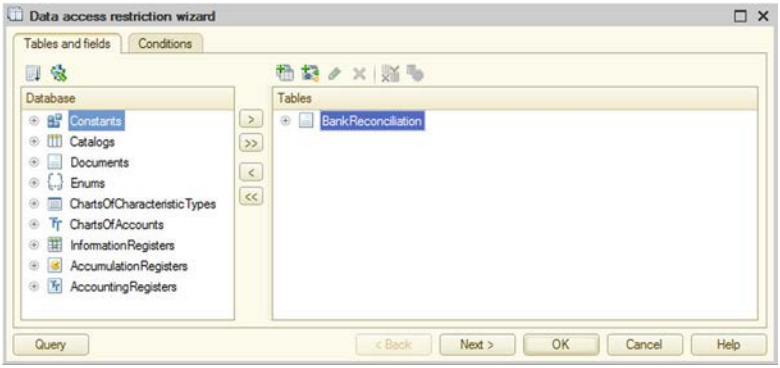


Fig. 60. “Tables and fields” tab of the restriction wizard

Creation of conditions for setting data access restrictions is performed using it.

In the Tables and fields tab, select the necessary objects in the Database list and move them to the Tables list. If several tables are specified, the Links tab is added to the wizard form.

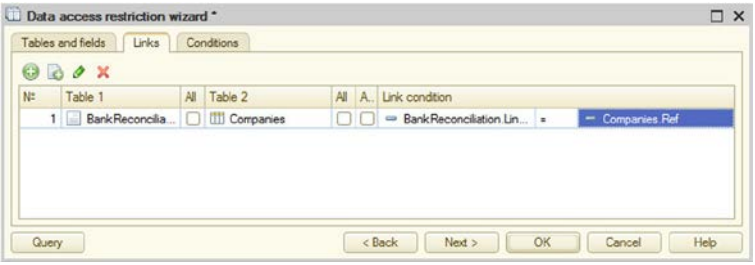


Fig. 61. "Links" tab of the restriction wizard

The Links tab enables you to specify the criteria for the links between the table fields. To enter a new condition, click the Add button and select one of the tables in the Table1 column. Select the table which fields are linked to the fields of the first table in the Table2 column. Controls that are used to create table link criteria are located below the criteria list.

If a simple condition type is selected, then the related fields of the specified tables are selected in Field1 and Field2 and the comparison condition is set. If the fields that are not to be compared are selected, then the row of the conditions list in the Linking condition column displays the following text: Incorrectly filled condition.

Specify the conditions using which the source data will be selected in the Conditions tab, if so required.

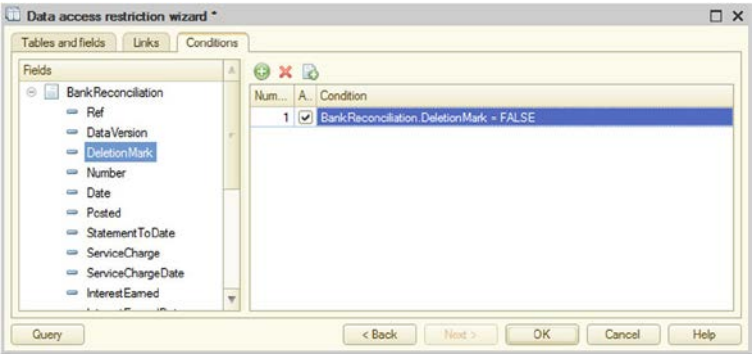


Fig. 62. "Conditions" tab of the restriction wizard

For each selected field, it is required to choose the type of condition and specify the parameter name. A session parameter is allowed to be used as parameter. Several conditions are allowed. In this case, the condition text is displayed in several lines in the Condition column of the condition table field.
You can review the query text anytime during query creation by clicking Query button.

5.5.4.8.10. Batch editing of access restrictions and templates

The batch access restrictions and templates editing mode is called using All access restrictions command from the context menu of the Roles branch. The displayed form contains two tabs: Access restriction and Restriction templates.

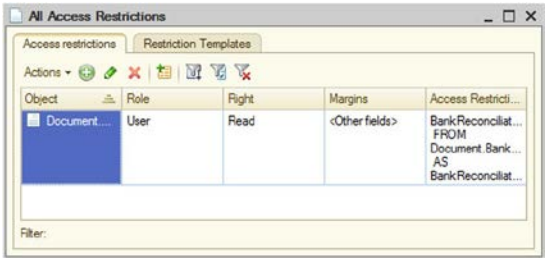


Fig. 63. All access restrictions and templates

In the Access restrictions tab it is possible view all the entered access restrictions in the common list (for all the roles, objects, rights, combinations of fields).
There is a possibility to add access restrictions for several roles, objects, rights and combinations of roles at once.

It is possible to filter the list by various criteria.

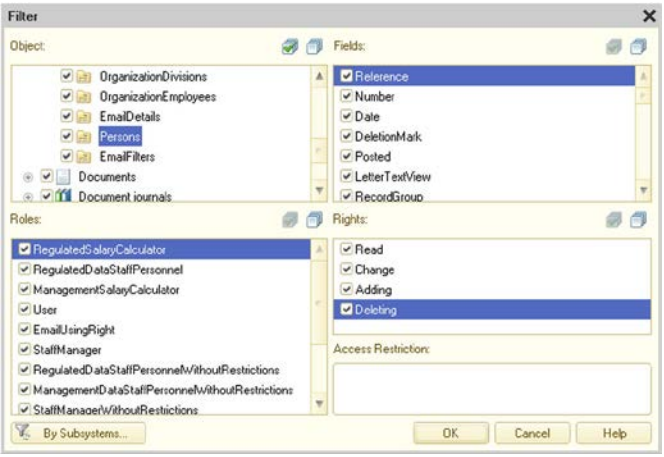


Fig. 64. Access restrictions filter

The batch editing mode allows removing the restrictions selected in the list.
It is possible to edit the selected restrictions. At the same time, it is possible to replace the field components and/or access restriction.
The batch editing mode also allows copying the selected restrictions to other roles.

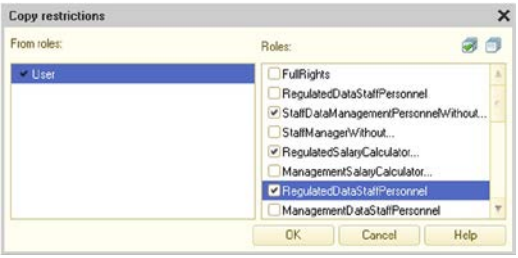


Fig. 65. Restrictions copying

In the Restriction templates tab, it is possible to see all the access restriction templates, available in the application, while the table only shows the first 10 lines of the template text itself, which end in the "…" character, if the template

text contains is more than 10 lines. The full template text will be displayed in the template editing window.

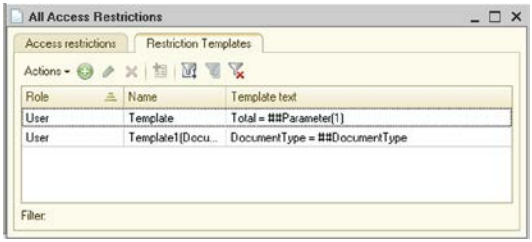


Fig. 66. All access restriction templates

It is possible to add an access restriction template to several roles at once.
It is possible to select the necessary templates using a set of criteria, as well as by the value of the current column.

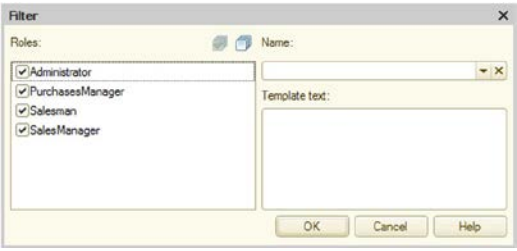


Fig. 67. Access restriction templates filter

If required, it is possible to copy one or more templates to other roles.

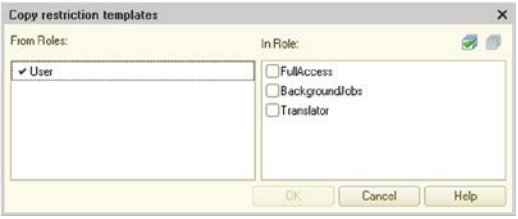


Fig. 68. Template copying

The editor also allows to edit the selected templates. While doing this, it is possible to change the template name and text itself.

5.5.5. Common attributes

Common attribute - is an attribute added to all or many configuration objects. Common attribute may be used in two scenarios:

- As a **common attribute**, i.e., to simplify specifying the attribute that is present in all or many configuration objects where this attribute retains its meaning and type. An arbitrary comment field in the application documents is an example of such common attribute use.
- As a component of the **data separation** - a special feature that allows dividing all the stored data, as well as the application operation into separate parts. In this case, **data separation** is enabled for the common attribute. An example of such use is the concept of a subscriber, when different data "owners" can operate independently in one physical infobase, while the users of such application will think, that there is nothing more in the infobase except "their" data.

In order to create a common attribute, it is necessary to create a Common attribute configuration object. This can be done in the Designer mode in a usual way, that is, select Common item in the configuration window, then Common attributes and add a new object.

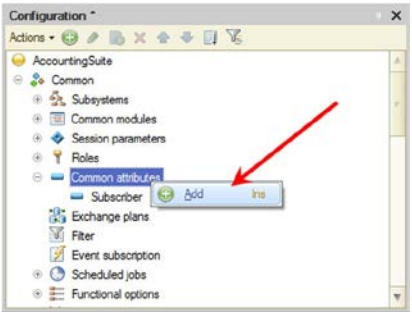


Fig. 69. Creating common attribute

As a result, the Common attribute configuration object will be created, which can be used to include the common attribute in the necessary configuration objects.

The common attribute mode of operation is set using the Data separation property. If this property is set to Do not use, then the created configuration object will be used only as an attribute, included in several (or all) configuration objects. If the property is set to Separate, the common attribute will be used as a data **separator**.

The list of configuration objects that include the common attribute is determined using the Components and AutoUse properties (or the corresponding tab in the Advanced window).

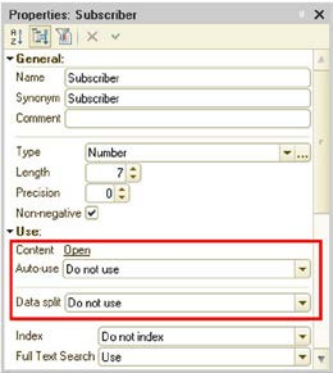


Fig. 70. Common attribute without separation

If the AutoUse property is set to Use, then the created common attribute will be automatically added to all existing configuration objects (for which it is possible to use common attributes), and will also be automatically added to all newly created configuration objects.
If the AutoUse property is set to Do not use, there will be no automatic adding of the attribute, and the Components property will be used to select the objects where the common attribute must be included.

This same property must be used if there are objects in the automatic use of the common attributes where the common attribute must not be added.

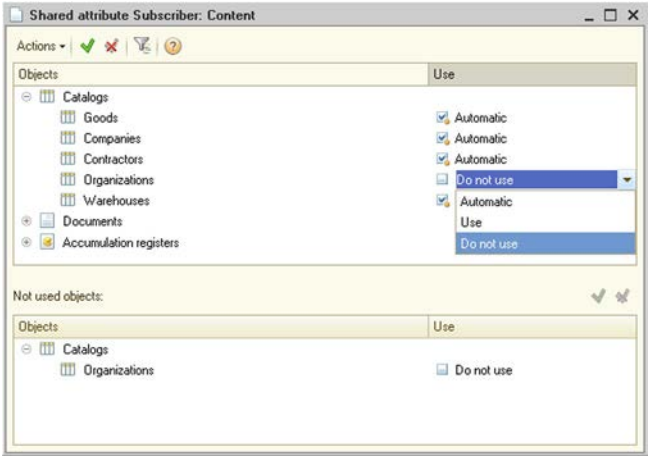


Fig. 71. Common attribute components

The window for editing the common attribute components is divided into two parts:

- the top section lists all the configuration objects that may be included in the common attribute;
- the bottom section - contains the objects which settings differ from the default settings determined by the value of the AutoUse property of the common attribute:
  - if the property is set to Use, the bottom section of the window contains a list of objects that are not included in the common attribute;
  - if the property is set to Do not use, the bottom section of the window contains a list of objects included in the common attribute.

Editing can be done both in the top and the bottom sections of the window, while the edited configuration object will move between the window sections, depending on the value in the Usage column.

For each configuration object, the Usage column can have one of three values:

- Automatically - means that the assignment of a configuration object to a common attribute depends on the value of the AutoUse property of the common attribute:
  - Use means that the configuration object is **included** in the common attribute;
  - Do not use means that the configuration object is **not incuded** in the common attribute.
- Use - means that the configuration object is **included** in the common attribute regardless of the value of the AutoUse property.

- Do not use - means that the configuration object is **not included** in the common attribute regardless of the value of the AutoUse property.

Thus, using the Components property editor, it is possible to, for example, selectively exclude some objects from the common attribute, despite the fact that the automatic use is set for it.

The common attribute (not in the data separation mode) may include the following configuration objects:

- Catalogs
- Documents
- sequences
- Document journals
- Charts of characteristic types
- Charts of accounts
- Charts of calculation types
- recalculation,
- Business processes
- Tasks
- Information registers
- Accumulation registers
- Accounting registers
- Calculation registers
- Exchange plans
- External data sources

When writing a document, the journal common attribute is assigned the value of the common attribute of the document or the NULL value if the document is included in the common attribute.

Common attribute can be used in data access restrictions. It makes sense to include external data sources in the common attribute if the common attribute is a separator. The peculiarities of using the external data source included in the separator can be found in the corresponding section.

Common attribute **may be of composite type**.

TIP. Using common attributes to describe data that is part of the business logic of specific objects should be avoided.

Common attributes are displayed in the form editor when editing the object forms included in the common attribute, and can be placed on the form.

5.5.6. Exchange plans

Exchange plan is used to implement data exchange features. Exchange plan:

- contains information about nodes that can participate in data exchange;
- determines the data to be exchanged;

- indicates whether the distributed infobase functionality must be used during the exchange.

One application can have several exchange plans, each of which may describe its own data exchange procedure. For example, if data is exchanged with remote warehouses and remote offices, then there will most likely be two exchange plans (one - for data exchange with warehouses, the other - for data exchange with offices), since the data that is exchanged with warehouses will be significantly “narrower” than the data intended for exchange with the offices.

5.5.7. Filter criteria

Filter criteria - is one of the components of the data filtering functionality. Using these, the 1C:Enterprise configuration specialist creates predefined filtering rules. In 1C:Enterprise mode, these rules will be used to filter the information in lists.

It is necessary to specify a name, a synonym and a comment in the Filter criteria object editing window.

The type of filter criteria can be any of the standard types or defined as the configuration tree objects. It is allowed to include attributes with a composite type, specified as `CatalogRef`, `DocumentRef`, etc., as well as attributes with a composite type defined by the chart of characteristic types (`Characteristic...`).

When creating a type of filter criteria, it is necessary to specify the types which are used for filtering. Based on this type in the Components tab, the configuration objects composition will be generated, which contain data of type included in the filter criteria type. It is necessary to set a mark in the list for those attributes, by which this filtering will be performed.

For the filter criterion to fulfill its functions, a list must be created for it in the Designer (the Components tab) consisting of the catalog and document attributes. There are practically no restrictions for the components of the list: for example, in contrast to the journal column, several attributes of one document and attributes of the document tabular section can be selected for the filter criterion.

There may be an arbitrary number of filter criteria, and each criterion can have several forms for presenting the filter results. This feature is useful when searching for various information. For example, it is required to select all the documents where a certain counterparty (in attributes and tabular sections) is used. In this case, it is possible to consider other conditions for the data filtering (for example, the search is performed only among the posted documents or within a certain timeframe, etc.).

NOTE. When opening the filter form, it is necessary to pass the `Filter` parameter to it with the set filter value (`Value` element).

The filter criterion can have an arbitrary number of forms for visual presentation of the filter results. For prompt retrieving the filter information, the form call can be placed in the user menu or on the toolbar.
If there are several forms of the filter criterion, then the Main form property indicates the form that will be called by default.
If there are several subsystems defined in the configuration, it is necessary to select the one which this filter criterion belongs to. Several different subsystems can be specified.
To call the filter criteria form, 1C:Enterprise system places the corresponding command in the form navigation panel.

5.5.8. Event subscriptions

Event subscriptions allow to assign event handlers to a single object or group of objects of the 1C:Enterprise language.
When adding a new event subscription, in addition to the common configuration object properties, it is necessary to specify the event source, the event, the handler of which is assigned, and the procedure that handles this event.
Applied objects, sets of register records, and configuration object managers can be event sources. Both multiple selection of objects that are event providers, and the selection of all objects of the same type (for example, all the documents) is allowed.
The event is selected from the drop-down list, where the list contains those events that are present in all the selected objects. If there are no such events, the list will be empty.
The event handler is selected in a window containing procedures that can be assigned as an event handler. Such procedures must meet the following requirements:

- The procedure must be located in a common module;
- The common module that contains the procedure must have the following properties set:
  - Global box is unchecked;
  - Client (ordinary application) box is checked;
  - Client (managed application) box is unchecked;
  - Server box is checked;
  - External connection box is checked.
- The procedure parameters number must exceed the selected event handler parameters number by one. This is due to the fact that in addition to the parameters that are passed by 1C:Enterprise to the event handling method, the source object of this event is also passed.

When the specified event occurs, the following sequence of actions is performed:

- First, the event is handled in the object itself and the event handler defined in the object or record set module is called. If the configuration extensions

are connected to the application, the program code extending the event handlers is also executed at this stage.

- If the `Fail` parameter is set to `True` or an exception is thrown during the execution of the handler, the action is aborted.
- Then the external handlers assigned to this event are called in an arbitrary order.
- If the `Fail` parameter is set to `True` or an exception is thrown during the execution of the assigned handler, the action is aborted.

As a source, the object itself (record set) that caused the event is passed to the assigned handler.

The assigned event handlers are called in the same context as the action that triggered the event. If the execution of the assigned handler needs to be passed to the server, the common module procedure executed on the server must be called in the handler code.

The event handlers assignment is also available using the 1C:Enterprise language. For this the `AddHandler` and `RemoveHandler` operators are used.

Objects that can be event sources have the `AdditionalProperties` property of the `Structure` type, which allows to store information between event calls, for example, if this object is new or old one.

5.5.9. Scheduled jobs

5.5.9.1. Main features of the job functionality

Main features of the job functionality are:

- determining scheduled procedures at the 1C:Enterprise programming stage;
- performing the specified actions as per schedule;
- calling the specified procedure or function asynchronously, i.e., without waiting for it to complete;
- monitoring the job progress;
- job management (cancelling, blocking of execution, etc.);
- possibility to wait for the completion of one or more jobs.

5.5.9.2. Background jobs

The background jobs feature is implemented using the 1C:Enterprise language. Background jobs are intended to perform application jobs asynchronously. These can generate child background jobs, for example, for paralleling complex computations across different working servers in a cluster in a client/server mode.

It is possible to limit the execution of background jobs with the same methods based on a certain applied characteristic. Programmatic creation and management of background jobs is possible from any user connection to the 1C:Enterprise infobase. The background job is run as the user who created it.

5.5.9.3. Scheduled jobs

Scheduled jobs are an integral part of a specific application and are described at the configuration stage.

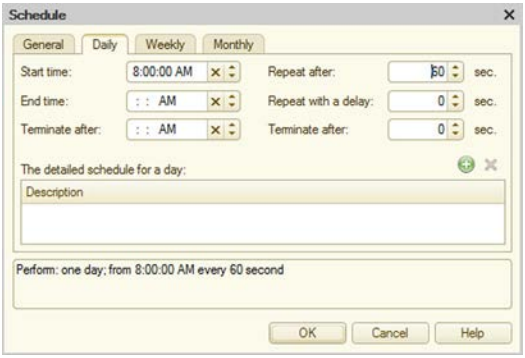


Fig. 72. Background jobs schedule

For each scheduled job a schedule can be set, according to which the scheduled job will be automatically started. The 1C:Enterprise supports one-time and periodic schedules. It is possible to set the start and end dates, daily, weekly and monthly schedules. The schedule can be set both at the configuration stage and at the execution stage (in the 1C:Enterprise mode).

In the process of starting, a scheduled job generates a background job, which performs the actual processing. The scheduled job can be executed as the specified user and has the possibility to be restarted (for example, in the event of an unexpected shutdown).

In the client/server administration utility the automatic execution of scheduled jobs can be prohibited for a specific infobase.

5.5.9.4. Peculiarities of the scheduled jobs execution

In the client/server mode, the scheduled jobs are started as per schedule by the cluster manager. Thus, even if no client connection is established to the infobase, scheduled jobs will be executed (provided that they are not prohibited for a particular infobase).

5.5.10. Functional options and functional option parameters

5.5.10.1. Purpose

Functional options allow the developer to describe the features of the application, which can be quickly turned on or off during the deployment phase and/or during 1C:Enterprise operation. For example, the ability to process additional properties of goods can be distinguished into a separate functional option. Then, if this feature is disabled, all the related features (with additional properties of the goods) will be "lost" the application interface.

The 1C:Enterprise is able to automatically consider the state of the settings made - to hide the disabled features, making the application interface more clear and understandable for the user.

During the development, situations arise when the value of a functional option must depend on certain parameters, for example, not all the companies maintain currency accounting. To implement this dependency, the Functional option parameters are used - which are the objects that set parameters of the functional options.

5.5.10.2. What do functional options affect

5.5.10.2.1. General information

The functional options may affect:

- **User interface** - when any functional options are turned off, the 1C:Enterprise system hides all items related to it in the user interface. This affects the following interface items:
  - global command interface;
  - form attributes (including form attribute columns of `ValuesTable` or `ValueTree` type);
  - form commands;
  - reports implemented using the data composition system.

IMPORTANT! If the client application uses the file mode of the infobase via the web server, changing the functional option will only change the user interface after restarting the web server (restarting the client application will not cause user interface change).

- **Algorithms written in the 1C:Enterprise language** - there is a possibility to retrieve the functional option values from 1C:Enterprise language and use them under various conditions, for example, to reduce the amount of computation.

IMPORTANT! Functional options and their parameters do not affect the database components. All the tables and fields are present in the database, regardless of the state of the functional options.

5.5.10.2.2. Global command interface

The effect of the functional options on the global command interface is that 1C:Enterprise hides the commands of all the objects related to the disabled options. For example, if the `Purchases` functional option value is `False`, the commands to open the `Purchases` section, to create the `GoodsArrival` document, to open the `GoodsArrival` list, etc., will be hidden.
In turn, the `Purchases` option may consider the functional option parameter value, for example, `Organization`. By changing the value of this parameter using the 1C:Enterprise language methods, it is possible to change the state of the functional option, and, consequently, the visibility of the interface item.
It is also necessary to consider the following peculiarities of the command interface generation:

- The command will be excluded from the command interface in the event that the attribute, which is a parameter of the command, is disabled by the functional option.
- The command will be excluded from the command interface if the command parameter type is disabled by the functional option. If the command parameter type is composite, the command becomes unavailable when all the parameter types are disabled.

5.5.10.2.3. Forms

In the form, the functional options can affect form attributes and commands and (as a result) change the visibility of the associated form items (fields and columns - for form attributes, buttons - for form commands). When developing a form, it is necessary to consider the following peculiarities of 1C:Enterprise behaviour:

- The main attribute of the managed form of the `<Type>Object` type (`CatalogObject`, `DocumentObject`, etc.) will be disabled if the corresponding configuration object is disabled by the functional option. Only those functional options that have no parameters are analyzed.
- The main attribute of the managed form of the `DynamicList` type will be disabled if the functional option disables the configuration object that is specified as the main table of the dynamic list. Only those functional options that have no parameters are analyzed.

- Dynamic list that displays the document journal does not consider the fact that some of the document types in this journal are disabled by the functional option. The dynamic list will always display all the documents in the journal.
- The reference type form attribute is disabled if the configuration object forming this type is disabled by the functional option. The composite type form attribute is disabled if the functional options disable all the component types.
- A form attribute of the `<Type>Object` type (including the main form attribute) is disabled if the configuration object forming this type is disabled by the functional option. Only those functional options that have no parameters are analyzed.
- The form table will be disabled if it displays the data of the form attribute disabled by the functional option.
- There are no types in the type selection dialog box (for example, for input fields associated with the composite type attributes), if the configuration objects that form these types are disabled by the functional option. Information about types that are disabled by functional options is cached on the client side and cleared after 20 minutes or during the call of `RefreshInterface()` method.

IMPORTANT! Unlike the command interface, the functional options parameter values are set only for a specific form instance.

5.5.10.2.4. Data composition system

The data composition system is mainly used for reports generating. Functional options affect the data that is displayed in the report, and the report settings available to the user. For example, if the Foreign currency accounting functional option is disabled, then the report displaying the Goods arrival documents register will not include the Currency and Currency amount columns, and the settings will not allow to filter, group, sort, etc. the `Currency` field.

5.5.10.2.5. Characteristics

Functional options affect the visibility of form fields that display the object characteristic value. For this purpose, it is necessary to include an attribute that stores the value of the characteristic in the functional option.
Consider an example. Characteristics are used for the `Goods` catalog, types of characteristics are stored in `Characteristics` chart of characteristic types, and values - as a `CharacteristicValues` information register resource.

The resource is included in the `CharacteristicsAccounting` functional option.

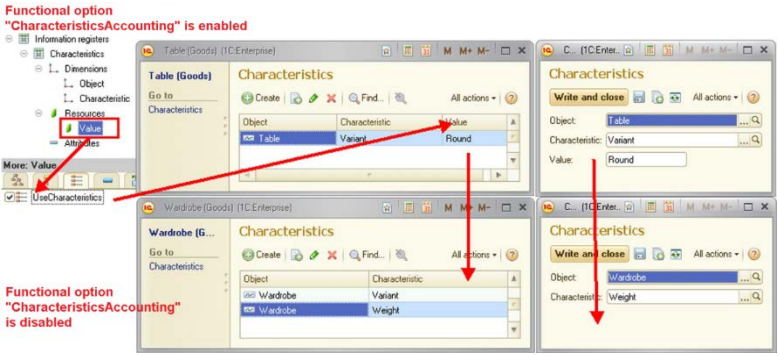


Fig. 73. Effect of the functional options on characteristics

If the `CharacteristicsAccounting` functional option is disabled, , the visibility of the fields (the Value column and the Value field), displaying the values of the characteristics in the forms is also disabled, as shown in fig. 73.

5.5.10.3. General operation procedure

The functional options feature includes two types of metadata objects: Functional option and Functional option parameters.

The functional option is a metadata object that can directly affect the application interface components (if the functional option stores its value in a `Boolean` attribute). Using objects of this type, it is possible to hide items that are related to unavailable functionality. For example, Currency accounting option may hide the `Currencies` catalog, `Currency` filed from the documents, Currency amount column from the reports. The source of the functional option value is the metadata object selected as the `Storage` property, for example, it can be a constant.

If the functional option value is stored in the catalog attribute or the information register resource, additional information is required, which indicates how to select the option value. For this purpose, a separate metadata object - Functional option parameters is provided.

It can be said that the functional option parameters are the coordinate axes of the functional option value space. At the same time one functional option parameter can determine the value of "its" coordinate axis simultaneously for a variety of functional options.

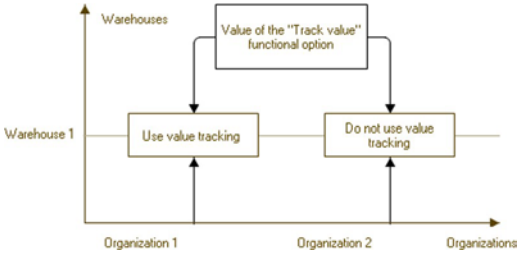


Fig. 74. Parameterizable functional option

Consider an example: let's assume that the sum accounting depends on the warehouse owned by a particular organization (see fig. 74). In our infobase it is possible to keep records on behalf of various organizations and in different warehouses.

To store the values of the functional options, let's create an information register, where the dimensions (coordinate axes) will be:

- `Organization` (of the corresponding type);
- `Warehouse` (of the corresponding type).

Information register resource will be the value of the sum accounting functional option.

Then the general configuration structure will be as follows:

- Information register `SumAccounting`:
  - dimension `Organization`;
  - dimension `Warehouse`;
  - `SumAccounting` resource of `Boolean` type.
- `Organization` functional option parameter. `Usage` property indicates the `Organization` dimension of the `SumAccounting` information register.
- `Warehouse` functional option parameter. `Usage` property indicate the `Warehouse` dimension of `SumAccounting` information register.
- `SumAccounting` functional option. `Storage` property indicates the `SumAccounting` resource of the `SumAccounting` information register.

As a result, in order to determine the need for sum accounting, it is necessary to specify the values of the functional option parameters (`Organization` and `Warehouse`) in each specific case and retrieve the functional option value.

Thus, in the example shown in fig. 74, the sum accounting is allowed for Organization 1 and Warehouse 1, and sum accounting is prohibited for Organization 2 and Warehouse 1.

5.5.10.4. Interaction with other objects

Functional options can be assigned to the following configuration objects:

- Subsystems
- Common commands
- Common forms
- Constants
- Filter criteria
- Catalog
- Document
- Journal
- Chart of accounts
- Chart of characteristic types
- Chart of calculation types
- Business process
- Task
- Exchange plans
- Report
- Data processor
- Accumulation register
- Information register
- Accounting register
- Calculation register
- Command
- Metadata object attribute
- Tabular section
- Tabular section attribute
- Accounting flag
- Extra dimension accounting flag
- Addressing attributes
- Register dimension
- Register resource

Functional options can also affect the visibility of form items.

5.5.10.5. Creating

5.5.10.5.1. Functional option creation

In order to create a functional option, it is necessary to create a Functional option configuration object. This can be done in the Designer mode in a usual way, that is, select Common item in the configuration window, then Functional options and add a new object.

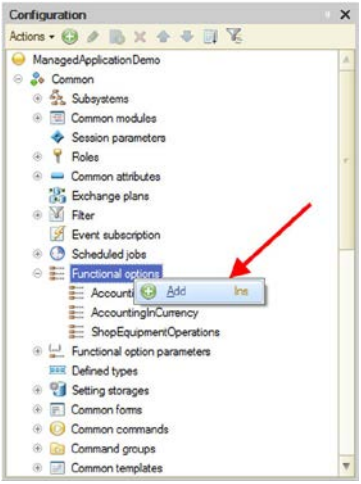


Fig. 75. Functional option creation

As a result, a Functional option configuration object will be created that can be used to assign functional options to other metadata objects.

Fig. 76. Functional option value storage

In addition to the name, the object has a mandatory property - Storage. In the editor, it is possible to select one of the objects for it, which will be the option value source. The list of available objects includes:

- constants

- catalog attributes,
- information register resources.

There is no restriction on the option value source type, but only those functional options that store their values in the attributes of Boolean type are suitable for controlling the interface. Functional option values with other types are available for analysis in the 1C:Enterprise language only.

The Privileged mode upon retrieval property is responsible for the method of retrieving (and caching) the functional option value.

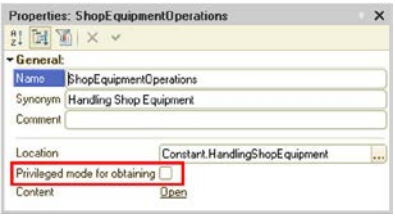


Fig. 77. Privileged mode when retrieving the functional option value

If this property is set, the functional option value is retrieved in the privileged mode. The retrieved value is cached for all the sessions related to this infobase.

If the Privileged mode upon retrieval property is reset, the functional option value is retrieved in normal mode. Caching is performed for the current session. Both the value (if it was retrieved) and the value retrieval impossibility flag (if the value could not be retrieved) are cached.

The cache is reset when the session parameters change.

TIP. It is recommended to set the Privileged mode upon retrieval property for all cases when the functional option value does not contain confidential information.

5.5.10.5.2. Creating functional option parameter

In order to create a functional option parameter, it is necessary to create a Functional option parameters configuration object. This can be done in the Designer mode in a usual way, that is, select Common item in the configuration window, then Functional option parameters and add a new object.

In addition to a name, the parameter has a mandatory Usage property. It contains a set of objects which values will determine how the functional option value will be selected. The list of available objects includes catalogs and information register dimensions. For each functional option parameter in this list, it is possible to select one catalog (from the entire list of catalogs) and one dimension for each information register.

IMPORTANT! Using the same metadata object in several functional option parameters is not possible.

5.5.10.6. Usage

5.5.10.6.1. Assigning to metadata objects

A metadata object (for example, a catalog) can be assigned to one or several functional options. To do this, use the Functional options property, which contains references to the functional options created in the configuration.

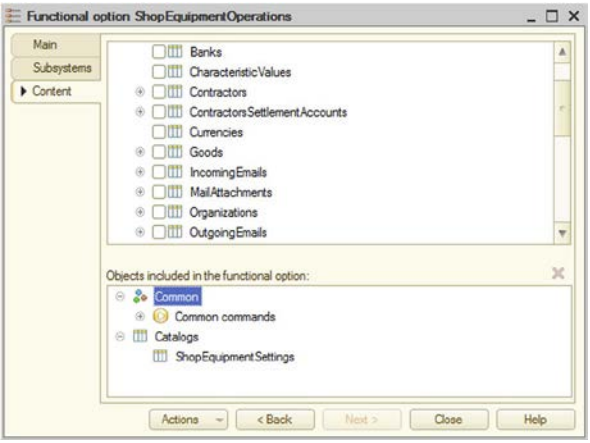


Fig. 78. Assigning the functional option to an object

The list of available options is limited only by those options which an object with the `Boolean` value type is assigned to in the `Storage` property.

IMPORTANT! If no functional option is assigned to an object, it is always considered visible. Otherwise, the object is considered visible if at least one of the functional options assigned to it is enabled (i.e., the functional options are combined “using OR logic”).

5.5.10.6.2. Assigning to form attributes and commands

Objects belonging to the form (Attributes and Commands) can also be used in the functional options functionality.

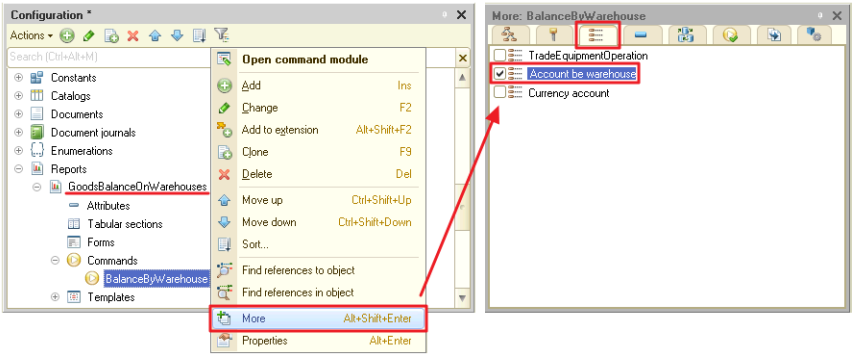


Fig. 79. Assigning the functional option to a command

This can be done in the form editor by setting the Functional options property for the required object.
The state of the functional options will affect the display of form objects in the same way as in case with metadata objects. For example, in case of a command that has been disabled using the functional option, all the associated buttons will be removed.
If no functional option is assigned to a form attribute or command, then a form attribute or command is always considered visible. Otherwise, the form attribute or command is considered visible if at least one of the functional options assigned to it is enabled.

5.5.10.6.3. Usage in the data access restriction functionality

In terms of the data access restriction functionality. Functional options can be used in the same way as in the Session parameters. It is permissible to use only parameter-independent options, that is, those that are bound to constants.

IMPORTANT! The 1C:Enterprise controls the uniqueness of names between session parameters and functional options.

5.5.10.6.4. Functional option value determination

The functional option value is determined by the object specified in the `Storage` property. In the case of a constant, its value is used. For the option

associated with the catalog attribute or information register resource, - the values are stored in these objects. In order to find a specific object that stores the functional option value, additional information is required - a set of the functional options parameter values.
If the option is stored in the catalog attribute, the parameter must contain a reference to a specific catalog item. When retrieving the functional option value, the deletion mark flag value of the catalog item that stores the functional option value is ignored. If the option is stored in the information register resource, the values of all the register dimensions must be specified. In this case, each dimension must be characterized by its own parameter.
If not all parameters are specified for a functional option of a `Boolean` type, then the addition of all the values with unset parameters “using OR logic” is performed. For example, if the functional option is stored in the information register with the `Organization` and `Warehouse` dimensions and only the `Organization` dimension is set, then the functional option value will be `True`, if at least one of the warehouses listed in the `Warehouse` dimension, has `True` as the functional option value .
For a functional option that has a type other than `Boolean`, the situation with not fully specified parameters causes generation of an exception.
The 1C:Enterprise language methods allow retrieving the option value, depending on the parameters passed as well as on the parameters set for the command interface or a specific form. In case when the value of an object specified in the `Storage` functional option property is changed in a transaction, the actual value of the functional option will be changed only after the transaction is completed. As long as the transaction is open - the functional option value will be equal to the value in effect at the time of the transaction start.
If the functional option is bound to the periodic information register resource, 1C:Enterprise uses a slice last to retrieve the option value. If it is required to retrieve the option value for any other date, it is necessary to specify a value for the `Period` (`Period`) functional option parameter, which is of the `Date` type that will be used as the date of slice. This parameter does not need to be created in the metadata. It is provided by 1C:Enterprise automatically.
When using parameterized functional options, the following peculiarities of behaviour must be considered:

- In the list forms, the attribute column associated with the parameterized functional option will be displayed if at least one enabled value of this functional option is stored in the infobase.
- If it is required that the attributes associated with the functional options are disabled by default when opening the form, then it is necessary to set the values of these parameters to values that are absent in the infobase (for catalogs - a null reference, for information registers - dimension values, for which no records exist). In this case, the functional option will be `False`.

- In the event that a reference to a group is specified as a parameter (if the functional option parameter type allows to create the groups), and not a reference to an element, the 1C:Enterprise behavior will be as follows:
  - if the attribute which stores the functional option value is used for both element and group, then the functional option value will be determined by the value of this attribute.
  - if the attribute which stores the functional option value is not used for a group, then retrieving the functional option value using the `GetFunctionalOption()`, `GetFormFunctionalOption()` and `GetInterfaceFunctionalOption()` method, the `NULL` value is returned. If the functional option parameterized by this value affects the user interface, 1C:Enterprise will perceive it as disabled one (the functional option will be `False`).
- For command-generating metadata objects, it is possible to set a binding to a parameterized functional option. In the command interface the commands of such objects will be displayed only if there is at least one combination of functional option parameters, where the functional option value is `True`. However, using the `SetInterfaceFunctionalOptionParameters()` method, it is possible to set specific functional option parameter values, and the visibility of commands will be determined by the specified parameters.
- The dynamic list automatically uses the functional options used by the form. If the attributes that are used in the dynamic list query are disabled for a set combination of functional option parameters, the data for them will not be filtered and displayed in the dynamic list, and the attribute will be removed from the lists of available attributes in the dynamic list data display settings dialog box (in 1C:Enterprise mode).

5.5.10.7. Managing functional options in the 1C:Enterprise language

The `GetFunctionalOption()` and `GetInterfaceFunctionalOption()` global context methods return the functional option value. The difference between these two consists in the fact that the first method allows to specify a set of functional option parameters, and the second one -returns the functional option value based on the parameters specified for the command interface. The form has its own method, which returns the option value for the parameters specified within the form, - `GetFormFunctionalOption()`.

To update the global command interface, it is necessary to explicitly call the `SetInterfaceFunctionalOptionParameters()` method. The command interface will be updated to reflect the new state of the functional options.

NOTE. If the functional option value changes in the database, then global command interface and the forms open at this time do not update automatically. To do this, use the `RefreshInterface()` method after writing the functional option values to the database.

It is worth noting that the setting the functional option parameters (and running the `RefreshInterface()` method) causes the following consequences:

- for each form, closing of all the auxiliary forms is called (with the call of the corresponding handlers);
- forms that refuse to close are not closed;
- main form item components are updated;
- if the main form was active at the time of interface update, the main form is displayed in accordance with the new item components;
- if the auxiliary form was active at the time of interface update, then:
  - the command will be executed to open the auxiliary form, if this form is available after the interface update;
  - otherwise, the item components of the main form are updated and the form is displayed;
- if auxiliary form opened with the help of a command, not related to the form navigation panel, was active at the time of interface update, the item components of the main form will be updated and displayed instead of this form.

In order to update a specific form, it is necessary either to re-open it or to call the `SetFormFunctionalOptionParameters()` method, while the above sequence of actions is applicable only for the form in the context of which setting of the form functional option parameters was called.

The parameters do not have to be specified all at once, it is possible to change the value of a specific parameter or a set of parameters selectively. But it is more efficient to perform the batch setting of values by one call.

To retrieve the parameter values it is necessary to call the corresponding function (`GetInterfaceFunctionalOptionParameters()` or `GetFormFunctionalOptionParameters()`), which will return the set parameters as a structure, where the key will be the parameter name.

When opening, the form automatically uses the functional option parameters set for the command interface.

5.5.11. Defined types

Defined type - is a special configuration object that is intended to simplify and enhance the ease of subsystem deployment when developing an application. As an example, a versatile subsystem for storing contact details may be considered. The structure of this subsystem includes a information register that stores this

data. There is a dimension in this register that indicates for which object the contact details are stored. Counterparty, own organization, physical entity, etc. may be the dimension value. Where the type (with the `CatalogOfOrganizationsAndPhysicalEntities` name), describing this storage object, can be used not only as the register dimension type. During the application deployment, the components of this type may change - new catalogs can be added or “unnecessary” types can be excluded from it. If instead of the `CatalogOfOrganizationsAndPhysicalEntities` type a composite type is used in every position where such an entity is used, it is necessary to change the type components everywhere. It is difficult and there is a possibility to forget about any attribute with the necessary composite type. If a defined type is used, then during application finalizing, it will only be necessary to change the defined type components (and, of course, all the fragments of the application code where this type is used). All the attributes for which the `CatalogOfOrganizationsAndPhysicalEntities` type is indicated will be changed automatically.

Defined type has the following features:

- May be composite;
- Cannot be used to indicate the chart of characteristic types value type;
- Cannot be included in the composite data type of another attribute;
- A defined type cannot be included in another defined type;
- The defined type cannot be included in the following objects type:
  - Chart of accounts accounting flag;
  - Chart of accounts extra dimension accounting flag;
  - Numerator number type.

A defined type can include the following types:

- Reference types: `CatalogRef`, `DocumentRef`, `EnumRef`, `ChartOfCharacteristicTypesRef`, `ChartOfAccountsRef`, `ChartOfCalculationTypesRef`, `BusinessProcessRef`, `BusinessProcessRoutePointRef`, `TaskRef`, `ExchangePlanRef`, `ExternalDataSourceCubeDimensionTableRef`
- Object types: `CatalogObject`, `DocumentObject`, `ChartOfCharacteristicTypesObject`, `ChartOfAccountsObject`, `ChartOfCalculationTypesObject`, `ReportObject`, `DataProcessorObject`, `BusinessProcessObject`, `TaskObject`, `ExchangePlanObject`
- Record sets and related types: `InformationRegisterRecordSet`, `AccumulationRegisterRecordSet`, `AccountingRegisterRecordSet`, `CalculationRegisterRecordSet`,

RecalculationRecordSet, ExternalDataSourceCubeRecordSet, SequenceRecordSet, ExternalDataSourceCubeRecordManager, ExternalDataSourceCubeDimensionTableObject

- Object managers: CatalogManager, DocumentManager, EnumManager, ChartOfCharacteristicTypesManager, ChartOfAccountsManager, ChartOfCalculationTypesManager, ReportManager, DataProcessorManager, BusinessProcessManager, TaskManager, ExchangePlanManager, DocumentJournalManager, InformationRegisterManager, AccumulationRegisterManager, AccountingRegisterManager, CalculationRegisterManager, ConstantValueManager, FilterCriterionManager, ExternalDataSourceTableManager, ExternalDataSourceCubeManager, SettingsStorageManager, ExternalDataSourceCubeDimensionTableManager, SequenceManager
- System enumerations: AccumulationRecordType, AccountType, AccountingRecordType, HorizontalAlign, VerticalAlign, SizeChangeMode, AnalysisDataType, ClusterizationMethod, DataAnalysisTimeIntervalUnitType, DataAnalysisResultTableFillType, DataAnalysisNumericValueUseType, AssociationRulesDataSourceType, DataAnalysisColumnTypeDecisionTree, DataAnalysisColumnTypeClusterization, DataAnalysisColumnTypeSummaryStatistics, DataAnalysisColumnTypeAssociationRules, DataAnalysisColumnTypeSequentialPatterns, PredictionModelColumnType, DataAnalysisDistanceMetricType, AssociationRulesPruneType, DataAnalysisStandardizationType, DecisionTreeSimplificationType, DataAnalysisAssociationRulesOrderType, DataAnalysisSequentialPatternsOrderType;

- Other types: `Custom`, `Number`, `String`, `Data`, `Boolean`, `ValueStorage`, `UUID`, `DynamicList`, `ValuesTable`, `ValueTree`, `ValueList`, `FormattedString`, `BinaryData`, `SpreadsheetDocument`, `FormattedDocument`, `TextDocument`, `NULL`, `FixedArray`, `FixedMap`, `FixedStructure`, `DataCompositionSettingsComposer`, `GraphicalSchema`, `GeographicalSchema`, `Chart`, `GanttChart`, `Dendrogram`, `TypeDescription`, `Image`, `Color`, `Font`, `StandardBeginningDate`, `StandardPeriod`;

When performing a configuration verification or during a check before updating the database configuration it is checked if the defined type components is valid at each specific place of use. Thus, the same defined type can be used as the type of one configuration object and cannot be - used for another one.

To describe the type in the 1C:Enterprise language, use the `DefinedType.<DefinedTypeName>` type statement.

NOTE. If it is necessary to use an application with earlier 1C:Enterprise versions (earlier than version 8.3.3), then all the defined types must be removed from the configuration.

5.5.12. Settings storages

To store information about user settings that must be saved between sessions, the platform has settings storages implemented.

There are two types of settings storages:

- **Standard storage** - is the storage used by the 1C:Enterprise by default and storing data in the infobase system tables.
- **Settings storages** - special metadata objects that describe data storing in a certain infobase object. For example, the operations with settings that are stored in the catalog can be described in this object.

The platform uses the following storages:

- **System storage** - 1C:Enterprise saves all the possible settings that are required for operation, to this storage. These settings include form size settings, tabular document print settings, etc. Standard settings storage is always used as a system settings storage. That is, the data of the system storage is always stored in the infobase system table.
- **Common settings storage** - this storage stores various application settings. The platform does not write any data to this storage automatically. The developer has to implement this storage using 1C:Enterprise language to save/restore applied user settings.
- **Reports user settings storage** - this storage stores user settings for reports.
- **Report option storage** - this storage stores report options.

- **Form data settings storage** - this storage stores form data. This storage can be used, for example, to save data processing attributes. At the same time it is possible to select an individual storage for each report and data processor.
- **Dynamic list user settings storage** - this storage stores user settings of dynamic list.

When developing a configuration, it is possible to define own settings storages for all storages, except for system storage. To do this, it is necessary to create a settings storage object in the corresponding branch of the metadata tree and then specify it in the desired configuration property. The Configuration object properties have the same names as the above storages.

Data storages can be stored both in the infobase system table, and in some special infobase object, for example, in a catalog or information register. For example, it is possible to create a settings storage object in the configuration and specify in the configuration property that this storage must be used to store report settings. Thus, report settings will not be saved to the system table, but in a certain object, for example, in a catalog, which makes it possible to arrange operation with uniform report settings, implement the system of rights, exchange settings, etc.

It makes sense to create own storage in cases when a special settings storage structure is needed, special settings management features are required, settings exchange within a distributed database is needed and other similar cases.

NOTE. To store the settings, it is recommended to select such objects for which 1C:Enterprise supports the identification method, where the identifying attribute can be converted to a string and back without losing data. A catalog and a standard `Code` attribute that is unique throughout the catalog, may be given as an example.

5.5.12.1. General principles of the settings storage operation

The `SettingsStorage` metadata object is intended to provide the storage of application configuration settings. Due to the implementation of event handlers and creation of object forms, the settings processing functionality is modified in such a way that the location of the settings storage (instead of system tables, configuration objects created by the developer are used) and visual settings processing are changed.

An arbitrary number of settings storages can be defined in the configuration.

The settings storage can be used for both program operations and for program and interactive operations. In the former case, to provide the necessary functionality, a mandatory implementation of the `SettingsStorage` object moduly by the handler is required:

- `SaveProcessing` - contains an implementation of the `Save()` method. It is necessary to save the settings to a certain object in this handler. For example, to the catalog item.
- `LoadProcessing` - contains an implementation of the `Load()` method. It is necessary to retrieve the settings from a certain object in this handler. For example, from the catalog item.

IMPORTANT! If this or that handler is not implemented, the action that the is implemented by the handler will be unavailable. For example, the `SaveProcessing` handler is not implemented, saving the settings will not be available.

During the development of the storage, the developer independently determines how the storage object will be identified, thereby determining the parameter type. For example, if the settings are saved to the catalog, then the `Code` field or the `Reference` value (of the catalog item) can be used as the settings key.
If interactivity is required for working with settings (settings save and restore forms), then it is required to implement the settings save and restore forms and fill in the corresponding properties of the `SettingsStorage` object (Main save form and Main load form).

IMPORTANT! The implementation of the forms for saving and restoring settings is mandatory for performing interactive operations. Programmatic saving and recovery of settings is possible without the implementation of these forms.

When the user applies the settings save or load commands, 1C:Enterprise retrieves the corresponding form of the settings storage object and displays it. For example, when saving report settings, 1C:Enterprise will use the form for saving the metadata object, which is specified as a report settings storage (directly for the report itself or for the entire configuration). In this case the parameters will be passed to the form. See `SettingsDescription` object description in the Syntax Assistant for more detailed information about these parameters.
When creating a form using the wizard, the necessary parameters will be automatically added to the list of form parameters.
The forms must use the passed parameters and filter the settings list accordingly. For example, it is necessary to display only the settings for the settings object (for example, a report) specified in the `ObjectKey` parameter.
If the user chooses the setting, then the form operation result must be the value of the `SettingsChoice` type. In this value, the `SettingsKey` property must contain the key of the selected setting (for example, a catalog item code or another parameter identifying the setting), and the `AdditionalProperties` property - must contain additional information, which could be specified by the user in the form:

```
Close(New SettingsChoice(SavedSettingKey));
```

IMPORTANT! Saving settings in the standard settings storage is not supported if the object key length exceeds 256 characters; settings key length exceeds 128 characters; username is longer than 64 characters.

5.5.12.2. Creating metafata object

To create a Settings storage, create a configuration object with the same name. This can be done in the Common branch, Settings storages item.

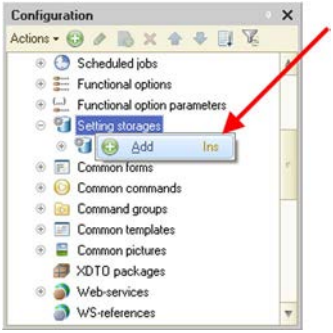


Fig. 80. Creating settings storage

5.5.12.3. Standard settings storage

The standard settings storage is represented in the 1C:Enterprise language by the `StandardSettingsStorageManager` object. This object implements the same set of methods as the `SettingsStorageManager` object, and additionally implements the following methods:

- `GetList()` - method of getting the list of settings for the selected setting object;
- `Delete()` - deletes a specific setting of the selected setting object.

Standard storage saves settings in the system tables of the infobase.
The system settings storage takes a string as the key of the settings object and as the settings key.
As the settings, the system storage takes any value that can be placed to the value storage.

NOTE. When using the `Save()`, `Delete()` and `SetDescription()` methods of the `StandardSettingsStorageManager` object it must be considered that if the object which the settings are related to (for example, form) has already been used in the current session, the changes will be applied in the next session only.

TIP. In case of a significant programmatic settings change (for example, when copying settings from one user to another), it is recommended that the user be prompted to restart the client application.

5.5.12.4. Saving form settings

The developer can manage saving the form data in the settings. To do this, when developing a form, they need to use the following form properties:

- Saving form data in settings - using this property, the form developer can enable the possibility to save form data (with a choice of setting, where the data will be saved to). If the form has requirement to save enabled, the form provides the commands to save/load settings.
- Automatic data saving in settings - indicates the requirement to automatically save the settings when closing the form and to restore the settings when opening the form. At this, it does not matter if the list of settings is used.

If the form is set to have data saving possibility, it is necessary to specify which form attributes must be saved (the Saving column on the Attributes tab of the form editor).

When saving settings the full name of the form is used as the object key. An object of the `Map` type is saved in the settings, where the paths to the stored attributes are stored as keys, and the attribute values -are stored as the values.

5.5.12.5. Saving report settings

The Report and External report objects have the Option storage and Settings storage metadata properties. These properties specify the storages to save the report options and settings to, respectively. If no storages are specified, the storages specified in the configuration properties are used. If the configuration properties also do not indicate specific `SettingsStorage` objects, the system storage is used.

The report form provides commands for saving and loading report options and settings.

If it is required to save some additional information in the data composition settings or the user data composition settings, `AdditionalProperties` properties of `DataCompositionUserSettings` and `DataCompositionSettings` objects may be used. The `AdditionalProperties` property is an object of `Structure` type.

5.5.12.6. Saving dynamic list settings

User dynamic list settings are stored in the storage specified in the configuration properties (`DynamicListsUserSettingsStorage`). If settings storage is not specified in the configuration properties, the system storage is used.

The possibility to automatically save user dynamic list settings is controlled by the Automatically save user settings property of a form attribute of DynamicList type. When the property is set to True, user settings are automatically saved when the form is closed and loaded when it is opened.
The dynamic list form provides commands for saving and loading user settings. The dynamic list also provides a command for setting default settings. When this command is executed, the settings that are in the List.SettingsComposer.Settings property will be applied to the dynamic list.
For a dynamic list, it is possible to invoke the OnUpdateUserSettingSetAtServer event in the following cases:

- When opening the dynamic list form;
- After editing user settings, if the settings components have changed.

5.5.12.7. Settings storage development procedure

The recommended procedure of the settings storages development is given below:

1. It is determined, which storage (see the beginning of the section) will be used. For example, the configuration settings form data settings storage will be implemented.
2. The list of metadata objects that will use this storage is determined, and the list of information stored in the storage, its structure and types is identified. This information will help us select the correct metadata object, where our settings will be stored.
3. Based on the information from item 2, an object (and its structure) to store the settings is created. Let our settings be stored in the catalog items. Since in our case the structure of the stored data is extremely heterogeneous, there is no sense in implementing a separate set of attributes for storing the settings for each form. Therefore, the settings will be stored in the catalog attribute of ValueStorage type.
4. An SettingsStorage type object is created, and the implementation of the save and load forms is performed for it. This ensures interactivity when saving and restoring settings.
5. For the created SettingsStorage object, event handlers are implemented that are related to saving and restoring settings. If this operation is not performed, then the settings reading/writing operations will not be performed. To solve this problem, it is necessary to implement SaveProcessing and LoadProcessing event handlers in the module of the created SettingsStorage object type.
6. In the objects selected in item 2 (or in the configuration properties), the corresponding properties are filled in, thereby indicating to the objects which storages will contain the settings. In our example, it is necessary to

specify a reference to the object created in step 4 in the Form data settings storage configuration property.

7. If necessary, event handlers are implemented which are related to saving and restoring settings in those applied objects where it is required.

5.5.12.8. Working with settings storage using the 1C:Enterprise language

This example illustrates the copying of the current user's settings to other 1C:Enterprise users. `SystemSettingsStorage` is used as the settings source.

NOTE. The example below is not complete. It is intended to demonstrate the operation with the standard settings storage.

The username list is passed as a parameter of the `CopySettings()` procedure as an array.

```
Procedure CopySettings(UserList)

  SettingsDescription = New SettingsDescription;
  SettingsSelection = SystemSettingsStorage.Select();
  While SettingsSelection.Next() Do

    For Each User From UserList Do

      SettingsDescription.Presentation =
SettingsSelection.Presentation;
      SettingsStorage.Save(SettingsSelection.ObjectKey,
SettingsSelection.SettingsKey, SettingsSelection.Settings,
SettingsDescription, User);

    EndDo;

  EndDo;

EndProcedure
```

5.5.13. Common forms

The common forms functionality allows the use of forms available from any module of the current configuration.

If it is necessary to place the command for opening a common form in the command interface, this can be done using Use standard commands property. The command for opening a common form will be placed in the command interface of those subsystems which the common form belongs to. If the common form is included in the functional option and the functional option is disabled, the standard common form command is not displayed in the command interface.

If a common form is created, which will be used as a form of report, report settings or option, it is not recommended to set Use standard commands property for such form.

5.5.14. Commom commands

In this branch, the developer can create commands that do not have object specificity or serve to perform actions with objects that do not use standard commands.

5.5.15. Command groups

In this branch, the developer can create their own command groups. The created group will be located in the section of the command interface that is defined by the Category command property. A command group can be placed in:

- Navigation panel
- Form navigation panel
- Actions panel
- Form command bar

5.5.16. Common templates

The common templates functionality (print forms, report forms, reference data, etc.) allows creating print form templates that are available from any module of the current configuration.

5.5.17. Common pictures

5.5.17.1. General information

The Designer allows to include graphic pictures- images in the configuration. Images can be placed in certain controls, forms, templates, and can also be accessed using the 1C:Enterprise language.

The application can run on devices with different screens: size, resolution, point density. Point density is characterized by the **dpi** (or **ppi**) concept - i.e. the number of pixels per inch of the screen size. The higher the dpi value, the less physical space will be occupied by the image with fixed physical dimensions. In addition, the image may be displayed in a form for which a scale is indicated, that is, the format may be larger or smaller than its base size.

In order to better match the image to different screens, it is possible to specify a image by a set, where each image corresponds to a certain dpi interval of the mobile device screen. In this case, the platform automatically picks the desired image (from the set), based on the parameters of a particular physical device. The application programmer must not take any actions to display the desired image.

A set of images is a zip-format archive that contains images, which are described in the manifest file (manifest.xml) located in the same archive. The image may refer to one of the screen resolution options:

- bldpi - scale 85%

- lpdi - scale 100%, for mobile devices - resolution at least 140 dpi
- aldpi - scale 125%
- mdpi - scale 150%, for mobile devices - resolution 140 to 180 dpi
- amdpi - scale 175%
- hdpi - scale 200%, for mobile devices - resolution 180 to 270 dpi
- xdpi - scale 300%, for mobile devices - resolution over 270 dpi
- udpi - scale 400%

The name of each image can be arbitrary, and the ratio between the name of the image and the resolution is specified in the manifest. The ratio of the image size in the archive can be calculated based on the image size for the scale of 100%. Thus, if at a scale of 100% the size of a image is 16 pixels, then the following set of image sizes for different display scales can be obtained:

- bldpi: 16*0.85 = 14 pixels
- lpdi: 16*1.00 = 16 pixels
- aldpi: 16*1.25 = 20 pixels
- mdpi: 16*1.50 = 24 pixels
- amdpi: 16*1.75 = 28 pixels
- hdpi: 16*2.00 = 32 pixels
- xdpi: 16*3.00 = 48 pixels
- udpi: 16*4.00 = 64 pixels

The archive may also contain the so-called template images. These are the images, the informative part of which is present only in the alpha channel of the RGBA model. The values of the remaining channels in such image must form a white color (R=255, G=255, B=255). Template image is recommended to be generated in PNG format (32 bits per pixel). When displaying such image, the field allocated for the image will be filled with the current background color, and the text color (with relevant intensity) will display those areas of the image for which the alpha channel has values different from full transparency (A=0).

The archive must contain a image to allow ldpi resolution (**basic** image). The other images are optional. If the desired image it absent, 1C:Enterprise will use the image, the parameters of which are as close to the desired ones as possible.

If it is planned to use a image as an icon in a menu, toolbar, spreadsheet document, etc., it is important to set the correct size for it so that it is displayed without distortion.

Recommended image size for 100% display scale:

- for icons - not more than **16x16** pixels;
- for the use in tabular field - not more than **14x14** pixels;
- as the editing field selection button - not more than **9x9** pixels;
- for the image to be used as subsystem presentation - not more than **48x48** pixels.

TIP. If the image is planned to be used in several places, it is reasonable to limit its size to a smaller value.

5.5.17.2. Editing images

The Image Library window is intended for operations with the images. To call it, specify the Common images branch in the Configuration window and select All images item from the context menu.
A screen for maintaining a list of images (see fig. 81) is displayed on the screen.

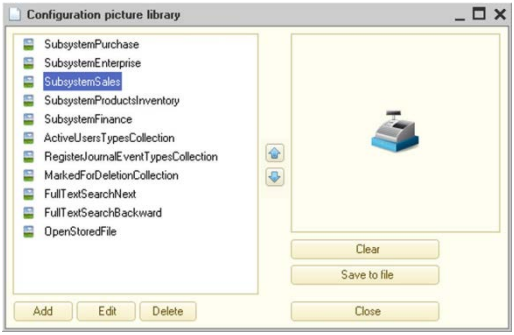


Fig. 81. Image library

To add a new image, click Add. A window is displayed on the screen where it is possible to select a image from a file or open a image editor and create a new image, and to select or modify a transparent background for the image. To select a ready image, click the Select from file button and select the file which contains the prepared image or a set of images. 1C:Enterprise allows to use the BMP, GIF, JPG, PNG, TIFF, ICO, SVG images, metafiles (WMF, EMF) or sets of images in zip archive.
The extensions NETSCAPE2.0, ANIMEXTS1.0, and XMP DataXMP are supported for the figures of GIF format. Other extensions are not supported and the 1C:Enterprise behavior in this case is not defined.
The following items are ignored for SVG images:

- Animation, java-script and any interactivity, including CSS properties associated with interactivity, for example, pseudo-class `:hover`.
- Style properties:
  - `color-interpolation`
  - `color-rendering`
- Zero-length path markers
- `feTile` primitive filter
- Elements of embedded fonts:
  - `font`
  - `altGlyph`
  - `altGlyphDef`
  - `altGlyphItem`
  - `glyphRef`

- `enable-background` property
- References to external resources
- The `image` element does not support the relative path to the image file.

TIP. For images that will be used as interface icons (see recommended image sizes at the beginning of the section), it is recommended to use formats that support lossless compression (PNG and GIF) in order to avoid distortion and minimize traffic between the client and the server.

It is necessary to specify the name by which this image will be selected using 1C:Enterprise language.

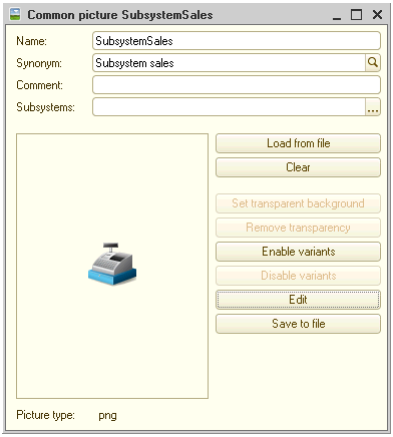


Fig. 82. Image properties

To make the image fit well into a control or form when pasting, it is advisable to set a transparent background to it. To do this, just select any color that is not used in the picture as the background color when editing an image, create an image and save it. For an existing image, any color can be specified. Due to the established transparency of a given color, details of that section of the form that covers the area of the image become visible through it.

To set a transparent background, click the Set transparent background button. The mouse pointer will change its appearance. Place the pointer on the image section, that you want to make transparent, and click the left mouse button. The selected color becomes transparent.

To remove transparency, click the Clear transparency button.

Set transparent background and Clear transparency buttons are only available for BMP, JPG and TIFF images. For images in other formats, these buttons are unavailable.

Similar actions can be performed using the Open link in the Image property. This opens the image selection dialog box.

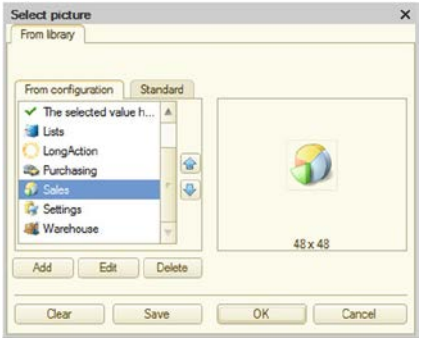


Fig. 83. Picture Selection

To edit the image press Edit button. Image editor is started.
The Enable set of options and Disable set of options buttons allow to process images, which are represented by a specially designed archive containing several images, each of which is drawn for its own scale option (described in detail earlier in this section). Clicking the Disable set of options button causes all images to be deleted from the image set except the base image (the image for ldpi resolution). If there is no image for ldpi resolution in the set, the set becomes empty. If the set contains an image for ldpi resolution, and it is set as an outline, then the template flag is reset. Clicking the Enable set of options button causes the current image to become the base image (for ldpi resolution). If required to edit one of the images in the set, click the Edit button. In this case, a dialog box will be opened where it is necessary to select one of the images in the set that is required to be edited.

Using the image editor, it is possible to set the following interface options: Personal computer (Taxi), Personal computer (Interface 8.2), Mobile device (Taxi) and Any. The other options can be specified only by direct editing the manifest file.
The Collection mode check box indicates that each image in this option set is a collection of images. If the collection mode is disabled for an image, the columns responsible for the collection parameters (width and height of the collection cell in pixels) will be deleted from the image dialog box.

If an image for any resolution is not set, the input field corresponding to the image will be blank. Clicking ... button in the corresponding image set editing dialog box will open the image selection dialog box. Clicking X button will clear

the corresponding image of the dialog box. Editing an individual image of a set does not differ from editing an individual image.

5.5.17.3. Image manifest

If an image contains a set of options, such image must be represented by an archive containing the packed actual image files and the manifest file (manifest.xml), which describes which images are prepared for which scale options in this archive.

Manifest (manifest.xml) must match the schema below:

```
<xs:complexType name="PictureVariant">
  <xs:sequence>
  </xs:sequence>
  <xs:attribute name="name" type="xs:Name" use="required"/>
  <xs:attribute name="screenDensity"
type="tns:PictureVariantScreenDensity" use="required"/>
  <xs:attribute name="isTemplate" type="xs:boolean" default="false"/>
  <xs:attribute name="interfaceVariant"
type="tns:PictureInterfaceVariant"/>
  <xs:attribute name="glyphWidth" type=:xs:decimal" />
  <xs:attribute name="glyphHeight" type=:xs:decimal" />
</xs:complexType>

<xs:simpleType name="PictureVariantScreenDensity">
  <xs:restriction base="xs:string">
    <xs:enumeration value="bldpi"/>
    <xs:enumeration value="ldpi"/>
    <xs:enumeration value="aldpi"/>
    <xs:enumeration value="mdpi"/>
    <xs:enumeration value="amdpi"/>
    <xs:enumeration value="hdpi"/>
    <xs:enumeration value="xdpi"/>
    <xs:enumeration value="udpi"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="PictureInterfaceVariant">
  <xs:restriction base="xs:string">
    <xs:enumeration value="taxi"/>
    <xs:enumeration value="taxiCompact"/>
    <xs:enumeration value="version8_2"/>
    <xs:enumeration value="version8_2_OrdinaryApp"/>
    <xs:enumeration value="version8_0"/>
    <xs:enumeration value="taxiMobile"/>
  </xs:restriction>
</xs:simpleType>
```

`screenDensity` attribute specifies, which display scale this image will be used for. The `interfaceVariant` attribute specifies, which interface option the image is intended for:

- `taxi` - Taxi interface.

- `taxiCompact` - Taxi interface with compact display option. This option is only available when manually editing the manifest.
- `version8_2_OrdinaryApp` - ordinary application interface. This option is only available when manually editing the manifest.
- `version8_2` - Version 8.2 interface.
- `version8_0` - interface for 1C:Enterprise version 8.0 and 8.1 configurations. This option is only available when manually editing the manifest.
- `taxiMobile` - mobile application interface.

The `glyphWidth` and `glyphHeight` attributes describe the size of the collection item for the image specified in the item. These values will be further passed to the image editor, which splits the image into a collection using them.

5.5.18. XDTO packages

The XDTO functionality is a versatile way of presenting data for interacting with various external data sources and software systems.

5.5.18.1. Importing XML schema to XDTO global factory

To import the XML schema from the .xsd file to the XDTO global factory, select the XDTO branch in the configuration tree and execute the Import XML Schema... context menu command.

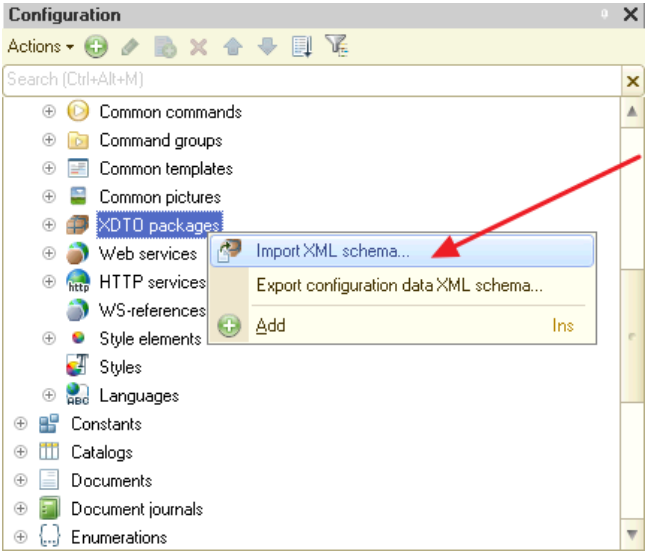


Fig. 84. Importing XML schema

After specifying the required .xsd file, the existence of the XDTO package in the configuration tree will be checked, which namespaces are the same as those imported from the file. If such packages exist, a list of these packages will be displayed and the user will be prompted to specify those packages that must be updated (existing packages are not updated by default).

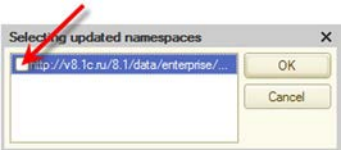


Fig. 85. Namespace selection

When importing XML schema (XSD) in the XDTO package, it is necessary to ensure correct dereference of all the schemas specified in the import and include directives (import, include) -transitive closure. To do this, the import and include directives (import, include) must contain a `schemaLocation` attribute containing the address (or path) of the imported or included schema.
After that, the import will be performed, as a result of which new XDTO packages will be added to the configuration tree, and packages marked for updating will be updated.

5.5.18.2. Exporting the schema of XML configuration data

In order to export the XML schema corresponding to the configuration data types (excluding the XDTO packages created in the configuration tree) to .xsd file,

select the XDTO branch in the configuration tree and execute the Export XML schema of configuration data... context menu command.

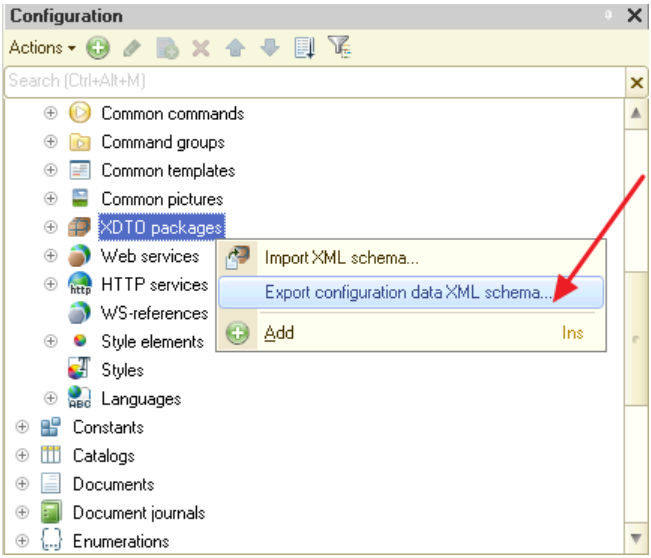


Fig. 86. Exporting configuration schema

After selecting a directory and specifying a file name, the XML schema will be exported to the specified file.

5.5.18.3. Exporting XDTO package XML schema

To import the XML schema corresponding the existing XDTO package, to .xsd file, select the required XDTO package in the configuration tree and execute the Export XML Schema... context menu command.

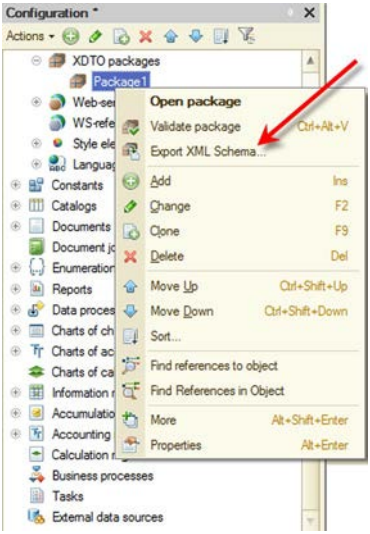


Fig. 87. Exporting XDTO package XML schema

After that, the XDTO package being exported will be checked. If errors are found, the corresponding messages will be displayed in the message window, and the export procedure will be interrupted.

In case of successful verification, user will be prompted to select the directory and the name of the .xsd file, after which the XML schema will be exported to the specified file.

5.5.18.4. XDTO package verification

In order to check the XDTO package, select the required XDTO package in the configuration tree and execute the Verify package command from the context menu.

As a result, the XDTO package model will be verified.

If errors are found, the corresponding messages will be displayed in the message window.

5.5.18.5. XDTO package editing window

5.5.18.5.1. General information

The XDTO package editing is performed in the XDTO package editing window.

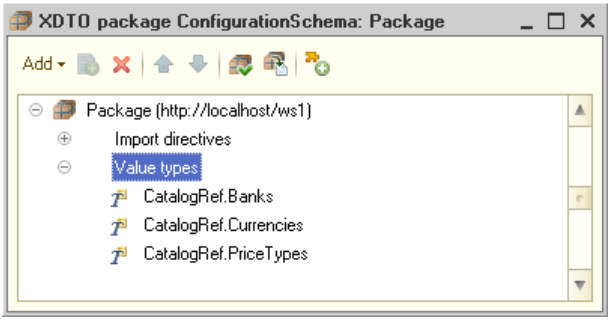


Fig. 88. XDTO package editing window

When a new XDTO package is added to the configuration tree, the XDTO package editing window opens automatically.
In order to open the editing window for the existing XDTO package, select the required XDTO package in the configuration tree and execute the Open package command from the context menu.

5.5.18.5.2. XDTO package hierarchical structure

The XDTO package editing window contains the hierarchical structure of the XDTO package, displayed as a tree.
XDTO package identifier containing the this package namespace URI is located at the root of the tree.
At the first level of the hierarchy the following package items can be located:

- Import directives - list of import directives. Each import directive is a link to another package containing the types that this package refers to in one or another way. When working with this XDTO package using the 1C:Enterprise language, this list of import directives will be available as an `XDTOPackageCollection` object contained in the Dependencies property of XDTO package.
- Value types - list of XDTO value types, which are included in XDTO package.
- Object types - list of XDTO objects, contained in XDTO package.
- Properties - list of XDTO package properties. Represents objects/values declarations that may be the root elements of XML documents belonging to the namespace URI of this XDTO package.

- Each XDTO value type is described by a hierarchical structure and may contain the following items:
  - Pattern - describes one XDTO facet of Pattern type.
  - Enumeration - describes one XDTO facet of Enumeration type.

Each XDTO object type is described by a hierarchical structure and may contain the object property set.

5.5.18.5.3. XDTO package properties

The XDTO package properties are edited in the property palette.
If the property palette is open for the XDTO package selected in the configuration tree, it will contain the following properties: Name, Synonym, Comment and Namespace URI. In addition, the property palette will contain the Package reference, which can be used to go to the XDTO package editing window.

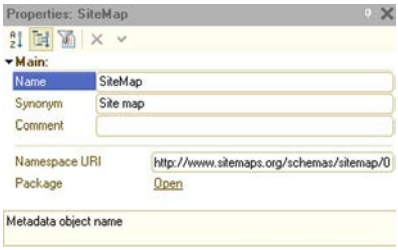


Fig. 89. XDTO package properties

If the property palette is open for the XDTO package selected in the XDTO package editing window (root element), it contains the only property - Namespace URI. This property specifies the XDTO package namespace URI, which all the types defined in this package belong to.

5.5.18.5.4. Import directive properties

The import directive properties are edited in the property palette. For the import directive, the property palette contains the only property - Namespace. This property specifies the imported package namespace URI.

Fig. 90. Import directive properties

5.5.18.5.5. XDTO value type properties

The XDTO value type properties are edited in the property palette.

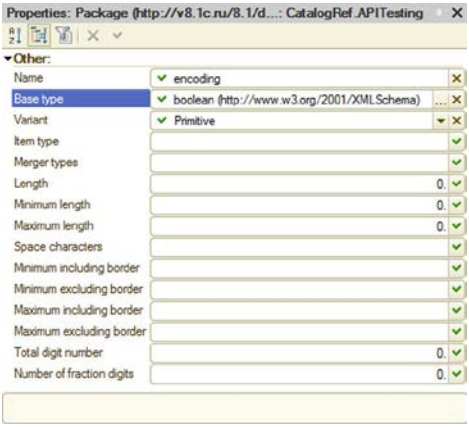


Fig. 91. XDTO value properties

For the XDTO value type, the property palette contains the following properties:

- Name - XDTO value type name.
- Base type - base type for this XDTO value type.
- Option - simple type option (atomic type, list, combination). If the value is set, it must be consistent with the Item type and the Combination types values .
- Item type - list item type when XDTO value type is determined by a list. In this case all the facets and Subordinate types property must be empty.
- Combination types - list of types, forming the combination if the XDTO value type is determined by the combination. Only XDTO value types can be combined. In this case all the facets and Item type property must be empty.
- Length - length facet.
- Minimum length - minimum length facet.
- Maximum length - maximum length facet.
- Space characters - space character facet.
- Minimum, including border - facet of minimum, including the border.
- Minimum, not including border - facet of minimum, not including the border.
- Maximum, including border - facet of maximum, including the border.
- Maximum, not including border - facet of maximum, not including the border.

- Total number of digits - facet of the total number of digits.
- Number of decimals - facet of the number of decimal digits.

5.5.18.5.6. XDTO object type properties

The XDTO object type properties are edited in the property palette.

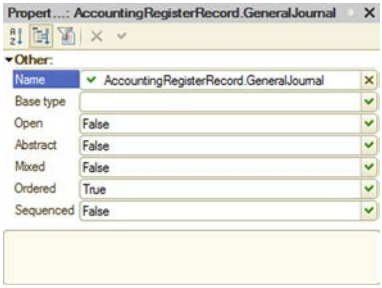


Fig. 92. XDTO object properties

For the XDTO object type, the property palette contains the following properties:

- Name - XDTO object type name.
- Base type - base type for this XDTO object type. This may be XDTO object type only.
- Open - flag, if the XDTO object type is open. This property indicates whether an instance of an XDTO object can contain additional properties that are not defined in its type.
- Abstract - flag, if the XDTO object type is abstract.
- Mixed - the property indicates, if the corresponding XDTO object has mixed contents. If the Mixed property is `True`, then the Serial value must be `True`, since the mixed content cannot be modeled without using the XDTO sequence.
- Ordered - flag of whether the sequence of the items representing property values is strictly the order with the XDTO object type properties. If the Ordered property is `False`, then the sequence of the XML items is not controlled at the input, and at the output it is determined by the sequence of the properties, unless the Serial property is `True`.
- Serial - this property indicates whether the instance of the corresponding XDTO object contains the XDTO sequence. This flag is `True` in those cases where the following order of the nested XML items cannot be determined unambiguously by the sequence of the properties in the type or the corresponding XDTO object has mixed content. The XDTO sequence allows to explicitly define the sequence of the items as they appear in the XML document. For objects of types where the Serial property is `False`, the sequence of nested elements corresponds to the sequence of properties.

5.5.18.5.7. Properties of XDTO object type properties

The properties of XDTO object type properties are edited in the property palette.

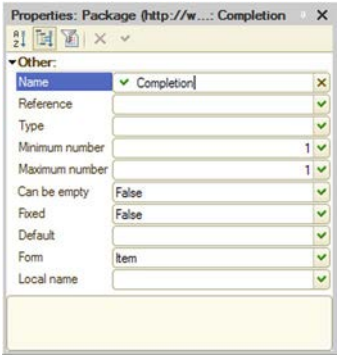


Fig. 93. XDTO type property palette

For the XDTO object type, the property palette contains the following properties:

- Name - property name. Within the XDTO object of the same type, property names must be unique.
- Type - property type. It can be either the XDTO value type, or the XDTO object type.
- Minimum quantity - minimum quantity of the property values. The minimum quantity of property values can be >= 0. Naturally, the Minimum quantity value must be less than or equal to the Maximum quantity value (unless, of course, the Maximum quantity is -1);
- Maximum quantity - property of an XDTO object type can be defined as containing one or many values. A property is considered to contain one value if the Maximum quantity property is 1. If the Maximum quantity property is bigger than 1, then it is considered that the property may contain multiple values. Such a property is modeled as a list in the object structure. The Maximum quantity property indicates the maximum quantity of the property values. The maximum quantity > 1 can be set only for properties that are represented as an XML item.
- Possibly empty - indicates whether the property can have an undefined value. Possibly empty property, when it is `True`, can only be defined for properties with the Item presentation form. If the Maximum quantity is > 1, the undefined value is valid for the property values list item.
- Fixed - indicates whether the property value is fixed. If set to `True`, the fixed value itself can be retrieved through the Default property.
- Default - default property value. The default value type can only be the XDTO value type. At the same time, this value must be compatible with

the property type (to be of the same type as the type of the property or of the inherited type). When creating an XDTO object, the property, if it permits a single value, is assigned the default value. For properties with multiple values, the list of values is initially empty, regardless of whether the default value is defined or not.

- Form - property presentation form in XML. This may Text, Item or Attribute. If the presentation form is Attribute or Text, the value of the Maximum quantity property cannot exceed 1. If the property is set to Text, then the value of the Minimum quantity property must also be 1. One type has the only one property that can be in the Text presentation form, while the other properties must be have the Attribute presentation form.
- Local name - the local name used to represent the property. For the properties with Text represnetation form - it is a blank string.

5.5.18.5.8. Global property

The properties of the global property are edited in the property palette.

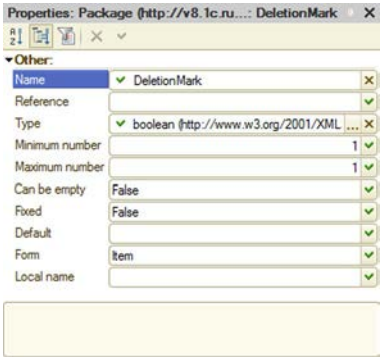


Fig. 94. Property palette of the global property

For the global property, the property palette contains the following properties:

- Name - global property name. Within the XDTO object of the same type, global property names must be unique.
- Reference - reference to the package property root definition.
- Type - global property type.
- Minimum quantity - minimum quantity of the property values. If the Minimum quantity = 0, the property value may be not set.
- Maximum quantity - the maximum quantity of the property values. If Maximum quantity is -1, the quantity of property values is not limited.
- Possibly empty - indicates whether the property can have an undefined value.
- Fixed - indicates whether the property value is fixed.

- Default - default property value. The lexical presentation of the property value must follow the rules for checking this property type.
- Form - property presentation form in XML. This may Text, Item or Attribute.
- Local name - the local name used to represent the property.

5.5.19. Web services

The Web services functionality allows using 1C:Enterprise as a set of services in complex distributed and heterogeneous systems, and also allows to integrate 1C:Enterprise with the other industrial systems using a service-oriented architecture.

5.5.19.1. Adding a Web service

To add a Web service to the configuration tree, select the Common - Web services branch and execute the Add context menu command.
As a result of the command execution, the Web service editing window will open.
In the Miscellaneous tab of the Web service editing window, set the following parameters:

- Namespace URI - contains Web service namespace URI. Each Web service can be uniquely identified by its name and the URI of the namespace which it belongs to.
- XDTO packages - a list of XDTO packages which types can be used as return value types for operations and Web service operation parameter types.
- Publication file name - Web service description file name, which is located on the Web server.

To access the Web service use the address that generates as follows: <Web server host name>/<Virtual directory name>/ws/<Web service name> or <Web server host name>/<Virtual directory name>/ws/<Web service address>.
For example, if the virtual directory has the name DemoWS, the name of web service in Designer is shown as OperationDemoWS and the address is DemoWorkWS, the access to the web service can be performed via either address (to get access from a local computer): http://localhost/DemoWS/ws/DemostrationofWSoperation or http://localhost/DemoWS/ws/DemoWorkWS.
In addition, the tab contains the Module button, which allows to open the Web service module for editing.

5.5.19.2. Web service hierarchical structure

Each Web service described in the configuration tree can contain a set of operations. Each operation must correspond to the exported procedure described in the Web service module.

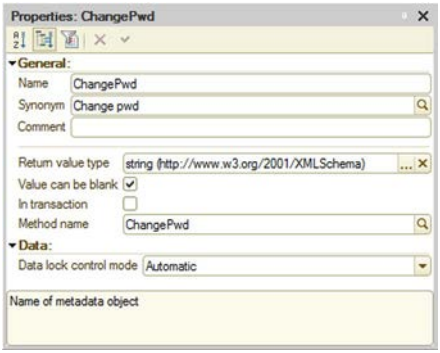


Fig. 95. Web service description

In turn, each operation may contain a set of parameters, the names of which must correspond to the parameter names of the procedure that describes this operation.

5.5.19.3. Web service operations

Web service operation is added in the Operations tab. The operation properties are edited in the property palette.

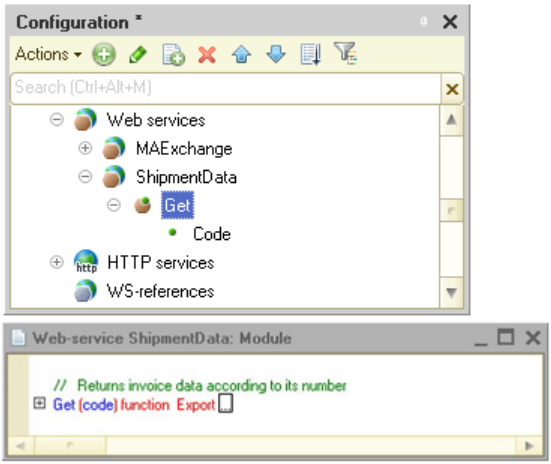


Fig. 96. Web service operation property

Return value type - type of value, returned by the Web service operation. It may be either XDTO value type or XDTO object type.
Possibly empty value - indicates whether the return value can be undefined.
In transaction - indicates, if the Web service module code will be executed in transaction. If the property is set, then when the Web service is called, the transaction will be automatically started, and upon completion, the transaction will either be docked or rolled back (depending on the results of the execution). If the property is not set, the transaction will not start when the Web service module execution starts.
Method name - name of the Web service module procedure that will be executed when this property is called.
Data lock management mode - indicates, which locks will be used when accessing the data.

5.5.19.4. Operation parameters

On the Operations tab for the specified operation, it is necessary to set the parameters for the Web service operation. The parameter properties are edited in the property palette.

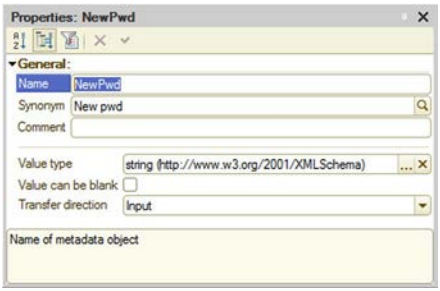


Fig. 97. Operation parameter properties

Value type - the value type of the Web service operation parameter. It may be either XDTO value type or XDTO object type.
Possibly empty value - indicates whether the operation parameter value can be undefined.
Transfer direction - determines the data transfer direction using this parameter. Values:

- Input - means that the parameter can only be used to transfer data to the Web service.
- Output - means that the parameter can only be used to retrieve data from the Web service.
- Input - Output - means that the parameter can be used for both data transfer and retrieving data from the Web service.

5.5.19.5. Specifying system defined types

In order to use the types defined by the 1C:Enterprise in the Web service (for example, in the parameters and return value of the operations), it is necessary to define the XDTO packages in the configuration and for each package specify the platform packages set, which include these type, in its list of imported packages (Import directives property). The namespace URI for specifying the type is contained in the article of the Syntax Assistant for an object of this type.

5.5.19.6. Publishing web services

Web services publishing is described in the appropriate book.

5.5.20. HTTP services

The HTTP services functionality allows using 1C:Enterprise as a set of services in complex distributed and heterogeneous systems, and also allows integrating 1C:Enterprise with the other industrial systems.

5.5.21. WS references

The 1C:Enterprise can use web services provided by the third parties using static links created in the configuration tree.

5.5.21.1. Adding WS reference

In order to add a static link to an external web service to the configuration tree, select the WS references branch and execute the Add context menu command or the corresponding Actions menu command.

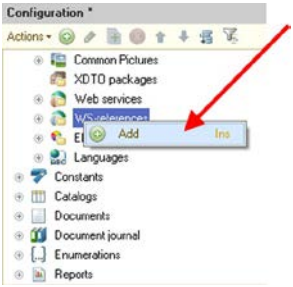


Fig. 98. Adding WS reference

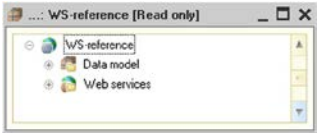
In the opened window, enter the description URL of the web service to be added, for example: http://users.v8.1c.ru/ws/products.1cws?wsdl.

NOTE. When adding a WS reference, remember that 1C:Enterprise the trailing character "/" from the entered URL, due to which the URL http://localhost/ws/ws-service/?wsdl and URL http://localhost/ws/ws-service?wsdl in terms of 1C:Enterprise will be considered identical.

5.5.21.2. WS reference hierarchical structure

The WS reference hierarchical structure is viewed in the WS reference viewer window. The reference element property values can be viewed in the property palette.

In order to open the WS reference viewer window, select the required WS reference in the configuration tree and execute the Properties context menu command. After that, in the property palette, use the WS reference link, which opens



a window for viewing WS reference.

Fig. 99. WS reference

The viewer window contains the WS reference hierarchical structure displayed in a tree view.

The following items can be located at the first level of hierarchy:

- Data model - contains a list of XDTO packages describing the type structure used by web services referenced by this WS reference.
- Web services - list of Web services referenced by this reference.

The structure and properties of the data model are viewed in the same way as when working with XDTO packages, except for the fact that editing the package properties displayed in the WS reference viewer window is impossible.

The WS reference structure is viewed in the same way as when working with Web services, except for the fact that for each Web service, supported Web service connection points are displayed, for which, in turn, a list of operations and parameters of each operation is displayed.

Fig. 100. Web service structure

Various Web service connection points allow executing the operations using different protocols.

5.5.22. Styles and style elements

Style - is a collection of various settings used for the user interface design. Using the style allows to quickly change or set the desired design. Style includes a specific set of system style elements and an arbitrary number of user style elements. The main configuration style is set using the Main style configuration property. It is also possible to perform a programmatic style setting using the `MainStyle` global context property. This property is writable only in the `SessionParametersSetting` event handler. Reading this property is possible at any time.

The following elements are the system style elements (indicating the areas of application of each style element):

- Activity color (`ActivityColor`). The following is displayed in this colour:
  - Client application for personal computer:
    - Current form element border.
    - Current menu item background.
    - Current table row and cell background.
  - Mobile client:
    - Current menu item background.
    - Current table row and cell background.
    - Form title background.
    - Indicator color.
    - Desktop pictures color.
    - Is used for the formation of the form table line context menu button background.

- Accent color (`AccentColor`). The following is displayed in this colour:
  - Client application for personal computer:
    - Group title text.
    - Check box, switch and strip control.
    - Indicator of the current form in the open panel.
    - Image of the notification center on the toolbar (if there are new notifications).
  - Mobile client:
    - Selected page title text color.
    - Check box, switch and strip control.
    - Buttons color.
    - Only in case of operation in Android OS:
      - Band color in the input field in the edit mode.
      - Color of the bands on the spin-wheel date picker.
- Navigation color (`NavigationColor`). The following is displayed in this colour:
  - Client application for personal computer:
    - Sections panel background.
  - Mobile client:
    - Unused.
- Additional navigation color (`AuxiliaryNavigationColor`). The following is displayed in this colour:
  - Client application for personal computer:
    - Current section functions panel background.
  - Mobile client:
    - Unused.
- Important color (`ImportantColor`). The following is displayed in this colour:
  - Client application for personal computer:
    - Default button background.
  - Mobile client:
    - Unused.

Setting up any other system elements of Taxi interface, except those listed above, is not supported. However, it is possible to use other system style elements for designing form elements. They will contain effective color combinations, but cannot be changed.

Style elements configuration objects are intended for uniform design of various form elements in cases where the design that is automatically offered by 1C:Enterprise is not sufficient. For example, it is required to make the color of any labels in configuration forms the same. In this case, it is reasonable to create a style element, set a color for it and use the created element to set the form element text color.

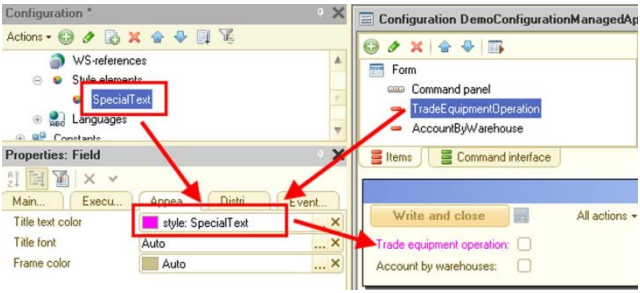


Fig. 101. Style element usage

Style elements may be of three various types:

- Color,
- Font,
- Border.

NOTE. The names of the system style elements are not available for selection as user style elements values.

In the font selection dialog in the thin client, the list of fonts contains fonts installed on the current computer and a list of special fonts enclosed in angle brackets (<>). If the <Text font> font is selected, then the 1C:Enterprise interface font will be used, the other fonts correspond to the respective operating system fonts. The <> element exists in the beginning of the <Text font> font size list. The choice of this font size (its value is 0) means that the font size of the 1C:Enterprise interface will be used. The font style (bold, italic, etc.) is taken from the style by default, but can be changed by the user without any restrictions. When another font is selected, user changes of size or style are not saved and are set to default values.

It is possible to programmatically access the value of a style element using the `Value` property.

Example:

```
Metadata.StyleItems.ColorOfNegative.Value
```

5.5.23. Languages

Languages configuration objects are intended to create a program interface in different languages. For each configuration object of the Languages type, a separate row is reserved for those metadata attributes which are allowed to be presented in different languages.

Consider an example of the form for `Products` catalog item. For the Russian language the form labels are as follows:

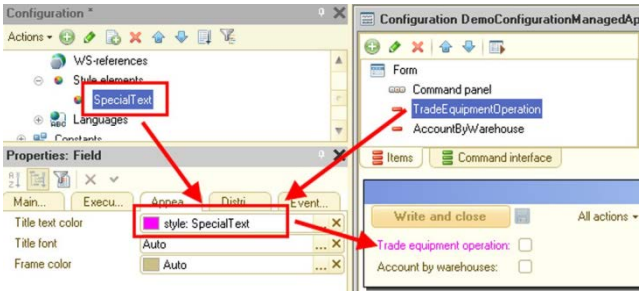


Fig. 102. Form in Russian

If in the Languages branch several object are created (for example, Russian and English), then in order to change the language it is necessary to select Configuration - Configuration editing language item. In the language selection window, select the configuration viewing language.

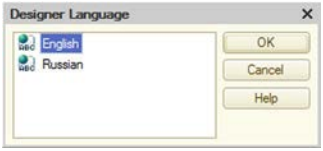


Fig. 103. Selecting the configuration editing language

A similar result can be achieved using the language selection button located in the status bar to the right from the CAP and NUM buttons (bottom right corner of the main Designer window).

The Designer will replace the text of the labels with the labels in the selected language.

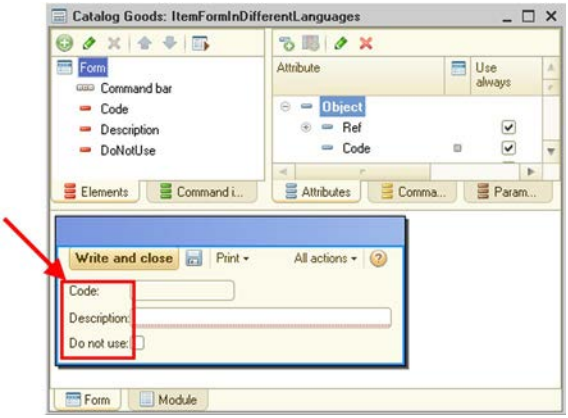


Fig. 104. Form in English

Of course, the text of the labels must be pre-entered for each control. To enter the text in the `Label` control property palette in the `Title` property (`Text` or `Synonym`, depending on the control type) it is required to press "magnifier" button (see fig. 105). The window Strings in different languages will be displayed.

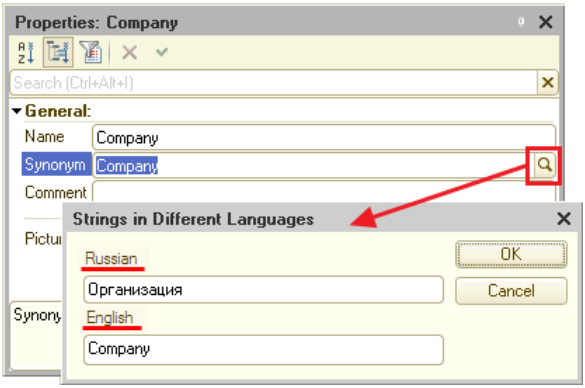


Fig. 105. Strings in different languages

TIP. There are no restrictions on the number of languages in the configuration. However, the objects of the Languages type should not be created “with a reserve”, since the Language object can be created at any time.

The Language code property specifies the language code, for example, EN for English.
If two or more objects of the Languages type are defined in the configuration, then for the Synonym and Title properties of the control a button for text editing in different languages appears (in the form of a magnifying glass).
The program creates the first language object in accordance with the choice of language (country) when creating a new infobase.
To generate a text presentation of an attribute for display in a form, the following rule is used:

- The title of the displayed object is retrieved in the current user's **configuration language**. If the title is set, it is used.
- An attempt is made to retrieve a synonym for the displayed object in the current user's **configuration language**. If a synonym is set, it is used.
- Further behaviour depend on the displayed object type:
  - for standard attributes, the presentation is obtained in the **platform localization** language;
  - for the objects created by the application developer, the object name is used in the same way (and in **that language**) as specified in the configuration.

IMPORTANT! If texts in the Synonym or Title properties are entered, changing the language code in the Language code property will cause a “loss” of the entered texts (texts will remain for the previous code value). Texts are “restored” when specifying the previous value of the language code.

To edit texts and perform localization tasks (creating an interface in different language), use the Interface text editing mode.
If the application is intended to be used on the mobile devices, it is recommended to specify language codes in accordance with ISO 639 standard (http://www.iso.org/iso/ru/language_codes). Otherwise, a warning will be displayed when verifying the configuration.
The following characters are not allowed in the language code: #, *, |, \, :, ", <, >, ?, /.

5.6. Common configuration object properties

This section describes the common properties of metadata objects.

5.6.1. General properties

Virtually all configuration objects have the following properties, which are located in the General properties category:

- Name - configuration object name. The name must consist of one word, begin with a letter and contain no special characters, except "_". The object name is used to access and manage the configuration object using 1C:Enterprise language tools. The names of the configuration objects must not coincide with the reserved words of the query language. The name of the configuration object cannot exceed 80 characters. Configurations created in 1C:Enterprise version 8.3.6 and earlier may have configuration object names longer than 80 characters, however, if when attempting to edit such object in version 8.3.7 or higher, - it will be necessary to reduce the name to 80 characters.
- Synonym - in addition to the name, its synonym can also be specified. If the configuration is created for use in different languages, the synonyms in the languages used must be specified. When operating the 1C:Enterprise, a synonym will be output in various selection lists, window titles, text labels, when creating interfaces based on the current language. The synonym has no restrictions on the character use. If no synonym is specified, the name is selected.

NOTE. The name or synonym displayed to the user are also called the **configuration object presentation**.

- Comment - arbitrary character string. Usually explains the object name.

5.6.2. Metadata objects presentation

The possibility to define various object presentations is implemented for many metadata objects. This is done so that the developer has the opportunity to define the presentation of standard commands, their tooltips and form titles.

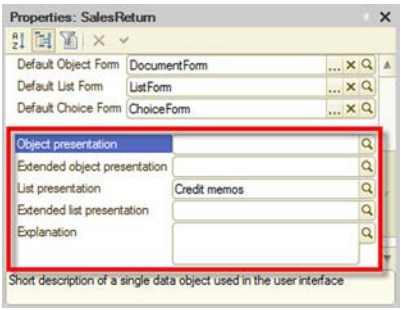


Fig. 106. Objects presentation

- Object presentation (for register- records):
  - one object name (for example, Current account);
  - used in the standard command presentation (object creation).

- Extended object presentation (for register- records) used to generate the object form title. For example, Company current account.
- List presentation:
  - object list name (for example, Current accounts);
  - used in the standard command presentation (command to open the list of objects).
- Extended list presentation - used to generate the list form (for example, Company current accounts).
- Extended presentation - report or data processor form title (for example, Report on current account offsets).
- Explanation - used to generate a tooltip for the standard commands (for example, Our organizations' current accounts).
- Image - image for subsystem presentation in the sections panel.

It should be noted that filling in the properties associated with the presentation of objects and lists is only necessary in cases when it is required to somewhat specify the default information displayed for the object.

For example, there is a Goods catalog (Name of metadata object - Goods, Synonym of metadata object - Goods), where products and services may be its items. But you want to use a singular noun for the command (create product) in the command texts and do not want to reflect information about the services, because it "lengthens" the command text. At the same time, you want to indicate to the user in the object form that both products and services can be edited in this form. In this case the Object presentation is reasonable to be filled with Goods text, and Extended presentation of the object property - with Goods (service) text. Then the command to create an Goods directory item will look like Goods: create, and the form title - Goods (service).

It is also possible to retrieve the information about the rules for the standard command text, command tooltips and form titles generation there.

5.6.3. Arbitrary data presentation

The standard data presentation which is generated by 1C:Enterprise may not suit the user. For example, if users, that speak different languages , use 1C:Enterprise and the system objects contain all the necessary information in different languages as well. For example, there are two names for products: in English and in Russian. And it is necessary to display the necessary text depending on the session localization code. It is also necessary to generate a presentation in the desired language wherever the presentation of the object is generated: dynamic list, report, etc.

To implement this task, there is a special feature that allows the developer to determine the attributes that are involved in the generation of the presentation, as well as to describe the algorithm using which the object presentation is generated. This is possible for the following configuration objects:

- Exchange plans
- Catalogs
- Documents
- Charts of characteristic types
- Charts of accounts
- Charts of calculation types
- Business processes
- Tasks
- External data source tables.

The presentation generation functionality consists of two stages: identification of the attributes involved in the generation of the presentation and the actual presentation generation.

To determine the list of attributes, the `PresentationFieldsGetProcessing` event handler of the corresponding object manager is used. The handler is called at the first attempt to retrieve a selected object presentation, and the result is preserved for the duration of the session. In this handler, the `Fields` array must be filled with the names of the attributes that will be involved in the presentation generation. If the `StandardProcessing` parameter in the handler is `True`, then after the handler completes its operations, the `Fields` array will be cleared and filled with the fields used to generate the standard presentation of this object. Otherwise, the platform will use only the names of the fields that were placed in the `Fields` array.

It is not recommended to include the fields of unlimited length in the list of fields used for the presentation. If this is done, then there may be problems when retrieving the field presentation for fields of a composite type and when using the presentation in queries with grouping and retrieving unique records.

It is also strongly not recommended to pass reference fields to the field list (including the `Reference` field) and retrieve the required data using dot from such fields. In this case, the system performance will be drastically reduced due to the fact that the required fields will be retrieved in extremely inefficient way. At the same time, data retrieval for the fields specified in the `Fields` parameter will be carried out in the most efficient way, and in this case there will be no “extra” readings of the objects for which the presentation is generated.

NOTE. The examples below are not complete. These are intended to demonstrate the techniques of the arbitrary representation operating.

Let's assume, that the `Goods` catalog contain `RussianName`, `EnglishName` and `SKU` attributes, which are to be involved in the presentation generating. In this case, the handler for retrieving the presentation fields (in the `Goods` directory manager module) will be as follows:

```
Procedure PresentationFieldsGetProcessing(Fields, StandardProcessing)

  Fields.Add("RussianName");
  Fields.Add("EnglishName");
  Fields.Add("SKU");
  StandardProcessing = False;

EndProcedure
```

If necessary to retrieve the presentation 1C:Enterprise will call the `PresentationGetProcessing` handler in the corresponding object manager. The values of the attributes involved in the representation generation are passed to this handler. The program code located in the data processor handles the generation of the presentation text.

IMPORTANT! The event handler `PresentationGetProcessing` is called whenever there is a need to retrieve a representation of any infobase object. Excessive data, or the wrong choice of data to generate a presentation, can cause significant 1C:Enterprise performance drop.

The `PresentationGetProcessing` handler generates the presentation and returns it using `Presentation` handler parameter. The data necessary for the representation generation are passed using the `Data` parameter. `Data` is a structure, where the key is the name of the attribute, and the value - is the value of the attribute for the current object. If the `StandardProcessing` parameter is `True`, 1C:Enterprise will attempt to generate a standard presentation for the current object, based on the data passed. If the `Data` parameter contains no attributes necessary for the generation of a standard presentation, then the presentation will be an blank string. If the presentation generation is performed for a new object (not yet stored in the infobase), then the `Undefined` value will be passed as the value of the `Reference` field of `Data` structure.

Consider an example of generating the presentation of bilingual catalog`Goods`:

```
Procedure PresentationGetProcessing(Data, Presentation,
StandardProcessing)

  StandardProcessing = False;
  SessionLocalizationCode = Upper(CurrentLocaleCode());
  If Find(SessionLocalizationCode, "RU") <> 0 Else
    Text = Data.RussianName;
  ElseIf  Find(SessionLocalizationCode, "EN") <> 0 Else
    Text = Data.EnglishName;
  Else
    Text = Data.RussianName;
  EndIf;
  Presentation = Text + "(" + Data.SKU + ")";

EndProcedure
```

It is worth noting that if 1C:Enterprise language error is made in the `PresentationGetProcessing` handler, the application may unexpectedly shut down.

5.6.4. Standard attributes

In order to redefine certain interface properties (such as a synonym, filling check, etc.) of standard attributes (for example, `Code`, `Name`, `Parent`) and standard tabular section (for example, `ExtraDimensionTypes`, `BaseCalculationTypes`) of the applied objects at the configuration level, it is possible to configure these properties.

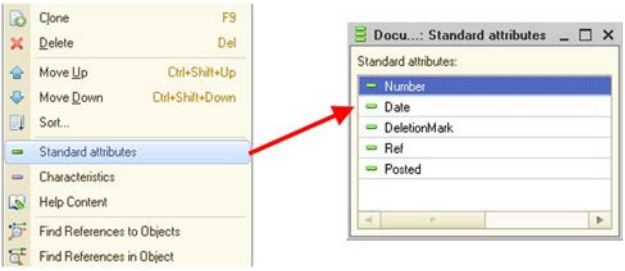


Fig. 107. Standard attributes

To do this, there are commands that open lists of standard attributes and standard tabular sections in the property palette of several objects (see fig. 107). Such commands are available for those objects that have standard attributes and standard tabular sections.

Using the property palette, it is possible redefine certain properties of standard attributes in such a way that these more fully meet the requirements of the applied task being solved. For example, for the `Owner` property of `CurrentAccounts` catalog it is possible to set the Counterparty synonym.

Then in all the form the default presentation of this attribute (Owner) will be "Counterparty:".
If the standard attribute (or the standard tabular section) properties are not specified, the default standard attribute properties will be used.
In terms of the set of properties, standard attribute is almost identical to any other attributes, except for the fact that the following cannot be changed:

- standard attribute name,
- standard attribute type,
- standard tabular section name,
- standard attribute indexing.

IMPORTANT! The presence of the standard attribute description does not change its name, that is, all the calls to the attribute in the 1C:Enterprise language and in the query language remain unchanged.

5.6.5. Predefined data

5.6.5.1. General information

Predefined data - are the elements of applied objects that are created in the Designer and can be accessed by name without prior search. It makes sense to create predefined data in the event that some data element is needed all the time and access to this element from the program code needs to be to simplified. For example, in the Goods catalog a Service predefined element can be created, accessing which will be as follows: Catalogs.Goods.Service. Both elements and groups of elements can be predefined. Groups can be created when predefined elements are created in hierarchical objects, for example, in a hierarchical catalog. Predefined data can be created:

- for catalogs,
- charts of accounts,
- charts of characteristic types,
- charts of calculation types.

After creating a predefined element in the Designer, the possibility to create or update the associated data element is determined by several parameters:

- The value of the object property in the infobase. Retrieving and setting the value of this property in 1C:Enterprise mode is possible using the GetPredefinedDataUpdate() and SetPredefinedDataUpdate() methods;
- Value of configuration object property PredefinedDataUpdate. Set in the Designer;
- The value set for the entire infobase. Retrieving and setting the value of this property in 1C:Enterprise mode is possible using the

`GetPredefinedDataUpdate()` and `SetInfobasePredefinedDataUpdate() methods`;

- Type of distributed infobase node. The value of this parameter is determined automatically:
  - For the root node and non-distributed infobase, the value is Update automatically;
  - For the subordinate node of the distributed infobase - Do not update automatically.

Each parameter described above can have one of the following values (except for the parameter describing the type of the distributed infobase node):

- Do not update automatically - in this case 1C:Enterprise does not create or update a data element when creating or modifying predefined data. It is should be noted that if such property is set for a configuration object, an exception may be thrown when accessing predefined data, since there is no data element associated with the predefined element;
- Update automatically - in this case 1C:Enterprise will automatically create (or update the existing) data elements for new (or modified) predefined data;
- Automatic - 1C:Enterprise automatically determines the need to perform the update.

The action to be performed (to update or not to update) is determined by sequential analysis of the above parameters (in the specified order) until a value other than Auto is found. This value will determine the capability to create or update predefined elements.

The data element and the predefined data element are linked using the `PredefinedDataName` property. Using this property it is possible to perform the following actions:

- Associate a predefined data element with a data element. To do this, assign the `PredefinedDataName` property the name of a predefined data element to be associated with the data:

```
ElementReference = Catalog.Goods.FindByDescription("Delivery
service");
Object = ElementReference.GetObject();
Object.PredefinedDataName = "Service";
Object.Write();
```

As a result of this operation, the data element with the Delivery service name will be mapped to a `Service` predefined element in the `Goods` catalog.

When writing an object, it is checked that there is no other data element in the infobase with the name of the predefined data set to the value which the data element is written with.

- “Unlink” the data element from the predefined data element. To do this it is necessary to assign `PredefinedDataName` property to a blank string and write an element:

```
Object = Catalog.Goods.Service.GetObject();
Object.PredefinedDataName = "";
Object.Write();
```

When attempting to access the `Catalog.Goods.Service` predefined element after executing the above example, an exception will be generated.

Thus, the process of changing the data element associated with the predefined data is carried out in two stages:

- "Unlinking" of the existing data element from the predefined element is performed;
- New data element is associated with the predefined element.

Creating data elements associated with predefined data can be done in one of three ways:

- Automatically, during the infobase restructuring. This happens in the following case:
  - Automatic creation and update of the predefined data is permitted;
  - Predefined data was previously created in a given data area or infobase.
- Automatically, when accessing the table that stores the configuration object data for the first time. Creation of predefined elements in this case will be performed, provided that the creation of predefined elements is not prohibited.
- Using 1C:Enterprise language by specifying the `PredefinedDataName` property when creating data element. This method can be used if the automatic creation of predefined elements is prohibited in the configuration objects properties.

Updating data associated with predefined data will be performed if automatic updating of predefined data is enabled, predefined data is associated with the real data, and predefined data is modified in the Designer. In all other cases, the information entered in the Designer will not be passed to the infobase.

The reordering of predefined elements in the predefined tabular sections of the chart of accounts and the chart of calculation types is processed in the associated data, i.e. the order of the elements in the data will be the same as it is specified in the metadata. The data entered by the user (not predefined) will be located after the predefined ones.

If a predefined data element is deleted in the Designer, the following actions will be performed:

- The data element associated with the predefined data element will be marked for deletion.
- The corresponding entries will be deleted from the list of element extra dimension in the chart of accounts.

- In the lists of leading, base and displacing calculation types, the corresponding records will be deleted.

For the forced creation of predefined data elements (for catalogs, charts of characteristic types, charts of calculation types and charts of accounts), use the `InitializePredefinedData() method`. Calling this method will force the creation of predefined data elements when the following conditions are met:

- Predefined elements were not previously created in the current data area.
- Updating predefined data is configured in such way that the creation and updating of predefined data must be performed.

If the specified conditions are not met - the creation of predefined elements is not performed.

If the automatic updating of the predefined data is disabled in the application, then deleting the predefined element causes a change in the associated data object (if any). The `PredefinedDataName` property value is replaced with the value `#xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx`, where `xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx` - is the remoted metadata item UUID.

If it is necessary to retrieve a list of predefined elements for a configuration object, use the `GetPredefinedNames() method`.

An example of retrieving a list of predefined values:

```
List = Metadata.Catalogs.Goods.GetPredefinedNames();
```

If the method returns an empty array, the configuration object has no predefined values. Otherwise, it is possible, for example, to perform a check for the presence of a related element in the data for each predefined element.

5.6.5.2. Usage in the distributed infobase

When operating in the distributed infobase mode, predefined elements are passed with the configuration. The data elements associated with the predefined data are passed in the usual way, using an exchange plan. In this case, the automatic creation of predefined data in the peripheral infobase will not occur.

IMPORTANT! If the data associated with the predefined elements were received by the subordinate node before the configuration with the associated predefined elements, then the automatic "association" of the data and the predefined data will not occur. This will require to re-export the necessary data elements from the central to the peripheral infobase.

If the initial image of the peripheral infobase is created before the operations with the main infobase started, it is possible to use the `PredefinedDataInitialization()` special method to create predefined elements so that they are automatically included in the peripheral infobase. The method must be called before the creation of the initial image.

When cancelling the master node assignment (for example, to perform any special actions), consider the following point: when determining whether it is necessary to create or update predefined elements, such infobase will not be considered peripheral. Due to this fact, during the first infobase start in 1C:Enterprise mode, the mode for creating or updating predefined items can be defined as Update automatically. To avoid this, use the /SetPredefinedDataUpdate command line parameter (or use the `SetInfobasePredefinedDataUpdate()` global context method).
The same method can be used if a universal data exchange functionality between infobases with similar, but not identical standard solutions is used, and it is necessary to make one master infobase (in terms of predefined data). For the infobase where it is necessary to create predefined elements, specify the Create automatically value, and for the other infobase specify - Do not create automatically. In this case, the association between the configuration element and the data object will be performed by the predefined element name.
Also, when using a universal data exchange functionality, the following peculiarities must be considered:

- When writing a data object that is predefined in the source infobase, 1C:Enterprise checks for the presence of the same predefined data name in the type-forming configuration object. If such name exists - the object will be imported without changes. If there is no such name - the `PredefinedDataName` property will be cleared for the object during import.
  For example, a universal data exchange between the same infobases is performed. Each infobase contains the `Products` catalog. This catalog in each infobase contains the `Service` predefined element. References to these data objects are different.
  Then the following will occur during import: 1C:Enterprise will detect the presence of a `Service` predefined element in the target infobase and perform the import with the preservation of the `PredefinedElementName` property. This will be possible if the `DataExchange.Import` property is set to `True` before the element is written. Otherwise, an exception is thrown.
  However, when attempting to access the `Service` predefined element in the target infobase, an error will occur, since 1C:Enterprise cannot have two objects associated with the same predefined element.
  This situation must be avoided. An application developer must independently handle an attempt to import a data object with a duplicate name of predefined data.
- When importing the data object, which predefined data name references the remote element (in `#xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx` format), such element will be written in any case. However, 1C:Enterprise will not permit to write two objects that reference one

remote predefined element (with the same predefined data name value), except for the case when the `DataExchange.Import` property is set to `True` before the element is written.

5.6.5.3. Separation mode operation

5.6.6. Objects numeration

5.6.6.1. General information

There are objects in 1C:Enterprise that must be numbered automatically or manually. Numbering is available for the following objects:

- Catalogues;
- Documents;
- Charts of characteristic types;
- Business processes;
- Tasks;
- Exchange plans.

Depending on the object, various types and numbering modes are available. Numbering can be performed for the number (documents, business processes, tasks) and code (all other objects). Consider the numbering peculiarities of objects in more detail.

Code/number type

The code/number of the configuration object can be numeric or string type.
If the code/number is set as string, then several different additional features appear:

- The code/number can consist not only of numbers, but of any available characters.
- Available code length - defines the code/number length. If this property is set to `Fixed`, the length of the code/number will always be equal to the value specified as the Code length or Number length property. In this case, the line will be supplemented with spaces. If the property is set to `Variable`, then the length of the code/number can not be greater than the value specified in the Code length or Number length properties, but the actual length will be determined by the number of characters involved in generating the number.
- For the code/number it is possible to set different prefixes.

When using a string code/number it is necessary to understand how the numbering works. Consider the numbering example. Catalog code type - `String`. Code length - 3 characters. Assume the operator entered the number 01 and wrote the catalog element. As a result of this action, the maximum value of the number that can be created by the automatic numbering system will be 99. This happened because the manual entry of a new number resulted in an obvious limitation of

the number length to two digits. The automatic numbering system “cannot” automatically expand the number created if the number length is explicitly limited. Thus, for the automatic numbering of string codes/numbers to operate, it is recommended either not to specify numbers manually or to set leading zeros so that any code/number is always the maximum possible length (including the leading zeros).
It is also worth remembering that specifying leading zeros will ensure the correct sorting of codes/numbers in case if the code/number type is a string.

Automatic numbering flag

This property determines whether the system will automatically assign a code/number to a new object when the object is written. If the automatic numbering is enabled, 1C:Enterprise has the following behavior:

- A code/number is assigned that does not coincide with any number in the infobase;
- The code/number assigned must be 1 greater than the maximum of the code/number among the objects in the infobase (which are involved in numbering) and the codes/numbers assigned to the objects not yet recorded in the infobase.
- When deleting or manually decreasing the maximum code/number, the numbering (if possible) continues from the maximum remaining number/code. Continuation of numbering from the lowest possible code/number is generally not guaranteed.

Uniqueness check

This property determines whether the system will control the uniqueness of the code/number for the object. The control will be performed both for the automatically assigned code/number, and for the code/number that is specified manually.

Numerator

For documents, it is possible to specify a numerator, which, for example, allows to ensure a common number space for several different types of documents.

5.6.6.2. Code/number series

For catalogs and charts of characteristic types, it is possible to indicate how the new number will be generated. This can be done using Code series property:

- In entire catalog/In entire chart of characteristic types - in the process of generating a new code for the element of the catalog/chart of characteristic types, a code will be generated that is unique in the entire catalog/chart of characteristic types;
- Within subordination area - in the process of generating a new code for the catalog/chart of characteristic types element, a code unique within the element hierarchy will be generated (elements having the same parent will

have different codes, elements having different parents may have the same codes);

- Within owner subordination area - in the process of generating a new code for a catalog element, a code unique within the subordination will be generated (elements having the same owner will have different codes; elements having different owners may have the same codes). This option is applicable only for catalogs.

5.6.6.3. Code / number periodicity

For documents and business processes it is possible to ensure the periodicity of numbers. This means that the numbering will start from the beginning, each time a new period begins. Periodicity is configured using the Periodicity property. This property can have the following values:

- Nonperiodical - numbering does not depend on time periods;
- Within a year - numbering starts from the beginning with the new year beginning;
- Within a quarter - numbering starts from the beginning with the new quarter beginning;
- Within a month - numbering starts from the beginning with the new month beginning;
- Within a day - numbering starts from the beginning with the new day beginning.

The uniqueness control during periodic numbering will be performed within the established period (for a year, quarter, month or day). If nonperiodic numbering is set, then uniqueness will be controlled in all the existing objects of this type.
Belonging to a particular period is determined by the date of the document or business process, and not by the current date. Due to this, a change of the document or business process date may cause a number change in the case when the new date refers to a different periodicity (if set).

5.6.6.4. Numbering prefixes

When automating various tasks, there may be a task of ensuring separate numbering of objects of the same type depending on certain conditions. For example, the records are kept on behalf of several firms in the infobase, and each firm needs its own numbering of documents. This problem is solved using the code/number **prefixes**. The code/number prefix - is a sequence of characters that precede the code/number within which the objects are numbered. The use of numbering prefixes has several features:

1. Prefixes are used with a string code/number type only.
2. The code/number length, when using the prefix, may be greater than without using the prefix (by the prefix and the separator size). Thus, if the code length was initially set to 6 characters, and it is necessary to start using the prefix 3 characters long, then to preserve the numbering space of 6 digits,

it is required to increase the code/number length to 9 characters (6 characters - the code itself and 3 characters - prefix).

3. To automatically change the prefixes depending on the object attribute values, it is necessary to perform additional actions in 1C:Enterprise language.

To set a new number, with the prefix, the `OnSetNewCode()`/`OnSetNewNumber()` handler or `SetNewCode()`/`SetNewNumber()` methods must be used with specifying of the required prefix. If the prefix is not specified or not fully specified, then among the prefixes existing in the information base, the maximum prefix will be determined (in alphabetical order) and a new number will be received within this prefix.

Consider examples of the numbering system behavior.

There are two objects, for which `A-151` and `B-152` numbers are specified. When setting the new code for `A-` prefix the `A-152` number is received, and for `B–` prefix - `B-153` number is received. If the 1C:Enterprise contains objects with the following numbers: `AA151`, `AB150` and `BC114`, then, depending on the prefix set, several variants of numbers received are possible (each description of the prefix actions proceeds from the starting number values and does not consider the actions of the other examples):

- `AA` prefix is specified. `AA152` number will be received in this case.
- `AB` prefix is specified. `AB151` number will be received in this case.
- `A` prefix is specified. `AB151` number will be received in this case. This number will be received as a result of the fact that the prefix is not fully specified and 1C:Enterprise will determine the maximum prefix (in alphabetical order). This prefix will be `AB` and within this prefix the new number is received: `AB152`.

Consider an example of setting a prefix depending on the object attribute value.

Assume, that we have a document that contains an `Organization` attribute of `CatalogRef` type. The `Prefix` attribute (of `String` type) in the `Organizations` catalog contains the document number prefix. Thus, each organization will maintain its own unique numbering. To assign a prefix, in the object module of the document used create a `OnSetNewNumber()` handler, where the document number prefix will be set depending on the organization selected in a particular document instance.

Example:

```
Procedure OnSetNewNumber(StandardProcessing, Prefix)
  Prefix = Organization.Prefix+" ";
EndProcedure
```

5.6.6.5. Automatic numbering feature peculiarities

Control of the object code/number uniqueness, if the Uniqueness control property is set, is performed when it is written, regardless of the objects automatic

numbering mode and how the object is assigned a new code/number - manually or automatically. If the object is written in the “import mode” (when the object is written, the `DataExchange.Import` property is set to `True`) - the code/number is not checked for uniqueness.

5.6.7. Input by string

5.6.7.1. General information

For applied objects (catalogs, documents, charts of characteristic types, charts of accounts, charts of calculation types, registers, business processes and tasks, as well as exchange plans), it is possible to specify the attributes which are used for searching in the Input by string property. If the input by string is permitted (the corresponding attributes are set), then it is permissible not to perform a selection from the object form in the corresponding type input field, but to enter the data that is contained in the set object attributes.

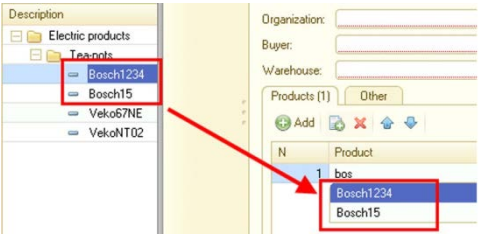


Fig. 108. Input by string

For example, we have the `Products` catalog, which contains several products starting with the word Bosch. Then, by entering the word bosc in the product input field, we get a list of products that begin with this word (see fig. 108).
The following may be used as the attributes using which the search is be performed:

- For catalogs, charts of calculation types, charts of accounts, charts of characteristic types, exchange plans- `Code` and `Name`.
- For documents - `Number`.
- For business processes and tasks - `Number` and `Name`.
- Attributes of `Number` or `String` type, for which Index property is set to Index or Index with additional ordering. An example of the latter is the field for entering the SKU, barcode, TIN.

To generate a list of attributes, press the select button and in the opened dialog box move those fields, for which the input by string is supported, to the left list.

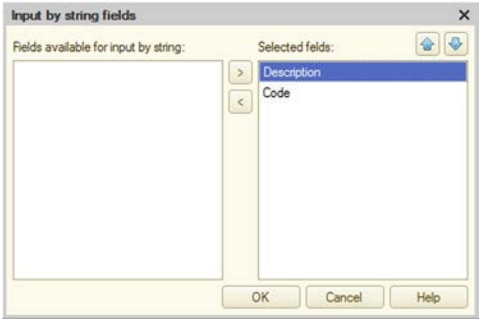


Fig. 109. Configuring input by string

If there are several fields, then set their sequence. When searching by string, the search is performed in the fields in sequence in which they are presented in this dialog box. For example, for the `Products` catalog element, input can be carried out both by code and by SKU. If the value of the one element code coincides with the value of the another element SKU, then the found values will appear in the list of values in the order in which they were specified during configuration.

IMPORTANT! When performing a data search (during input by string), data access restrictions apply.

In addition to specifying which fields will be searched, 1C:Enterprise allows to further customize some search parameters. The String search method property is used to specify the string search method:

- Only in the beginning of the field selected for input by string (Beginning value). In this case, the search string is not broken into words, and the entered text is searched entirely. For example, if the search is performed in the following data: Ivanov Maksim and Maksimov, then when entering the Maksim word in the input field the Maksimov data will be found, and when entering Ivan Maks in the input field nothing will be found.
- In any part of the field (Any part value). If the search string consists of several words and the search is performed in any part, then the search uses a query similar to the following (for two fields and two words):

```
(Field1 DETAILED "%Слово1%" Or Field2 DETAILED "%Слово1%") AND (FIeld1
DETAILED "%Слово2%" Or Field2 DETAILED "%Слово2%")
```

The Full-text search property allows to use full-text search when arranging the input by string. Full-text search cannot be used for external data source tables. The use of full-text search has several features:

1. Full-text search is always performed by the beginning of words.

2. Full-text search is used for the initial filtering of objects in which further search will be performed (the number of objects in which the search will be performed is reduced).
3. All the non-indexed references from the table, which was used for the input by string, will be added to the results of the full-text search.
4. In the list of selected objects, the search will be performed by the fields that are selected for use in the input by string functionality in the manner specified by the String search method property.
5. For each word from the search query, the "*" character is added at the end.

These features can cause the following consequences:

- The data will be found if the line search method is set in the value Any part and requires to search the fragments located at the beginning of words.
- The data will not be found if the line search method is set in the value Any part and requires to search the fragments located not at the beginning of words. It is recommended disabling the fulltext search if the line fragments located not at the beginning of works are needed to be found. The line search method may be left unchanged.
- If the full-text search fails, the usual search will be applied, in accordance with the String search method property. This can lead to a situation where the object found turns into a not-found one (for the search by any part of a word). Example: the “a” character is entered in the input field, which is located not only in the beginning, but also in the middle of the array of words in which the search is performed. For a large data set, the full-text search will terminate with an error (the search result is too large) and using the normal search, data will be found that contain the letter “a” inside. Then the user enters another letter (for example, "ц") and the search string will become "ац". In this case, full-text search will only filter data that starts with “ац” and will ignore data that contains “ац” within.

If the drop-down list is generated by 1C:Enterprise (and not using the application program code), then the presentations of the found objects are generated using a special algorithm. The presentation of the found object consists of two parts: presentation of the attribute containing the search string (mandatory part) and additional presentation (optional part).

The presentation itself is generated as follows:

- The field containing the search string is determined:
  - In the case of using full-text search or in the normal search for several words, this field will be the first field from the fields specified in the Input by string property in which the first word from the search string is found.
  - When using the normal single word search, this field will be the attribute in which the search was performed.
- The presentation of a value located in a specific field will be a mandatory part of the presentation of the found object for the drop-down list.

- It is determined whether it is necessary to generate an additional presentation. For this, the components of the fields is determined, which will be used to generate an additional presentation:
  - If the object found has only one field used to generate the presentation, and the search string is found in this attribute - the fields of the alternative presentation are used.
  - Otherwise, the default presentation fields are used. The list of the default and alternative presentation fields is given below.
- If the list of fields to generate an additional presentation is empty - the additional presentation is not generated. Otherwise, the additional presentation is generated as follows:
  - Text presentation of all the fields that are selected to generate an additional presentation is enclosed in parentheses and separated by ",";
  - The generated additional presentation is added to the right, separated by a space, to the mandatory part of the presentation of the object found.
- The following fields are used to generate the default and alternative presentations:
  - Catalog, business process, chart of characteristic types, chart of accounts, chart of calculation types:
    - Default presentation: `Name`, `Code`.
    - Alternative presentation: depending on which default presentation is selected, the remaining field is used for the alternative presentation.
  - Document:
    - Default presentation: `Number`, `Date`.
    - Alternative presentation: none.
  - Task:
    - Default presentation: `Name`, `Number`, `Date`.
    - Alternative presentation: none.
  - External data source:
    - Default presentation: the field specfied in Presentation filed property of the external data source table.
    - Alternative presentation: none.

The Selection data retrieval mode property controls how the platform searches for data when input by string. If the property is set to Background, then a background job is used to retrieve data, and an animated image is displayed in the drop-down list, similar to the background report execution. The background job is used if the required data has not been received for display to the user within 200 ms. For the infobase file mode, it is possible to retrieve data in the background if another background or scheduled job is not performed at the time of receiving the data. During the background retrieval of selection results, the user can continue to edit the input field. By specifying the Directly value as a property value, the search is performed without using background jobs.

When configuring input by string it is recommended to observe the following recommendations:

- The Background value for the Selection data retrieval mode property is recommended to be set for objects where a large amount of data is planned.
- Enabling full-text search is recommended for the objects in which a large amount of data is expected.
- It is recommended to enable the string search in any place alongside with a background search, or in tables, where a large amount of data is not expected. It is desirable, also with a large amount of data, if the search in any place is enabled, to enable the full-text search (so that most of the searches would be performed through full-text search).

5.6.7.2. Input field behavior features

When entering data in the input field, consider some features of operating the selection list. If input results in finding several objects, beginning with the entered text, the list of these objects will be located in the drop-down list. During input, the user can use the Up arrow and Down arrow buttons to move through the list, while continuing to type in the field itself. In this case, to make a selection, it is necessary to select the required element and confirm the selection by pressing the Enter or Tab button. To generate a selection list for the input field, the following algorithm is used:

1. Data is retrieved when performing a search in the first field (from the list of selected fields) by the beginning of a string and by substring. The quantity of records retrieved as a result of the search by the beginning of the string will be called RS, and the quantity of records retrieved as a result of the search by substring will be called RP.
2. If `RS+RP<=10`, the drop-down list will contain all the results of each search.
3. If the RP and RS exceed 5 (each), the selection list will contain 5 records from each list.
4. If any of RP or RS values is less than 5, then all the entries from the smaller list and as many records from the larger list as required to complete the drop-down list to 10 will be added to the drop-down list.
5. If the total quantity of records in the drop-down list turned out to be less than 10, then the algorithm is repeated from step 1 for the next field from the list of selected fields, using the difference between 10 and the quantity of results retrieved as the number of data necessary for retrieval during the next step.

5.6.7.3. Selection list programmatic generation

If the developer is not satisfied with how the selection list is generated, they can override it themselves.

This can be done in two ways:

- Directly in the form - in this case, the special generation of the selection list can only be used for this single field. To do this it is required to define `StartChoice`, `AutoComplete` and `TextEditEnd` event handlers.
- In the corresponding object manager module - in this case, special selection list generation will be performed for all the input fields in which the values of the used object are entered.

If the field is associated with data that has a type of reference to an applied object (for example, `CatalogRef`, `EnumRef`, etc.), the selection list can be generated either in the corresponding object manager module, or using form module handlers. In other cases, the selection list can only be generated using the form module handlers.

More details on the second method are given below.

To programmatically generate a selection list, it is necessary to redefine the `ChoiceDataGetProcessing` event handler of the object module. The handler receives a set of parameters that determine the conditions for selection list generation. A parameter set is a structure that contains:

- Search string - text string containing the text that the user entered in the input field (`SearchString` property).
  When performing standard data processing, the following assumptions regarding the search string are used:
  - if this parameter is not specified or its value is `Undefined`, it is considered that the search string is not specified and it is necessary to return the entire available list of values;
  - a value of any type other than the `String` type, which is specified as the value of the `SearchString` property, will be converted to a string value.
- Filter - structure that describes the filtering as it is described by the `Filter` form parameter of the form extension for a dynamic list. This property is always present.
  The filter specified in this property will be converted (if possible) into a full-text search query, refining this query to increase the search speed.
- The property that indicates the mode of selecting groups and elements (transferred only for hierarchical lists) - is the `ChoiceFoldersAndItems` property.
- A property that defines the string search method: by the beginning of a word or by any part (the `StringSearchMethod` property).
- The need to use full-text search (the `FullTextSearch` property).
- The mode of selection data retrieval (the `SelectionDataRetrievalMode` property).
- In addition, the structure passes the elements, specified in the - Selection parameter associations and Selection parameters form element properties.

In addition, a variable is passed to the handler, which the generated selection list must be returned to, and the `StandardProcessing` parameter, which determines the 1C:Enterprise behavior after the handler is exited.
If the developer sets the `StandardProcessing` parameter to `False`, then they need to completely generate the selection list themselves.
If the standard processing check box is set to `True`, then 1C:Enterprise can be given the possibility to generate a selection list, but it is also possible modify the selection parameters (add additional selection values, change the selection mode for groups and elements, etc.).

NOTE 1. If 1C:Enterprise independently generates a selection list, the data access restrictions are applied when generating the list.
NOTE 2. The examples below are not complete. These are intended to demonstrate the various features for selection list retrieval.

Thus, the following code in the case of any text entered by the user will represent a selection of three products with codes 00000002, 00000003 and 00000004.

Example:

```
Procedure ChoiceDataGetProcessing(ChoiceData, Parameters,
StandardProcessing)

  ChoiceData = New ValueList;
  ChoiceData.Add(Catalogs.Goods.FindByCode("00000002"));
  ChoiceData.Add(Catalogs.Goods.FindByCode("00000003"));
  ChoiceData.Add(Catalogs.Goods.FindByCode("00000004"));

  StandardProcessing = False;

EndProcedure
```

In the following example, the filter generated by the input field will be expanded by setting an additional filter so that the services are not included in the selection list. In this case, 1C:Enterprise performs the selection list generation.

Example:

```
Procedure ChoiceDataGetProcessing(ChoiceData, Parameters,
StandardProcessing)

  Parameters.Filter.Insert("Type", Enumerations.ProductTypes.Product);

EndProcedure
```

As a final example, consider a simplified case of the filter implementation entirely by means of the 1C:Enterprise language. In this example, the filtering of products which names begin with the string entered by the user in the input field will be considered.

Example:

```
Procedure ChoiceDataGetProcessing(ChoiceData, Parameters,
StandardProcessing)

  Query = New Query;
  Query.Text = "SELECT ALLOWED
    | Products.Reference as Product
    |FROM
    | Catalog.Goods AS Goods
    | WHERE
    | Product.Name LIKE &Name";
  Query.SetParameter("Name", Parameters.SearchString + "%");

  Result = Query.Execute();
  ResultsTable = Results.Export();
  ProductsArray = ResultsTable.UnloadColumn("Product");

  ChoiceData = New ValueList;
  ChoiceData.LoadValues(ProductsArray);

  StandardProcessing = False;

EndProcedure
```

Another way to generate a selection list is also worth paying attention to: passing not a reference to the searched object (as in the examples above), but a structure of special content as the value list element value.

This structure consists of the following elements:

- `Value` - the selected element value. Structure element with this name is mandatory.
- `DeletionMark` - flag that the selected value is marked for deletion in the infobase. Structure element with this name is optional.
- `Warning` - the line with the warning text that the 1C:Enterprise will display when selecting such element from the list of values. Structure element with this name is optional.

If the `DeletionMark` property is `True` and `Warning` property is not specified in the structure, 1C:Enterprise will automatically generate the warning text. If the `Warning` property is specified, then it is displayed. Remember, that the `Warning` text ends with the question Continue? and is displayed as a question with Yes and No options.

Next, a modification of the previous example will be shown, where for warehouses with the `DontUse` attribute set to `True`, a This warehouse must not be used warning will be generated.

It is also possible to combine both values of `Structure` type and simple values in a single selection list.

Example:

```
Procedure ChoiceDataGetProcessing(ChoiceData, Parameters,
StandardProcessing)

  StandardProcessing = False;
  ChoiceData = New ValueList;

  // Generate a list with warnings
  Query = New Query;
  Query.Text = "SELECT
    | Warehouses.Reference,
    | Warehouses.Name,
    | Warehouses.DontUse
    |FROM
    | Catalog.Warehouses AS Warehouses";

  Result = Query.Execute();
  SelectionDetailedRecords = Result.Select();

  While SelectionDetailedRecords.Next() Do

    Structure = New Structure("Value",
SelectionDetailedRecords.Reference);

    // Fill in the warning
    If SelectionDetailedRecords.DontUse Then
       Structure.Insert("Warning", "This warehouse must not be used!");
    EndIf;

    Element = ChoiceData.Add();
    Element.Value = Structure;
    Element.Presentation = SelectionDetailedRecords.Name;

  EndDo;

EndProcedure
```

NOTE. If element presentations (including the structure elements with the `Value` name) are not specified explicitly, these will be retrieved automatically.

5.6.8. Forms

Form - is an object created for any information input and viewing, as well as for managing various processes. By using forms, an application asks the user to input a piece of information that is necessary for it to execute its function, or provides information to the user for viewing and editing.

In cases where it is necessary to ensure that the configuration functions simultaneously in the ordinary and managed modes, additional metadata object forms can be used. In this case, 1C:Enterprise will automatically select the form that corresponds to the current mode of operation.

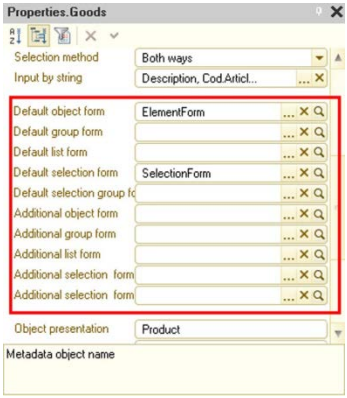


Fig. 110. Main and additional forms

The thin client and web client operate managed forms only. That means that:

- if both assigned forms - are managed, then the form that is assigned as the main one will be opened;
- if there is only one managed form among the assigned forms, it will be opened;
- if the managed form is not assigned, it will be generated automatically.

When selecting a form, a thick client attempts to choose a form that most closely matches the current operation mode:

- if no forms are assigned, the following will be generated:
  - in ordinary mode - ordinary form;
  - in managed mode - managed form;
- if only one form is assigned, then it will be opened;
- if two forms are assigned, ordinary and managed one, the following will be opened:
  - in ordinary mode - ordinary form;
  - in managed mode - managed form.
- if two ordinary or two managed forms are assigned, the form that is assigned as the main one will be opened.

A double set of forms can be used in cases when the configuration is transfered from unmanaged to managed mode, or it is necessary, for example, to make some configuration options available in a web client (or thin client). In this case, it is possible to implement the necessary features in managed forms and specify them as additional forms. Then, when operating in the web client (or thin client) mode, the necessary forms (managed ones) will be used.

However, it should be noted that the Use managed forms in an ordinary application and the Use ordinary forms in a managed application configuration properties affect the retrieval of default forms in a thick client:

- If the Use managed forms in an ordinary application configuration property is `False`, then when receiving the default form in the ordinary mode in the thick client, the ordinary form must be retrieved. If neither main nor additional form is ordinary, then the ordinary form is generated.
- If the Use ordinary forms in a managed application configuration property is `False`, then when receiving the default form in the managed mode in the thick client, the managed form must be retrieved. If neither main nor additional form is managed, then the managed form is generated.

If as a result the form that is automatically generated by 1C:Enterprise is opened and the client application startup mode differs from the startup mode set in the Main startup mode configuration property, then the object forms and record set forms open in the View only mode. Also, in the list forms and object forms, there will be no Create based on commands , and in the list forms, - there will be no the Create command, if 1C:Enterprise requires to automatically generate a form suitable for the current startup mode to execute these commands and the startup mode of the client application differs from the startup mode set in the Main startup mode configuration property.

If the managed application is selected as the main startup mode of the application, then for each managed form of the application, it is possible to choose the purpose of the form. The Purpose of use form property is equivalent (by contents and use) to a similar configuration property. The form editor behavior depends on the intersection of the configuration and the specific form purpose set.

If there is no Personal computer value in the intersection, then:

- Attribute, parameters, commands, elements properties that are not supported by the mobile platform are not available in the property palette;
- Editing the context menu of a form element is not available;
- Editing the extended tooltip for a form element is not available;
- The types of form fields that are not supported by the mobile platform are not available;
- The dynamic list settings editor does not allow to expand the available fields (this is due to the fact that the dynamic list operates with one table only on the mobile platform).

If there is no Mobile application value in the form purpose, then it is not included in the mobile application configuration.

5.6.9. Command

There are commands for the object to perform operations related to a specific metadata object. Non-parametized object commands will be available in the command interface of the subsystem that include metadata object. In case the

command is parametized, it will be available in the forms that contain form attributes (including first-level subordinate attributes of the main form attribute) of the same type as the command parameter.
The command execution procedure is required for the commands. To create it, use command module, in which you have to implement the `CommandProcessing()` handler. Precede this procedure by the `&AtClient` directive, as commands are executed in the client application. However, other procedures and functions located in the command module can be preceded by other directives if this is required for command execution.
Command module must include, for example, opening the report form with the pre-set parameters for printing the card of a specific accounting report or opening the products list form with the pre-set filtering by the product type. If the command from the Navigation panel group must open the same form in different tabs, then when receiving (or opening) the form you must specify the uniqueness key or the key form parameter.

NOTE. If the command is located on the navigation panel of the main application window, the repeated selection of this command will not lead to its repeated execution. This is true for standard commands and commands created in the configuration.

5.6.10. New object attributes entering mechanism

New object attributes can be entered: during interactive creation (except copying); when generating; when using `OpenForm()`/`GetForm()` methods; or when calling the `Fill()` method explicitly. The following can be entered:

- filter values when entering from the list:
- specific values entered in attribute properties of the configuration (filling values);
- values that the developer specified in the `Filling()` event handler.

Filling processing mechanism is implemented in the following objects:

- Exchange plans
- Catalogs
- Documents
- Charts of characteristic types
- Charts of accounts
- Charts of calculation types
- information register record sets,
- Business processes
- Tasks

To get a filling data the FillingProcessing() handler has the Filling() parameter. Depending on the way the handler was called, the FillingData parameter value might be as follows:

- Generation - the link to source object is transferred as the parameter value. The value of FillingData parameter will contain the link to the source object.
- Entering from the list with the set filter - the parameter value is the structure composed of the filter elements with the Equal to or In list comparison view (the list includes only one value). When creating a new document from the document journal form, the elements of by-column filter are preliminary transformed in such a way that the FillingData structure element name is document attribute name, not the journal column name.
- Entering new object or record without filter - the parameter value is Undefined .
- Entering new object or record with global command - the parameter value is Undefined .
- Programmatical execution of OpenForm()/GetForm()methods - when executing OpenForm()/GetForm() methods leads to creating new object form, the FillingData handler parameter value will be transferred as a value (Structure type) of FillingValues form parameter. If the FillingValues parameter is not specified in the form parameters, the handler parameter value will be Undefined.
- Programmatical call of Fill() object method - the information that was transferred as Fill() method value is transferred as the parameter value.

The interactive object creation operation is processed in OnCopy event handler of the object. When processing object creation via copying, remember that ValueStorage-type attribute will not be transferred automatically to the object created. Transferring such data must be explicitly described in OnCopy event handler of the object created.

Filling attributes from filling data is affected by the Fill from filling data property of the metadata object attribute. If this property is set to True, the attributes will be automatically specified by 1C:Enterprise from the filling data. If this property is set to False or the filling data lack the required value, 1C:Enterprise will try to fill the attribute from Filling value property.

IMPORTANT! The developer can control filling of standard object attributes just like the custom made attributes. For example, you can prohibit filling the standard Parent attribute and it will not be automatically filled by the current group.

If during the FillingProcessing() handler execution the StandardProcessing parameter is set to True, 1C:Enterprise will automatically fill the attributes (including the standard ones), for which the Fill from

filling data or Filling values properties are specified in metadata and which are not filled in the handler (the value is the default one for this attribute type). The Fill from filling data value is automatically specified by 1C:Enterprise for some standard attributes of a number of objects:

- For catalogs - `Parent` and `Owner` attributes;
- For charts of accounts, charts of characteristic types, and charts of calculation types - `Parent` attribute;
- For information register - leading dimensions;
- For standard attributes of other objects this property is not set automatically.

1C:Enterprise takes the data for filling by default from data attributes of the same name transferred in `FillingData` parameter.
The filling data are transferred into the created object form as a standard parameter of `FillingValues` form and are sent from this parameter by the form extension for filling by the object. You can also specify the `FillingValues` form parameter of the new object programatically, and all actions performed during interactive object creation will be performed, too.
When creating new object programatically, the processing of filling by 1C:Enterprise is not triggered automatically. To call the filling handler, use `Fill()` method.
Filling value - it is a property of metadata object attribute that allows to specify the default value for the attribute during the interactive object creation.

IMPORTANT! The attributes of Filling values property are filled after the `FillingProcessing()` handler is called. The attribute will be filled if its value was not filled before (in `Filling()` handler or by the standard filling mechanism).

The filling value type is similar to the attribute type. The value of primitive types or predefined data can be specified as the value of this property.

5.6.11. Attributes fill checking

5.6.11.1. General information

The data in the information system can be entered in a number of ways and these data are often incorrect. That why when developing a solution you sometimes need to put a lot of effort in implementing data input verification and notification on the incorrect information entered.
The fill checking mechanism allows to simplify configuration development significantly.
The platform supports automated checks of the specified applied object attributes and forms. It also allows to check this in the module. To manage attribute filling checks you can use the Fill check property. If the property is set to Don't

check - the platform doesn't check the attribute filling. The Display error value specifies that the check is needed.
The platform automatically checks the filling of:

- constants;
- catalogs, documents, reports, processings, charts of characteristic types, charts of accounts, charts of calculation types, business processes, tasks:
  - attributes and standard attributes,
  - tabular sections,
  - attributes and standard attributes of tabular sections;
- register record set from accounting, information register, accumulation register, calculation register, recalculations, sequences:
  - dimensions,
  - resources,
  - attributes and standard attributes;
- form attributes;
- report form attributes;
- processings form attributes.

NOTE. Attributes filling check is performed similar to the `ValueIsFilled()` function. The filling check of tabular sections implies that the tabular section is considered to be filled when it contains at least one row.

The filling check can be triggered in two ways:

- Calling the `CheckFilling()` method (of the object or form). When calling an update check programmatically using the `CheckFilling()` form method, the form attributes filling is checked. This includes converting values of the main form data attribute into an object and checking it.
- Automatically. The algorithm of automated checking is described further.

IMPORTANT! If the Compatibility mode configuration property is set to Version 8.1, the automated filling check doesn't work.

5.6.11.2. Default settings

The property of Fill check attribute is set to Display error by default for the following standard attributes:

- `ExchangePlan` - `Name`;
- `Catalog` - `Owner`, `Name`;
- `Document` - `Date`;
- `ChartOfCharacteristicTypes` - `Name`;
- `ChartOfAccounts` - `Code`, `Name`;
- `ChartOfAccounts.ExtraDimensionTypes` - `ExtDimensionType`;
- `ChartOfCalculationTypes` - `Name`;

- ChartOfCalculationTypes.LeadingCalculationTypes - CalculationType;
- ChartOfCalculationTypes.BaseCalculationTypes - CalculationType;
- ChartOfCalculationTypes.DisplacingCalculationTypes - CalculationType;
- InformationRegister - Period;
- AccumulationRegister - Period;
- AccountingRegister - Period;
- CalculationRegister - RegistrationPeriod, CalculationType, BegOfActionPeriod, EndOfActionPeriod;
- BusinessProcess - Date;
- Task - Name.

5.6.11.3. Operations

The automated fill checking is called by form extension before the interactive writing of all objects except documents, business processes, reports and data processors. For these objects the following behavior is set:

- For documents:
  - Filling check is called by the form extension before posting if the Posting property is set to Enable;
  - Filling check is called by the form extension before writing is the Posting property is set to Disable;
- For business processes the filling check is called by the form extension before the start;
- For reports the filling check is called by the form extension when the Generate button is clicked;
- For processings the filling check is called by the form extension when standard form command buttons—OK, Yes, Retry, Skip—are clicked.

If the attribute is the main form attribute of the following types: CatalogObject, DocumentObject, ReportObject, ChartOfCharacteristicTypesObject, ChartOfAccountsObject, ChartOfCalculationTypesObject, BusinessProcessObject, TaskObject, the filling check will also be called for the value of such attribute.

To call the system filling check, the form (which you currently operate) must have the Auto fill check property set. In this case, the FillCheckProcessingAtServer() form handler will be called first, and then the FillCheckProcessing() object module handler will be called.

IMPORTANT! If the Auto fill check property of the form is set to True, than when you trigger standard Write command (Post for documents and so on) and standard form commands—OK, Yes, Retry, Skip—the CheckFilling() method will be called. Otherwise, the filling check is not called for the form or the object.

The filling check is done in the following way:

- The list of form attribute names, for which the filling check is possible and for which the FillChecking property is set to ShowError. This list will not include attribute names with the type not supporting filling check (for example CatalogObject), but the main form attribute name will be included.
  The automatically generated attribute list will not include form attributes disabled with the functional options.
- The event handler of the FillCheckProcessingAtServer form will be called, in which the developer can describe his or her own filling check algorithm or change the content of the attributes checked. The generated list of attribute names will be transferred to the handler. If you need to add some attributes to the list in the handler, this is only possible for the attributes of the above-mentioned types (for which the filling check in the form is possible) and the main attribute. Adding the object-type attribute names (for example, CatalogObject) to the list will cause exception during the further automated checking. Adding the name of the non-existent attribute to the list will cause exception during further automated checking.
- After the event handler finished operation, the filling check mechanism gets back the list of names checked by the attribute (which was possibly changed in the handler). 1C:Enterprise analyses the attributes list and checks the filling of each attribute. If the attribute is the main object-type attribute (for example Object of CatalogObject type), the filling check procedure for the object itself will be called. If the attribute is the object-type attribute the exception will be called.

The developer can impact the checking process by defining object in the module, the records set and FillCheckProcessing event handler in the form module.

Having defined the event handler, the developer gains full control of the filling check. In CheckedAttributes parameter the handler gets the array of attributes, which are specified as the ones to be checked in the Designed. The developer can modify the array freely:

- delete the attributes that are checked manually or which, as he or she believes, don't require the checking right now;
- add the required attributes to be checked.

In case the developer is not satisfied with the standard checking procedure, he or she can write a custom checking algorithm and use `UserMessage` object to tell the user about the problems.
If during the checking the developer displayed error messages to the user, set the `Discard` parameter to `True` to notify the platform that the current action can be finished.
`CheckedAttributes` event parameter contains attribute names in the following format:

- For attributes and constants - `AttributeName`, for example, `Provider`;
- For tabular sections - `TabularSectionName`, for example, `Goods`;
- For tabular section attributes - `TabularSectionName.AttributeName`, for example, `Goods.Products`;
- For form attributes - `AttributeName`, for example, `DocumentObject`.

For attributes included into functional options without parameters, the option value is considered when checking the filling. If the functional option is enabled, the attribute will be included into the list of checked attributes. If the option is enabled, the attribute will not be included into the list of checked attributes. This means that the disabled field will not be transferred into `FillCheckProcessing` via the `CheckedAttributes`.
The attributes included into the functional option with the parameters are always included in the checked attributes list (`CheckedAttributes` parameter). In this case, the attribute must be deleted in `FillCheckProcessing` handler. To do this, get the functional option value by specifying the required object data as a parameters.

5.6.11.4. Rules for displaying the unfilled mark

The display of the unfilled mark is affected by the following configuration element properties:

- Fill check of the attribute of the standard metadata object attribute;
- Fill check property of the form attribute;
- Automark incomplete property of the form element;
- `MarkIncomplete` property of the form element (only available for programmatical change).

Automark incomplete form element property can be set to:

- Auto - the mark will be displayed if the form attribute or metadata object attribute (standard attribute)has Filling check property set to Display error and the attribute associated with the form element contains the empty value.
- Yes - in this case the platform controls mark display. The mark state (displayed/hidden) is located in `MarkIncomplete` property filled by the

platform. Programmatical change of the MarkIncomplete property is ignored.

- No - the incompleteness mark is only displayed is the MarkIncomplete of the form element is set to True.

When starting the interactive change of the input field, the incompleteness mark is not displayed anymore, and the MarkIncomplete property value is not changed. The mark display settings are restored (if necessary) after the interactive editing is finished or the value is set in the input field.

NOTE. If the table MarkIncomplete property is set, then in the table with no rows, the first row will be highlighted, and in the filled table - all rows.

Besides, the incompleteness mark will also be displayed if the message window has the message connected to the form field, regardless of the form element properties set and attribute associated with it. After clearing the message window, the incompleteness mark is unset for the form elements, which has MarkIncomplete property set to False.

5.6.12. Indexing object attributes

The most applied objects of the configuration has the Attributes group included into the subordinate objects. Additional object characteristics are specified in this group.

In 1C:Enterprise mode the data often has to be filtered by some attribute value or the data lists has to be sorted by attributes. The 1C:Enterprise tools allow this; however, if there are quite a lot of data, the task might take much time.

To speed up the process, set the Index property to the attributes used to filter and sort. If the property is set (Index or Index with additional order value is selected) such tasks will be more efficient. For primitive types of attributes, setting the indexing provides the user with the tool to sort list with a click of a mouse in column title area.

Along with sorting by attribute or filtering the data by value of some attribute, it's often required to additionally sort the data in the resulting list by the main presentation (name or code), i.e. within one attribute value the records must be sorted by presentation. To get the right result, choose Index indexing value and in list sorting conditions specify the attribute and the presentation.

If you need to minimize the time of such sorting and filtering, set the Index with additional order indexing value for the attribute (if such setting is possible).

IMPORTANT! The additional sorting mechanism will only be used efficiently when the 1C:Enterprise mode has the specific sorting order specified in the sorting conditions: firstly be attribute, then by presentation.

In case such sorting conditions are not set, then using Index with additional order value is pointless, because it will be equivalent to the regular indexing but with larger index size.
Indexing with additional order is used for attributes of catalogues, documents, charts of characteristic types, charts of accounts, charts of calculation types.
The regular indexing is allowed for the register attributes.

5.6.13. Rights

Rights for accessing the configuration objects can be edited using the role access rights editor and object editing window.
In object editing window you can edit access right for the current object in all roles presented in 1C:Enterprise.
The acces rights can be set for:

- metadata objects themselves,
- tabular sections,
- object attributes.

5.6.14. Quick choice

The approach to choosing objects when filling input field is set by Choice method and Quick choice properties.
The Quick choice property is responsible for the default selection mode. The property is only available when the Choice method property value is set to Both methods. Let's look at the example of Quick choice properties operation.
For example, we have a `Warehouses` catalog. It's Choice method property is Both methods and the Quick choice property is set too. In this case, by default, the choice from values in `Warehouses` catalog will be performed in fast choice mode in all configuration. When disabling the Quick choice property, the default property will be From form.
You can also redefine the choice mode explicitly for the metadata object attribute and/or form element. To do this, change the Auto value of the Quick choice property to Use - to enable the fast choice and Don't use - to disable the fast choice. By default the Quick choice property of the metadata object attribute and form element is set to Auto.
Thus, there is a three-level choice management scheme:

6. Firstly, the platform analyses the Quick choice property of the form element.
7. If the property equals Auto, it's analyzed for the object attribute.
8. In case the Auto properties are the same, the Choice method and Quick choice properties of the applied object that corresponds to the attribute type is analyzed. If at some of the first levels (form element and object attribute)

the value of Quick choice property is different from Auto, the analysis is stopped and the choice is made using the defined mode.

NOTE. If the Choice method property of the applied object equals From form or Quick choice (but not Both methods), setting the Fast choice properties of the attribute and form element doesn't affect the choice method.

5.6.15. Choice history during the input

In case the form input field is associated with the reference-type data (catalogs, documents, enumerations, business processes, tasks, charts of characteristic types, charts of calculation types, charts of accounts and exchange plans) the view management and creating the choice history when inputting is available for it. The management is based on the `ChoiceHistoryOnInput` property value that is present in the properties of the configuration object, configuration object attributes and the managed form input field.

The choice history management is implemented using the three-level management scheme:

1. Firstly, the platform analyzes the `ChoiceHistoryOnInput` property for the input field.
2. If the property value is Auto, the `ChoiceHistoryOnInput` property of the attribute displayed by the input field is analyzed.
3. If the property value is Auto, the `ChoiceHistoryOnInput` property of the configuration object that correspond to the attribute type is analyzed.

The `ChoiceHistoryOnInput` property value can be Auto or Don't use. In case the property value is Auto on all decision making "levels", the choice history will be displayed and saved if the following conditions are met:

- The reference-type value is displayed after the input
- The list selection mode is not set
- Quick choice is not used.

In all other cases the choice history will not be displayed and saved.

The choice history includes 7 values:

- No more than 5 of the most frequently used values;
- The last chosen values in such number that the total list doesn't include more than 7 elements.

If some of the last chosen values are in the frequently used list, the previous chosen value is displayed. If all the chosen values in the input field are more or less equally probable the choice history only include 7 of the last chosen values. When identifying the use frequency both chice frequency and the time passed after the last choice are considered.

5.6.16. Creating new item on input

In case the form input field is associated with the reference-type data (catalogs, documents, enumerations, business processes, tasks, charts of characteristic types, charts of calculation types, charts of accounts and exchange plans) it has the management of new object creation from input field available. The management is based on the `CreateOnInput` property value that is present in the properties of the configuration object, configuration object attributes and the managed form input field.

Management of the ability to create the item from the drop-down list is implemented using the three-level management scheme:

1. Firstly, the platform analyzes the `CreateButton` property for the form input field.
2. If the property value is Auto, the `CreateOnInput` property of the attribute displayed by the input field is analyzed.
3. If the property value is Auto, the `CreateOnInput` property of the configuration object that corresponds to the attribute type is analyzed. .

The analysis is stopped after the value that differs from Auto is identified on some level.

In case the item creation form the input field has to be done in a specific way, the developer must handle the `Creation` event. The event is happening when clicking the create button in the drop-down list.

If in the current event handler the `StandardProcessing` parameter is set to `False`, the system will not take any actions and it's expected that the developer will open the new item form. In case the `StandardProcessing` parameter is set to `True`, 1C:Enterprise will open the new object form, and the following parameters will be transferred to the form:

- `ChoiceMode`, `Boolean` type, value- `True`. The flag showing that the form is in the choice mode.
- `FillingText`, `Row` type. The parameter contains the text, entered in the input field before clicking the button to create new object.
- `FillingValues`, `Structure` type. The filling parameters of the object filled. The structure is filled from choice parameters and choice parameter connections. The structure only includes the values with the names starting with `Filter.`.
- `ChoiceParameters`, `Structure` type. The parameter value is the complete copy of the `FillingValues` parameter.
- `AdditionalParameters`, `Structure` type. The additional parameters that the application developer considers worth transferring to the object form. The structure is filled from choice parameters and choice parameter connections. The structure only includes the value with the names starting with `Additional.`.

The owner of the new object form opened is the input field, from which the object creation is called.
Choice parameters and parameter connections limit the list of current items, which can be chosen in the input field. There is a special mechanism, which ensures that the new item created from the input field complies with the limits:

- When opening the new object form, which was initiated from the input field, the `ChoiceParameters` form parameter value is copied into the managed form property of the same name.
- In case the item if created in the choice mode (`ChoiceMode` managed form property is set to `True`), then before calling the `AfterWriteAtServer` managed form event handler, the compliance of the created object attributes to the required parameters (`ChoiceParameters` form property) is verified. The verification results are used to set the `ChoiceAvailable` managed form property.
- Before closing the form (before calling the `OnClose` event handler) the `ValueChoice` event handler is called. The result of calling the handler (`StandardProcessing` parameter value) and `ChoiceAvailable` property, the decision on further action is made:
  - The `ValueChoice` event handler is not set of the handler is set and the `StandardProcessing` parameter after the hander operation is set to `True`:
    - The `ChoiceAvailable` property value is set to `True` - the choice of the created item is made;
    - The `ChoiceAvailable` property value is set to `False` - the message is generated on the created item not complying with the choice parameters and the created item is not chosen.
  - The `ValueChoice` event handler is set and after the handler operation, the `StandardProcessing` parameter is set to `False` - the created item is not chosen and the message is not generated. The developer must take the required actions himself/herself: display the diagnostic message, specify the default value in the input field and so on.

  If the created item is chosen:
  - The item form is closed;
  - The `ChoiceProcessing` event is called for the input field, whose drop-down list was used to create the item .

  If the choice of the created item is not made, the item form is closed and np further actions are taken.

5.6.17. Choice Parameter Links

Choice parameter links - this property allows to specify the list of attributes that will provide values used when choosing the attribute value, opening the choice form, displaying quick choice list and when inputting by row.
A contract with the contractor can be used as an example. Firstly, the contractor is chosen, then the contract is chosen from the list of contractor contracts. The choice changes automatically when changing the contractor.
To limit the choice, the Choice parameter links property value of the attribute is set, which corresponds to the attribute name, which will be used to filter the chosen values and the attribute. from which the filtering value will be taken.

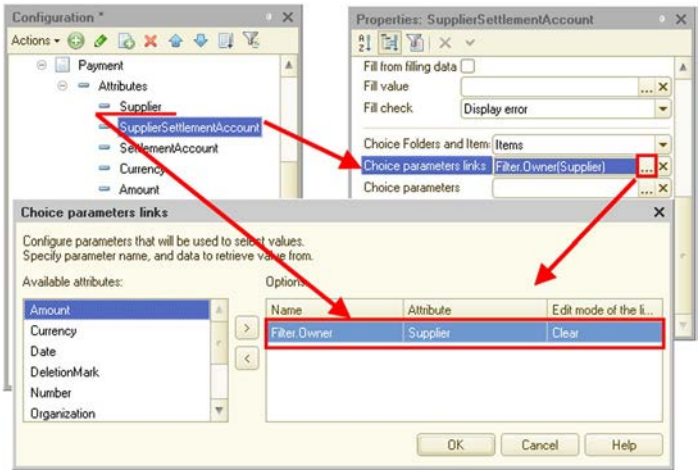


Fig. 111. Choice Parameter Links

Values specified in this property will be transferred to the opened form via the `Parameters` structure. The Name column value will correspond to the structure element key and the attribute value stated in Attribute column - to the structure element value. If the Name column includes the Filter.Owner-type value, the `Filter` form parameter will be created (`Structure` type). In this structure the item will be created with Code key and the value received from attribute specified in Attribute column (in the example - Provider).
If the name of some Choice parameter links property element is the same as name of some Choice parameters property element, the priority is given to the element from Choice parameter links property in case the value of the field specified in Choice parameter links element and sent to `ValueIsFilled()` function returns `True`. The element from Choice parameters property is ignored.

Also, you can set the of field clearing when changing link fields in the choice parameter links editing window. If the value of Linked value change mode property is Clear, the field will be cleared during the interactive changing of link value (the repeated choice of the value already present in the field is also considered changing) before the OnChange event has occurred. Otherwise (the property value is Don't change), the field will not be cleared. The clearing happens irregardless of the real value change in the link element and is made before calling the OnChange event handler.

If the field displays the table data (table columns or separate field connected to the current data), the clearing of such field is made is the data source of the form table is FormDataCollection or FormDataStructureAndCollection. If the attribute that requires clearing is connected to the table data and the link attribute is not a piece of the table data, the values in all rows with the attribute, for which choice parameter links are set, is made. The attribute can't be cleared if it's connected to the DynamicList -type attribute column.

For the standard Parent attribute of the subordinate catalog, the situation is possible, where the Choice parameter links property will be changed automatically by 1C:Enterprise. This change happens in the following cases:

- when the catalog changes subordination status;
- when the Hierarchical catalog property of the catalog is changed.

This change happens in the following way:

- If the catalog is subordinate and hierarchical then for the Parent attribute the Choice parameter links property is supplemented with the following parameter: Name: Filter.Owner - Attribute: Owner;
- Otherwise, the Name: Filter.Owner- Attribute: Owner parameter is deleted from the Choice parameter links property parameters list of the standard Parent attribute.

Consider an example: The form contains the `Provider` field and the `ProviderCurrentAccount` field. For the `ProviderCurrentAccount` field, the `ChoiceParameterLinks` property is set to `Object.Provider` value, which will be set in the `Owner` filter field.

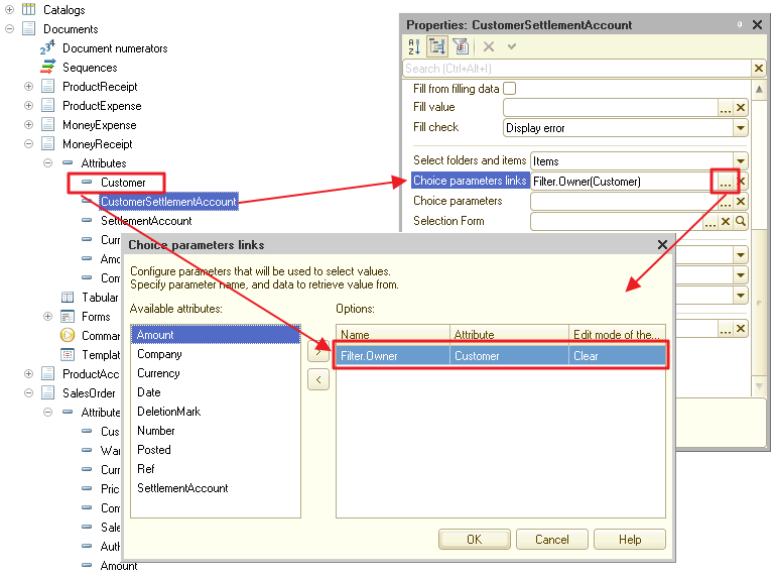


Fig. 112. Clearing connected form items

Then when changing `Provider` field value interactively, the `ProviderCurrentAccount` field will be cleared automatically.

5.6.18. Choice parameters

Choice parameters - this property allows setting the parameter values, which will be used when choosing the attribute value. Choice parameters will be performed when opening the choice form, when displaying the quick choice list and when inputting by rows. For example, the choice has to be limited by the goods with `Type` attribute equal to `Enumeration.GoodTypes.Service` value. The values list can be specified for the specific filter. To do this, choose Fixed array type when editing the Value column.

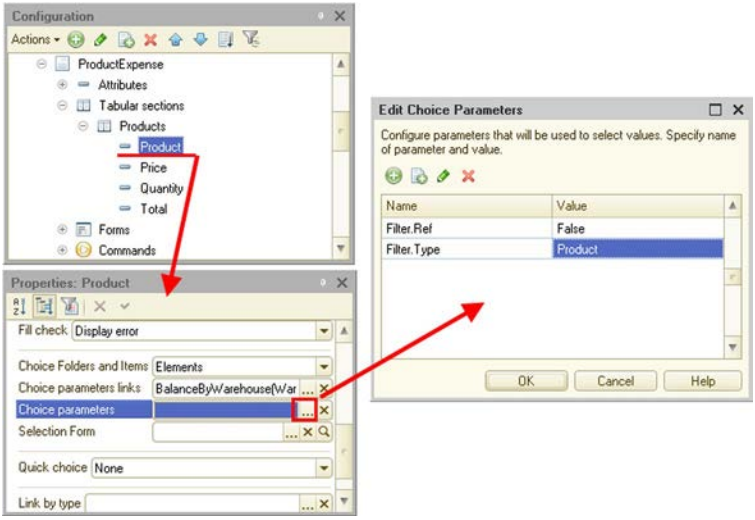


Fig. 113. Choice parameters

Values specified in this property will be transferred to the opened form via the `Parameters` structure. The Name column value will correspond to the structure element key and the Value column - to the structure element value. If the Name column includes the Filter.Code-type value, the `Filter` form parameter will be created (`Structure` type). The item with Code key and Value value will be created in this structure.

If some parameter name of the Choice parameter links property corresponds to the name of some parameter of Choice parameters property, the priority is given to the element in Choice parameter links property (the parameter from Choice parameters property us ignored). The priority will be given in case the value of the field, specified in `Attribute` parameter of the Choice parameter links property is filled (`ValueIsFilled()` function for this value returns `True` value).

NOTE. Values stated in the Choice parameters attribute property are also used by the dynamic list and data composition system.

When choosing an extra dimension value in the cell of the table connected to the accounting register record set, the choice parameter for Owner field is only set automatically if this type of extra dimension has the catalog type of the extra dimension additional values. The value of choice parameter is the correspondent extra dimension type.

5.6.19. Miscellaneous

The Use standard commands property defines the ability to include standard commands (for example, catalog list opening commands) of this object into a command interface. If the value of the property is False, the standard commands will not be displayed by 1C:Enterprise and you can only include the object into the interface using the command created in the Designer.

The Choice form property defines the form that will be used to choose the attribute value. Used for the attributes with the types created by the metadata object that has the form concept. For example, the Goods catalog has several choice forms and for the standard Parent attribute the special form must be used. To implement such behavior this special form must be specified in the Choice form property of the standard Parent attribute.

Format - for Number, Boolean, and Date types allows setting the format of data display on output.

Editing format - the property that allows setting the format for editing the data of Number, Boolean, and Date types.

Type link - established connection with the attribute, limiting the type of values input for the input field. Setting the link by type is reasonable for the attributes with the compound data type that are logically connected with another ChartOfCharacteristicTypes.Link-type attribute, including connection with the attribute containing extra dimension with the attribute, containing values of Link to chart of accounts type. Link by type item contains the number of extra dimension type in case the attribute, which is linked by type, has the values of Link to chart of accounts type.

For example, there are two attributes - CharacteristicType and Characteristic. The CharacteristicType attribute - ChartOfCharacteristicTypesGoodsCharacteristicTypes.Link type; the Characteristic attribute - Characteristic.GoodsCharacteristicTypes type. The Type link - CharacteristicType attribute can be set in the Characteristic attribute properties. In this case the type of the value chosen will be defined by the type, set for the chosen chart of characteristic types value.

5.7. Constants

5.7.1. General information

The 1C:Enterprise constants are designed to store persistent or conditionally persistent information. Such information either doesn't change at all during the business operation or changes rarely. The most simple example of such information - organization name that is usually not changed.
The main reason for using the constants is to write some information to a constant and then repeatedly use it in documents, calculations, and reports. The constant value can be edited from time to time.
Let's look at the example. Corporate documents often include CEO and chief accountant signatures. Naturally, these employees must be the ones that use their signatures. However, the documents also need a transcript - the last name of a person who sighned the document. Of course, names of CEO and chief accountant can be added directly to the document forms. But some name can change and then you will have to edit all form and change names all over again. This is why it's much easier to create two constants in configuration - for storing names of CEO and chief accountant to enter the names into constant once and then use contant names to get their values. When changing the CEO of accountant - simply enter new names and alll changes will be automatically applied to where the contants are used.

IMPORTANT! When changing the constant value, the previous value is lost. To get previous values instead of the constant, use information register without changes.

The 1C:Enterprise Designer allows to create almost infinite number of constants for storing any information.

5.7.2. Constant properties

The Constants branch of the configuration tree is designed for operating constants.
The constant properties are edited in the property palette. Please note that constant is the typified configuration object.
Type - describes the constant type. The constant type can be any of the standard configuration types (`Date`, `Number`, `CatalogRef`, `DocumentRef` and so on) or compound (consisting of several types).
Depending on the chosen type, the property palette may include additional properties specifying the type chosen.

If the constant editing form has to be displayed in the command interface, set the Use standard commands property. In this case the command for opening constant editor will be displayed in subsystems, to which the constant belongs. The constant editing form is set using the Main form property.
The forms for constant input are created in the General forms branch.

5.8. Catalogs

5.8.1. General information

When filling some document form it's oftten required to specify the information by selectim from the list.
Let's take the job application form as an example.
When filling the Place of birth row you have to specify the city. The total number of cities is quite large, but the list of cities is limited. In face, the place of birth can be specified by choosing from such list. This list is the catalog.
Thus, the catalog is the list of possible values of this or that document attribute (document in general).
The catalogs are used to prevent the unambiguous input of information.
For example, to ensure that the customer, sales manager, storekeeper and director understand what good are they talking about. Everyone have to call it concisely. The catalog is needed in this case. In commerce, it's usually a price list and if such catalog is stored on the computer, all products and services that the company sells if included into the list.
1C:Enterprise allows keeping practically unlimited number of catalogs. Each catalog is the list of object instances of the same kind: employees, organizations, goods and so on. Each object instance will be called the catalog item.
Keep in mind that the catalog structure is created in the configuration and its content itself - catalog items- is entered by the users. When configuring, the structure of information to be stored in the catalog is described, the desktop and, if needed, the print presentation of the catalog is developed, and various behavior traits are set.
As a rule, catalogs has the standard attributes, code and name,. Code can have `Number` or `Row` type.
1C:Enterprise provides various abilities for operating catalog item codes: automated code assignment, automated uniqueness control and others.
The 1C:Enterprise catalog can be hierarchical. There are two types of hierarchies: **folder and item hierarchy** and **item hierarchy**. In the first case, all catalog data will be divided into two kinds: "simple" catalog items and catalog folders. **Folders** - logic combination of the catalog items. The example of hierarchical catalog is the goods catalog where the folders are types of goods (Sanitaryware, Household chemical goods and so on) and items- specific goods (Mixer, Mirror, Detergent).

Hierarchical catalogs allows organizing information input with the required level of detail. Items and folders in the hierarchical catalog can be transferred between groups.
For catalogs of the Hierarchy of items type, there are no folders as a separate type. The items themselves play their role. The distinctive feature of such catalogs is functionality of all elements. The example of such catalog is departments catalog. Each department is described with the same set of attributes and it can logically contain another department or be the part of some department.
For the hierarchical catalogs, the Designer allows setting the limit on the number of catalog levels. The unlimited levels of nesting are allowed, too.
Apart from code and name, the set of attributes for storing additional information on the catalog item can be created.
For example, the `Contractors` catalog may contain such data as the full contractor name, tax ID, CEO and chief accountant last name and other information.
If the object of subject domain, corresponding to the catalog, has simple properties (for example full name or tax ID) and complex (list) properties, the set of tabular sections can be created for the catalog.
For example, in the `Contrators` catalog, the tabular section can be created for the contractor's phone numbers.
The attribute names must not match any of the attributes name from a tabular section.
To operate the information in the catalog, a screen form can be created. Separate form can be created to view the catalog list items, edit catalog items, choose the required item from the catalog.
The Designer allows creating several form of the same type, for example, the form for choosing the catalog item, and using different forms in different cases.
It's recommended to create different forms for displaying item list and choosing catalog items.

5.8.2. Catalog properties

The Catalogs branch of the configuration tree is designed for working with the catalogs.
When creating the new catalog, an object editing window opens.
Hierarchical catalog - if the property is set, the catalog has hierarchical structure and Hierarchy type and Limit of hierarchy levels properties become available.
Hierarchy type - the type of hierarchy in this catalog si defined. When choosing Folder and item hierarchy type there are two types of items in the catalog: folders and items. The folders combine other groups and catalog items. Usually, code, name, and parent (link to the higher level) is enough to describe a folder. The catalog item might include other attributes, specified on the Data tab. For the catalogs with this hierarchy type, folder forms and item forms can be created.

When choosing the Hierarchy of items type, all items in the catalog are equal. The example of such catalog is the department catalog and heads of expenditures.
If the Folder and item hierarchy type is set for the catalog, then when creating an item or folder form, the form elements connected to the attributes belonging to the item and folder can be located on the form (no restrictions). The form will allow editing data without checking the usage option set for the attribute. When writing an object, only data of its usage option will be written: for the item form, the item attributes and general attributes will be written; for folder - folder attributes and general attributes. No error messages are displayed in this case. When opening the form again, the form elements, connected to "other" attributes will be filled with the default values.
Place folders on top - the value is available when the Folder and item hierarchy value is set. If the Place folder on top property is set, when you display the catalog as a hierarchical list, the folders will be located on the top of the list and items will be located below. If the property is not set, the location of folders and items will follow the set sorting rules (by code, by name and so on). For example, when creating the new folder with the code that's bigger than all other folders and items (when sorting by code), if the first case this sub-folder will be lower in the list but higher than other catalog items; in the second case it will be located in the lowest row.
It's worth mentioning that setting and unsetting the Place folders on top property doesn't affect the catalog view in non-hierarchical list.
Number of hierarchy levels - the property is available if the Limit of hierarchy levels property is set. The catalogs in 1C:Enterprise can have more than one nesting level. If the Limit of hierarchy levels property is not set, the maximum number of nesting levels is not limited.
Owners - this property required a detailed description.
Any catalog can be used by itself or it can be subordinate to some other catalog or catalogs. For example, the contracts catalog can be used separately of it can be connected to the organizations catalog.
To subordinate the catalog to another one that already exists in 1C:Enterprise, press the editing button in the Owners list field and check the catalogs that are owners of the current catalog in the object choice window opened. Such catalogs are called owners in 1C:Enterprise and the catalog itself - subordinate.
Unlike the multi-level catalog, where all items have the same structure, using the subordinate catalog mechanism allows to connect the items with different structure. In this case each item of the subordinate catalog will be logically connected to one of the owner catalog items.
For catalogs with several owners, different items can have the owners of different types, but one item can only have one owner.
Using subordination - allows to manage limits applied to the owner. Possible use modes: only items, only folders, both items and folders. If a catalog has several owners, the limit is applied to all owners.

Code length - the property sets the maximum length of the catalog item code.
The Designer allows to set the code length to 0. This will be handy in case the catalog item code is not used.
When assigning this property, it's recommended to define the real code length. However, remember that when operating the configuration the code length can be extended if required.

NOTE. The maximum code length is 50.

Name length - the maximum length of the catalog item name is set in the property.
The Designer allows to set the name length to 0. This means the catalog will not have a name.

NOTE. The maximum name length is 150.

Code series - the property allows setting the code uniqueness checking range and automated code assignment.
If the In the entire catalog setting is chosen, when the code is assigned automatically or when the user enters it manually, the code uniqueness will be checked among all catalog items.
Setting Within subordination area setting is only true for hierarchical and subordinate catalogs. In this case the code uniqueness will be checked by 1C:Enterprise within the folder and the item of owner catalog, to which the new catalog item is entered or where the existing item is edited.
When setting the Within subordination area, it's usual that the catalog items in different groups has the same codes. This must be considered if you need to transfer items of multilevel catalog between folders. When the code of the transferred item corresponds to the code of the existing item the warning will be displayed and the item will not be transferred.
When setting a code series to Within subordination area, the owner in the hierarchical subordinate catalogs is provided with the automated numbering and code uniqueness control among the items with the same owner but different parents.
Code type - the property allows choosing the value type for the catalog item code: `Number` or `Row`. Choosing the code string type can be handy when the complex coding system is used and when the code includes letters and separators apart from digits. The most representative example- using markings of clothes as codes.
Please note that choosing a code string type doesn't exclude automated assignment of such codes.
For the first item 1C:Enterprise generates code that looks like 001 (a number of zeros depends on the maximum code length), meaning that the code is the characters line, but all characters are digits. When entering other catalog items,

1C:Enterprise will keep assigning the codes in a similar way: - 002, 003, and so on.
If the catalog keeping requirements include using digits and letters in codes, then for automated numbering the codes can be used that look like: AA001. The first part of code - AA characters - are text prefix, and the second part - 001 characters - will be interpreted by 1C:Enterprise as a digit. It will then be used when assigning the next code automatically.
For example, if the first code in the catalog is AA001, the next automatically assigned code will be AA002, then - AA003, and so on with increments.
The text prefix can be set manually (enter such "compound" code when entering new item in the 1C:Enterprise 8) or you can use prefix setting features if 1C:Enterprise language (`SetNewCode()` method).
Code allowed length - also available when the Code type property is set to Row. The property can be used to control the length of the row that will store the code. If the property value is Fixed, the length of the row with the catalog item code will always be the same as the value of Code length property. Otherwise, the row length equals the number of characters in the item code.
Attributes. Any new catalog can be presented as a table with two columns: the catalog item code and its name. The 1C:Enterprise allows storing additional information about the catalog item, apart from code and name. When editing the catalogs, you can describe the set of additional attributes designed to store such additional data.
When displaying the catalog on the screen, the attributes can be presented as a table columns of the catalog list form. Moreover, the data stored in attributes can be used when generating in calculations, when generating reports and so on.
The catalog attributes mechanism can be used to easily organize employee files, for example. To do this, create attributes to store data on education, ID and other HR information in the `Employees` catalog. The required employee information can easily be found with the native search tools in the catalog when using the configuration.
Tabular sections. To describe some data related to the catalog and not used by themselves, the tabular sections are used. The example is tabular section is the employee family members list (data on each family member are described in the tabular section attributes and the number of members can be arbitrary), work history and so on.
If the data on the employee family members can be used separately, such data could be included into the separate catalog, which would be subordinate to the `Employees` catalog.
The main difference between the tabular section and the subordinate catalog - the catalog items can be referred to and tabular section rows can't be referred to. When accessing the catalog item, the whole catalog with all tabular sections is read and goes from database to the memory. If the tabular section contains a large

number of rows, this might impair the performance. This is why the tabular section is worth using if you don't need to store links to the items and the number of items is limited. Each catalog can have the unlimited number of tabular sections.
Autonumbering. If you select this check box, the 1C:Enterprise will assign the codes to the new catalog items when writing. Users can edit automatically assigned codes.
Uniqueness check. If the code is used for the unambiguous identification of the specific catalog item, it must be unique (it must not be repeated). If the Uniqueness check property is set, the code uniqueness check will be performed automatically when entering new catalog item.
Default presentation - the property sets the catalog items presentation. For example, the values of `CatalogRef` type, entered into the attribute of the document/catalog or constant, will be presented as a code or catalog item name, depending on this property. For the list forms, this column becomes the default one. When opening the list this column becomes active.
Generation. On the Generation tab you specify, which configuration objects could be used as a basis for objects of this type and which objects could use the objects of this type as a basis. When clicking Generation settings wizard you start the generation procedure wizard..
The example of generation is inputting Release for sale document based on the Goods catalog.

5.8.3. Catalog attributes properties

The catalog has the following properties apart from the main ones:

- Type - describes the attribute data type.
- Usage - defines the usage of attribute for folders and items in the hierarchical catalog.

5.8.4. Predefined catalog items

The configuration developer can create a set of predefined items and folders for the hierarchical catalogs.

IMPORTANT! The predefined items can't be created for the catalog with an owner. And vice versa, the catalog that has the predefined items can't be assigned with the owner.

5.9. Documents

5.9.1. General information

Document - one of the main concepts in 1C:Enterprise. Documents are used to organize input of data on the business operations.
In most cases documents created during the task configuration setting are digital equivalents of the standard paper documents; however, this type of data has much more possible usage scenarios than the business standard data registration.
Each document contains information on the specific business operation and is characterized by number, date and time. Date and time - the most important characteristics of the document, because they allows to define the specific sequence of actions with timing.
Configuration only includes description of document structure and the specific document instances are entered by the user during operation. For example, in 1C:Enterprise, the Invoice document created during configuration will allow creating invoices with different content but the same set if attributes, behavior logic and so on.
To simplify, further in the document, the "document structure" will be replaced with "document" to describe means of document input and visualization.
The Designer allows to describe the document structure, organize forms for data input and describe the algorithm for building the print forms.
Apart from date, time and number, you can create a set of attributes for storing additional information.
If the object of subject domain, corresponding to the document, has simple properties (for example date, number and total amount) and complex (list) properties, the set of tabular sections can be created for the document. For example, in the Expenditure invoice document you can create a tabular section for the list of products or services sold.
To work with the documents in configuration you can create the required number of document lists of the same type and document journals of different types. The list form is different from the journal form. First of all, the list doesn't include the Document type column (because the list consists of document with the same type),and the journal usually has such column.
When creating the document you can specify the list of journals, in which the documents of this type will be operated. For the documents of different types you can specify one journal, which allows to customize document groups in journals. The journals assigned to the document can be changed.
The documents can change register state (posting). If the document is posted, the data entered by the user, are displayed in the system registers - goods balance has changed, debt to the contractor has changed and so on. The documents can be posted in real time (real-time posting) or with the past date (non-real-time posting).

5.9.2. Document properties

The Documents branch of the configuration tree is designed for operating documents. The configuration service objects - numerators and sequences - are located on the same branch.
This section includes description of specific document properties that are different from the general configuration object properties.
The document properties are edited in the property palette or editing window. The document data in the editor window are divided into tabs.
The Main tab includes the main document data.
The Data tab displays attributes and tabular sections.
The property palette includes main attribute properties and type of each attribute. For fast search or filtering by attributes, the Index property must be set for the attributes.
The Numbering tab includes data used to assign document numbering rules.
Each document has the required attributes, which are created automatically and which can't be deleted, - time, date, and document number. Unlike date and time, there are several parameters that can be set for the number. These parameters will control behavior of the attribute when operating the document of the created type. The combination if these parameters will define the document numbering rules in 1C:Enterprise.
Autonumbering - if you select this check box, 1C:Enterprise automatically assigns new document number. Users can edit automatically assigned numbers.
Numerator - a numerator can be assigned to the document. To assign the document with the numerator, select in in the Numerator property. In this case other elements for describing numbering rules, except for Autonumbering property, become unavailable, i.e. the rules for numbering documents of this type will be completely defined by the assigned numerator.
Using numerators allows to organize the consecutive numbering for different document types. To enable this, assign the same numerator to all documents that require the consecutive numbering During uniqueness control and assignment of the next number, all documents with this numerator assigned will be considered.
Number length - sets the maximum length of the product name.
Number type - the property is similar to Code type property of the catalog.
Number allowed length - also available when the Number type property is set to Row. This property can be used to regulate the length of the row that will store the number. If the value is Fixed, the length of row with the document number will always equal the value in Number length property. Otherwise, the row length equals the number of characters in the document number.
Uniqueness check - is the device is installed, then when the new property is set, is number is checked for uniqueness in the range specified in Periodicity.
Periodicity - the property sets the limits for document number uniqueness control and period of number repeatability. If the Uniqueness control property for the

document numbers is set, the range of control must be specified in the Periodicity property.
When the Autonumbering property is set the 1C:Enterprise will assign the next number to each new document. After the period set in Periodicity property is expired, the documents numbering will start from 1.
The Register records tab includes behavior settings when posting the document and unposting.
Posting - the value defines whether the document posting is available when writing. If you choose Allow, the document will make register records (change their state). Another consequence of this choice is automated calling of `Posting` event handler for document posting in 1C:Enterprise mode that is enabled when clicking the button of the document form, for which the Save and close predefined action is set during configuration (usually the button is OK). To create the `Posting` event handler you an use register records wizard. It's call button is located on the Register records tab of the Document object editing window.
Real-time posting - the property defines, whether the real-time posting is allowed for the document. For documents with the allowed real-time behavior, when you choose the date other than the current, the document is posted in non-real-time mode, because the fact is already happened and no control is required unlike in real-time mode (for example, verifying the balance, specified in the expenditure invoice of the good).
If the Real-time posting is allowed for the document, the document is created with "null" time. When posting, 1C:Enterprise gets the current time stamp, which can correspond to current date and time, but it can also be more than the current time. The document is assigned with the time stamp and then the real-time posting is performed.
In case the document with the set Real-time posting property is edited, the time is different from the current, meaning that when posting (current date is specified) the request for posting type is displayed. If you choose Real-time posting, the document will be assigned with the current time. If you change the time when editing (for example, specify the future time of the current day), 1C:Enterprise will also set the current time. If you choose Non-real-time posting as a posting type, then when the date format doesn't offer time input, the time of day beginning is specified in the document.
Register records deletion - the property is available if the Posting property is set to Allow. In case of repeated posting or unposting it defines the deletion mode for all records that the document wrote during posting:

- The Auto delete mode means that the deletion is performed by 1C:Enterprise while posting the already posted document (before writing new register records) and while unposting.
- The Auto delete off mode means that the register records are deleted programmatically under special conditions. This mode is used when you need to control deletion while posting or unposting.

- The Auto delete on unpost mode means that 1C:Enterprise will automatically delete register records only when unposting. The register records will not be deleted during the repeated posting. This mode is set by default.

If the Register records deletion document of the document is set to Auto delete off or Auto delete on unpost, then to avoid data duplication, clear the `Registry records` collection when proceeding to post.

Register records writing on post - the property defines system behavior when creating register records during document posting:

- The Write selected mode (the default mode) means that the register records collection will only be written if the `Write` set property is set to `True` (regardless of the modifications in records).
- The Write modified mode (set during conversion from 1C:Enterprise 8 and earlier versions) means that only modified record sets will be written (the 1C:Enterprise will automatically set the `Write` property to `True`).

After the document is written, the 1C:Enterprise set the `Write` property of all record sets registering document register records to the default one, even if the writing was unsuccessful.

Sequence filling - the mode of automated sequence filling is chosen. In the editing window on the Sequences tab, the occurrence of the document in a sequence.

In the document editing window of the Journals tab, you can mark document journals, in which the documents of this type will be displayed when user operates 1C:Enterprise. The required document journal form can be created later.

The Generation tab includes two lists of configuration objects. In the top list you must specify the objects used as a basis for document generation, and in the lowest - which objects will use this document as a basis when generating.

IMPORTANT! The document can be generated based on other document or the object of different type (catalog items, chart of characteristic types, chart of accounts, and chart of calculation types). And vice versa, the document can be used as a basis for generation of other document or object of different type.

To create the procedure that prepares source data for object generation, use special wizard.

On the Rights tab you can set the privilege mode of posting (Post in privileged mode property) and/or unposting (Unpost in privileged mode property):

- If the property is set, the corresponding action (posting or unposting) will always be performed by 1C:Enterprise in the privileged mode (when calling posting or unposting on the server side and in file mode). However, the privileged mode will not be set if the posting (or unposting) of the document is performed in client/server mode on the thick client side. The following is performed in privileged mode:
  - Automated register records deletion.
  - Automated register records writing.

- The handler (`Posting` или `UndoPosting`). However, the object writing is performed in the ordinary (non-privileged) mode.

- The privileged mode is enabled by the system after the object is written before posting (before deleting register records in case they are deleted automatically). The same applies to unposting.
- When creating new documents, the properties are set to `True`, if the default run mode - managed application - is set in the configuration properties, and `False` if the startup mode is ordinary.

5.9.3. Document posting mechanism

The information on the business operations is stored in registers. Documents can change register state. Such process is called **posting**. This mechanism is a recommended way of changing the register state. The posting is performed in **real-time** and **non-real-time** mode (Real-time posting property).

NOTE. As a rule, the real-time posting mechanism is used for current accounting.

5.9.3.1. Real-time and non-real-time posting

The real-time posting mechanism is designed to separate the cases when the document is posted in real time and the cases, when the document posting register something that already happened.

The real-time posting is required when the input and posting don't just register the event but take part in its creation and help the operator to input the information correctly. Of course, it only makes sense when the event is happening in real time.

The classic example is inputting and posting the document describing the sale of goods from the warehouse. When inputting such document, the operator's task is not to simply enter the list of goods sold to the customer correctly but to perform various checks. First of all, the operator must check, whether the product is in stock in the specified warehouse. It's important to remember that several operators work at the same time and they might write out the same goods. The 1C:Enterprise task is to avoid selling the same thing to different customers. Besides, checking the loan available for the customer and verifying payment might be required as well as many other types of checks.

At the same time, in case the document is input with the past date, i.e. it's known that the business event is already happened, for example, the specific product is already shipped to the customer, such checks are not required and you only need to register the event. In this case posting the document registers the event and doesn't take part in its creation.

Thus, the task of the real-time posting mechanism is to separate two posting modes: from the user's perspective, so he or she could understand, which type of posting is performed; and from the algorithm's perspective, so it could to perform actions in accordance with the current posting mode.
The document posting mode is identified by the date. If the posted document date correspond to the current date, the 1C:Enterprise will post such document in real-time mode without additional requests and you can learn it from the behavior processing to build the specific algorithm of document posting.

5.9.3.2. Form extension and posting

Apart from the Real-time posting document property, there's a way to set the posting mode for the document form extension. The Use posting mode property can have the following values:

- Non-real-time - the document will be posted in non-real-time mode. If there are not enough rights for non-real-time posting, the exception will be displayed.
- Real-time - the document will be posted in real-time mode. If there are not enough rights for non-real-time posting, the exception will be displayed upon the attempt to post a document with the past date.
- Auto - in this case, the 1C:Enterprise follow the algorithm:
  - if the document date is earlier than the current date, the non-real-time posting is performed;
  - if the document date is the same as the current date, the real-time posting is performed;
  - if the document date is later than the current date, the exception is displayed;
  - if the document can't be posted in the chosen mode (not enought rights and so on), the exception will be displayed, too;
  - if the posting mode is unknown on the client side, the `Undefined` value will be set to `PostingMode` parameter of the `BeforeWrite` event handler.

One of the objective if document posting is placing them chronologically on the time scale. Such placement is required, in particular, to write the balances off correctly in the balances register of the current accounting. To place the documents, we use such terms as **point of time** and **current timestamp**. Let's have a closer look at these terms.
To identify the document location of the time axis, the `Date` document attribute is used, which contains accurate time to seconds. This allows to control the document writing sequence. However, when large volumes of documents are created, it might happen that several documents have the same date value (i.e. they were created within a second). How to define the sequence of created documents in this case?

5.9.3.3. Point of time

To handle such situations, there is such term as **point of time**. The point of time is a combination of date, time and link to database object. It allows to unambiguously identify any reference object of the database on the event axis but it only makes sense for documents. Besides, the point of time allows to identify non-object data, for example register records subordinate to the register.

The concept of point in time is implemented in 1C:Enterprise language using the `PointInTime` universal object.

The sequence of several documents, which have the same date and time, on the events axis is defined by the system based on the links to these documents. It can be similar to documents creation sequence and it's not available for editing by the user, i.e. you can't control the document sequence within the second or calculate that one document was created earlier and another one - later.

5.9.3.4. Real-time timestamp

Current timestamp - is the `Date`-type value. The current timestamp - the "basis", which enables the real-time posting. The current time-frame is created by the system each time the document is posted. Its value is based on the current session date and the last created current timestamp.

5.9.3.5. Zone time

When operating the system from different time zones, consider this fact when getting the current timestamp. For example, the single database located in one city (time zone) might be used for accounting at several enterprises (for example, branches of a holding), which are located in other cities (and other time zones). In this case each branch needs the separate timestamp.

To account in new time zones, there are concepts of infobase time zone and session time zone.

Infobase time zone defines time zone set for new session by default. When the infobase is created the time zone is not defined. However, it can be set using the `SetInfobaseTimeZone()` global context method. The data on the infobase time zone is saved in database and doesn't change with the infobase dump and restoration operations. When creating the initial infobase image the infobase time zone, in which the image is created, is transferred to the image created.

If the infobase time zone is not set, the time zone of a computer with 1C:Enterprise server is used (in client/server mode) or the local computer time zone is used (in file mode).

Session time zone describes the time zone, in which the specific session runs. By default, the session time zone is the same as infobase time zone.

The session time zone can be set using the `SetSessionTimeZone()` global context method. The session time zone is saved until the end of the session. It is used to determine the current session date and the **current timestamp**.

5.9.3.6. Getting current timestamp

During real-time posting, the 1C:Enterprise changes the document time to ensure that the next document posted in real0time mode has the later point of time than the previous document posted in real time. The **current timestamp** concept is used for this. The 1C:Enterprise gets the current timestamp automatically during real-time posting, but it can also be received explicitly in 1C:Enterprise language using `GetRealTimeTimestamp()` method based on the current session date. **Current session date** is equal to the computer date modified to comply with the session time zone. Modification means converting the local computer time to the zone time set by the session time zone. The time is converted through universal coordinated time (UTC).

All users access the same current timestamp mechanism, and the mechanism issues such timestamp. The current timestamp mechanism provides the date that is later than the previous timestamp received by this or another user in this time zone. As a rule, 1C:Enterprise returns the current session time as a timestamp. However, in case the current time is more or equal do the latest issued timestamp, that mechanism returns value that is one second larger than the latest issued timestamp. Thus, upon access the user receives the value that might correspond to the current time but is always more than the previous issued value.

Remember that different sessions within the same time zone will use the same time to get the current timestamp. There will be as much separate current timestamps as the unique time zones set as the session time zones.

5.9.4. Numerators

5.9.4.1. General information

Numerator is a configuration object that describes document numbering rules: document number type and length, it's periodicity, the need for uniqueness control.

The main purpose of a numerator - provide consecutive numbering of documents with different types. The same numerator is set for the documents to enable this.

5.9.4.2. Managing numerators list

To operate configuration objects of Numerator type, there is a configuration three branch, which is located within the Documents branch and starts with the Numerators keyword.

5.9.4.3. Numerator properties

This section includes description of specific numerator properties that are different from the general configuration object properties.

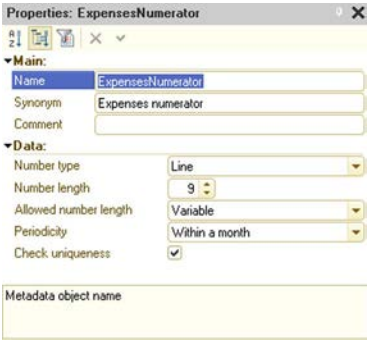


Fig. 114. Numerator properties

Number type - that value type for a document number - digits or text. Choosing the text type can be handy when the complex document numbering is used and when the number may include letters and separators apart from digits.
Number length - sets the maximum length of the product name.
Periodicity - the property sets two important numerator characteristics: the limits for document number uniqueness control and period of number repeatability.
If the Uniqueness control property for the document numbers is set, the Periodicity property sets the limits for such control.
For example, if the periodicity is set too Within a day, the uniqueness of document numbers will be controlled within one day: that document numbers can repeat on the other day but within one day they will be unique.
When the Autonumbering property is set the 1C:Enterprise will assign the next number to each new document. After the period set in Periodicity property is expired, the documents numbering will start from 1.
Uniqueness check - is the device is installed, then when the new property is set, is number is checked for uniqueness in the range specified in Periodicity.

5.9.5. Document sequences

5.9.5.1. General information

The document sequences are auxiliary configuration objects. They enable posting of specific documents with a strict chronological order.
All documents in the 1C:Enterprise form a single chronological sequence. To do this, each document has a date and time. Even if two documents have the same date and the same time, they are still arranged in a sequence, determined by the time of their entry into the system. Date and time of the document are subject to

change. Thus, regardless of the order of entry, documents can be arranged in a sequence that reflects the actual order of events that occurred in the economic life of the company and that are described in the documents.
During posting, a 1C:Enterprise document performs actions that are registered, by this document, in several accounting systems supported by 1C:Enterprise.
The algorithm of document posting, as a rule, reflects data recorded in the document (its attributes and tabular sections). However, in some cases, the algorithm for carrying out the document also analyzes the current results, using them during the implementation. For example, if a document writes off goods or materials at average cost, then to determine the amount of write-off, the carrying out algorithm will analyze the remnants of goods (materials) at the time of the document. If the write-off is performed using LIFO or FIFO methods, the posting algorithm will analyze the existing balances of goods (materials) broken down by lots at the document time, defined by the document posting date and time.
Obviously, documents that use these totals should be posted in strict sequential order. However, in practice, due to errors in the input of information and the late receipt of documents, it is often necessary to introduce or correct documents in hindsight. Back-entering or back-correcting a document invalidates all register records generated by the documents following this one. For example, if it became clear that the quantity of goods was incorrectly indicated in one of the incoming invoices at the beginning of the month, then in all subsequent expenditure invoices writing off existing lots, it is necessary to re-analyze the balances considering the changes made and re-record movements of the registers. Therefore, all documents that analyze the balances following the corrected document must be reposted.
In case the infobase, in which the sequences are used, is included into distributed infobase, it's recommended to include document sequences, which include documents sequences with automated sequence filling enabled, into exchange plans. Otherwise, there might be an unlimited data exchange and document reposting.
For automatic control of document reposting, objects from Sequences branch are used. Each Sequence object entered into configuration provides control over the document posting of the specified types. Thus, there may be several independent sequences in 1C:Enterprise.

5.9.5.2. Managing sequences list

The Sequence-type objects are created in Configuration window. There is a separate configuration tree branch for sequences, which is located "inside" the Documents branch and starts with Sequences keyword.

5.9.5.3. Document sequence properties

This section includes description of specific document sequnce properties that are different from the general configuration object properties.
The sequences properties are edited in Sequence editing window.

On the Usage tab, you can select documents, related to this sequence, and register rcords affecting the sequence.

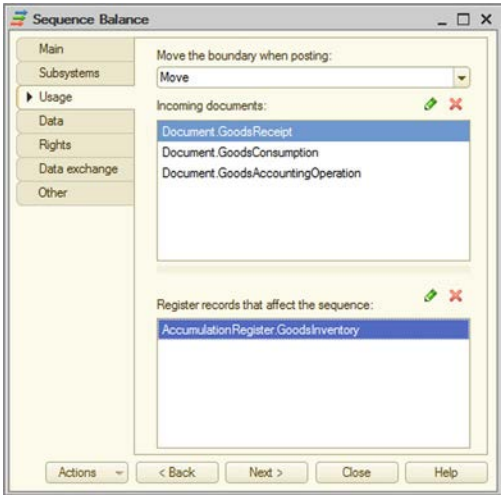


Fig. 115. Sequence properties

Move boundary on posting. If the Move value is set for the property, the document registered in this sequence will try to move the boundary of this document sequence when this document is posted. In case the Don't move value is set for the property, the document will not try to move the boundary of this document sequence when this document is posted.

Incoming documents - in the upper list of the Sequence window the types of documents related to this sequence are specified.

Choose the types of documents that will analyze the register state when posted as the ones that will be affected by the sequence. Such documents may include invoices, transfer note, release note, sales note and so on.

Register records affecting sequence - one of the main sequence properties. It defines, which register records will affect the need to repost documents in this sequence, i.e. which register records and totals of which accounting mechanisms are used by the documents in this sequence when posting. For example, registers can be such register records.

To customize the parameter, add register types, which will brake this sequence by their register records, to the list.

Dimensions. Sequences can have the subordinate objects called dimensions. They are created on the Data tab of the editing window.

In case there are no dimensions created for the sequence, all incoming documents will be reposted when this sequence is restored. If you need the sequence to account the specific situation, not all of them, the dimension should be included

into the sequence. In this case, only those documents that change the register state with regard to the dimension properties will require reposting.
If the state of registers in the sequence changes, the later documents with the same values in attributes (listed in the Conformity with document attributes dimension property) that are included into the attributes of deleted (added) register records (listed in Conformity with register record attributes dimension property) become irrelevant.
For example the sequence account changes in the register state for Arrival invoice and Expenditure invoice documents. In case there are additional criteria of the need to repost the specified documents must be considered (for example, the documents should be reposted by the specific product or service value), the dimension must be added into the sequence. Its type must be specified in property palette of the dimension (`CatalogRef.Products`) and connection with the register attributes must be set.

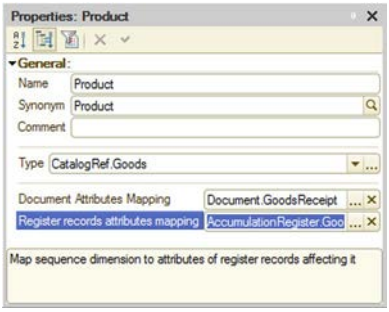


Fig. 116. Sequence dimension properties

According to the chosen dimension type, the dimension document and register lists to choose from only include those object that comprise dimensions of the specified type.
Entering the dimension allows to reduce time of document reposting, which is particularly important when there is a large number of documents because online those documents that contain data of the specified dimension value will be reposted.

5.9.5.4. Operating document sequences

When operating in the 1C:Enterprise mode, the 1C:Enterprise will automatically maintain the boundary of each document sequence entered into configuration. This sequence boundary is the point of time, which is the date from the document itself. When posting the documents from the sequence sequentially, the sequence boundary will be set at the point of time of each document posted. However, when the document included into the sequence but created later than that already posted document from the same sequence is posted later than the sequence

boundary, the sequence boundary will not be shifted, because the document posting sequence will be corrupted. The document postng algorithm can analyze this situation.
When posting the documents with their past date, unposting or deleting documents, when register records are deleted specified as affecting this sequence are deleted or written, the sequence boundary is moved to the point of time of the changed document. Before moving the boundary backwards, the 1C:Enterprise verifies that there are borders to move. Such verification is performed without exclusive boundary locking.
Thus, the sequence boundary will move forward with the sequential posting of the documents related to this sequence and will move backwards when changing register records, related to this sequence, with the past date.
Using point of time in sequence boundary results in the following: deletion of the document that creates point of time of the document sequence doesn't lead to clearing the link to this document from the point of time. This results in link search mechanism (`FindByRef()` and `DeleteObjects()` methods) finding link to already deleted documents.
In the document reposting mode (called by choosing All functions - Standard - Document posting) there is a special feature for restoring a document posting sequence. It allows you to automatically repost all documents related to the current sequence, from the sequence boundary to the specified moment.
In the example with goods accounting, the posted expenditure invoices will move the sequence boundary forward. Any change in register records from goods accounting register written earlier than the sequence boundary will move the sequence boundary backwards to the point of time of this document. After this, the documents poster later than the sequence boundary will not move it forward if the already posted documents from this sequence will be in between the sequence boundary and the posted document. The sequence restoration mode will repost all expenditure invoices. Please note that although the register records of the receipts affect the sequence boundary. they will not be reposted because they don't use balcances in the postig mechanism and are not included into the list of documents related to this sequence. After the sequence restoration the documents posted after the sequence boundary will move the boundary forward again.
The sequence recovery mode allows you to automatically repost all documents related to the sequence, from the current position of the sequence border to the specified moment. Choose the last position to be reposted in the upper part of the dialog box.

5.9.6. Document generation

One of the ways to enter new documents in 1C:Enterprise is generation. From the user perspective, the generation mode allows to enter documents and catalog items, filling its attributes by copying information from another infobase objects - document or other object type.

On the Generation tab you can choose objects, which can be used as a basis for generation of the specified object type (Generation field) as well as objects that can be generated based on this document type (Can be used as a basis for generating field).
To implement the generation mechanism you must implement the `Filling` event handler in the document module.
The handler text can be customized by the specialist who configure the 1C:Enterprise. The handler text must include execution of some operations of information transfer, depending on the type of the source object, and other required actions.
Generation settings wizard is designed to simplify the creation of such handler.

5.10. Document journals

In 1C:Enterprise document journals enable operation with docyments of different types. When operating the journal forms, the user can enter, view. edit, and delete documents.
Users can search documents in the journal by column content and document number of filter documents.
The wizard allows creating any number of journals.
When creating a journal you can create an arbitrary number of screen forms containing columns for displaying document type, number, date and time as well as other columns for attributes of the documents from the journal.
If there are no journal forms created, the default form is created automatically in 1C:Enterprise mode.

5.10.1. Journal creation

The Document journals branch of the configuration tree is used to operate document journals.
In 1C:Enterprise the processes of journal creation and document posting by specific journals are closely connected. Specifying display of document information in a specific journal is synchronized with the data on documents in the journal.
When creating journal and documents, the created journal or document is automatically added to the relevant list. To display document data in journal, you have to specify whether the document belongs to the journal. You can specify this both in journal and in document because this operation is synchronized.

5.10.2. Journal editing

This section includes description of specific journal properties that are different from the general configuration object properties and methods to edit `Journal`-

type of configuration objects that are different from general methods for editing configuration objects.
To edit journal properties (create the list of additional columns and specify their content, at journal forms and print form templates and so on), use and it's in the window.
On the Data tab, the list of documents in the journal and the list of journal columns are generated.
The document attribute included into the journal is specified in each subordinate object in Columns branch.

5.10.3. Document journal columns

When the new document journal is created in configuration, the unlimited number of journal forms can be created to operate it. The journal form is created using the configuration object form wizard. The wizard places a spreadsheet field with columns, displaying document attributes, in the form. When creating the journal form, the form wizard creates the following columns: image (to display document status), type, date and time. If you need to include additional information into the journal, generate the list of additional columns and place them in the forms.
The list of documents that are used as a source of data for the journal are located on the Data tab of the of the editing window. To create additional column in the lower list, add the subordinate Column object and specify attributes of documents, which will be used as a source of data displayed in the column.
To choose document attributes, which will be located in the journal column, click the selection button in the Links property of the column property palette. This opens the attributes selection dialog box.

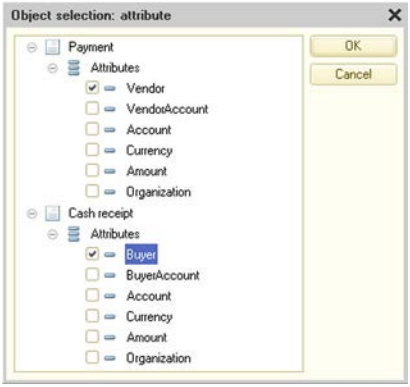


Fig. 117. Choosing attributes for column

IMPORTANT! You can't choose several attributes of the same document.

If the attribute of some document is not chosen, the document journal column will not display information on all documents of this type. When choosing, use your best judgement and don't mix completely different concepts in the same column (for example, contractor name and document total).

Apart from mandatory columns (Date, Number, Document type) and columns specified in their Columns subordinate journal object the group, you can add any number of additional columns in the journal.

In case all documents with the string-type number included into the journal have the fixed number length, the Number journal column will also have a fixed length. If at least one document with the string-type number included into the journal have the variable number length- the Number journal column is created with the variable length.

The display of journal columns is affected by the user access rights. In case the user doesn't have the View right for attribute of at least one document included into the journal column, such column will not be available for the user.

The new column is added into the column list of the chosen Document journals object. The additional graphs let the user to get the most important information on the documents when viewing the journal without opening the document itself. The following properties of attributes included into the column are automatically applied to the element of the form displaying the journal column:

- Password mode - if search mode is set for at least one attribute included into the column.
- Format - in case of the full match of attributes included into the column in all languages specified in configuration.
- Hint - if the hints match for all attributes, one hint is displayed; if the hints don't match, they are displayed separated with commas.
- Mark negatives - if this property is set for all attributes included into the column.
- Multiline mode - if this property is set for all attributes included into the column.

5.10.4. Interactive creation of new document

If the document journal includes more than one document type, the journal list form offers two options for creating the new document:

- The Create button is located on the command bar and after clicking it (or clicking Ins button) the dialog box with the list of document types displayed in journal is opened.

This option is only used automatically when the Combatibility mode configuration property is set to Version 8.3.3 and lower.

- The common bar includes the Create button with a menu containing the list of all document types displayed in the journal. The same menu is generated after clicking Ins. The menu is generated for the current table cell. In case the journal only displays one document type, the menu is not generated.
  Document types disabled by the functional options and for which of the current user doesn't have the `InteractiveInsert` right are not automatically included into the generated menu when creating the new document.
  This option is only used automatically when Compatibility mode configuration property is set to Version 8.3.4 or older or Not used. This behavior is similar to command bar of the dynamic list with automated filling.

In case of automated document type list generation, the types are sorted by incoming document presentation.

If the command bar of the dynamic list or form (in case the main form attribute is a dynamic list) doesn't use automated filling, you can generate such menu manually. To do this, follow these steps:

1. Create the button group with the required name (for example, Create) and Submenu type in the command bar.
2. Add the Create by parameter standard dynamic list command into the group. For the button associated with this command, specify Parameter property value, document type, which will be created after clicking this button. The presentation of relevant document type (specified in Parameter value) will be used as the button presentation.
3. Add the same number of command as the planned number of documents created with this menu.

When creating the button with the menu manually, remember: when clicking Ins in dynamic list, the menu will include all document types displayed by the dynamic list, not only those, located in the command bar menu.

If the `BeforeAddRow` table event is processed in the journal form list, you can analyze, which document needs to be created, in this event. The parameter of `Parameter` event handler procedure will contain the type of created document if the document creation is initiated using the button associated with the Create by parameter command (if this parameter is specified). In all other cases this parameter value will be set to `Undefined`.

5.11. Enumerations

Enumeration is the service data type, which is not used separately but with other data types. The enumeration can be defined as the list of possible attribute values.

Enumerations are used when entering attribute values of documents and catalogs or when entering constant values in cases where you need to exclude ambiguous information input.
Let's use the "customer status" concept as an example. In the simplest form, there are retail customers and wholesale customers. The customer status is usually associated with the level of discount from retail price.
Such "statuses" list - retail, wholesale - is an example of simple enumeration. When writing out an invoice, the user has to specify the customer status by choosing it from the list. The specified customer status defines the prices.
If the customer status is entered during configuration of enumeration task, an expert configuring 1C:Enterprise can set the options for retail price calculation depending on the specified status.
First of all, the enumeration could not be updated when operating it: the list of values is set when configuring enumeration in the Designer.
The enumeration lacks nesting - all its values are on the same level.
The main feature of enumeration is that the list of values is known and is available in the Designer - the configuration itself uses the specific enumeration values.
Enumeration allows to limit the number of possible options, for example when entering new document attribute. The list of enumeration values is created in configuration, meaning that you can organize verification of the chosen value and describe actions, which has to follow.
The Enumerations branch of configuration tree is designed to operate enumerations.
To edit an enumeration you have to create the list of enumeration values. The Enumeration object editing window is used to edit enumerations. When creating the new enumeration, the Use standard commands property is disabled.
On the Data tab you can generate enumeration values.

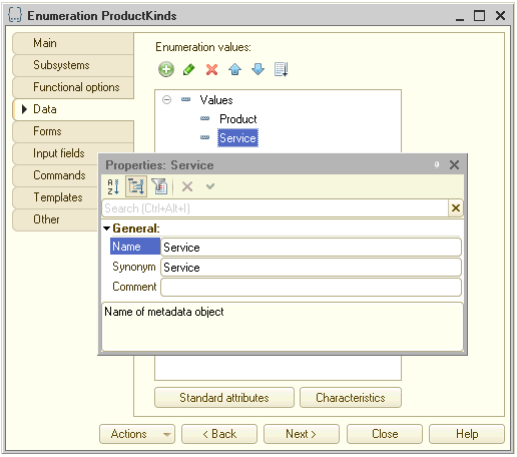


Fig. 118. Adding enumeration value

The name and synonym are specified in the property palette.
In 1C:Enterprise mode each value from enumeration values list is represented by a synonym; and if the synonym is not set, 1C:Enterprise uses its name.
In the example on fig. 118 the `ProductTypes` enumeration value will be represented as Goods (by the entered synonym).
On the Forms tab you can create list and choice forms. This allows to create various forms for choosing (depending on the applied context). The list forms allowed to print enumeration lists. When creating the list form, the Use standard commands property is enabled automatically.
On the Templates tab you can create printing templates.

5.12. Reports and data processors

5.12.1. General information

Any accounting automation system only operates properly when it has the means for data processor the collected information and getting consolidated data in a format that is easy to use and analyze. As a rule, the accounting automation system allows to generate reports to do this. The designer allows to generate sets of various reports, which meet the users need for reliable and detailed data out.
The configuration objects from Reports configuration tree are used to generate reports in 1C:Enterprise. Each object of this type can contain the algorithm for generation of paper or digital report in 1C:Enterprise language or data composition scheme, which can be used by 1C:Enterprise as a basis for automated report generation. The report can contain one or various forms, which can be used to organize input of some parameters that affect the algorithm. To output the results of the algorithm on the screen or printer, the report can have the templates for describing print forms (templates) create in the wizard.
Editing the properties of Report-type and Data processor-type objects and creation of the subordinate objects is performed in the editing window.
In 1C:Enterprise, configuration objects are used to perform various actions with the information. These objects are located in Data processors branch of the configuration tree. For example, they can be used to delete outdated data from the system, import information from other systems, and many other things. The nature of performed actions reflects the name of configuration object - Data processor, because this results in changes of information stored in the system.
The processing can contain one or various forms, which can be used to organize input of some parameters that affect the algorithm. The display and printing of algorithm results are made using the print form description template wizard (templates).
The main difference between the report and data processor is the ability to use data compositions scheme. In other details the data processor is the same as the report.

You can use the same report forms, settings, and report options for several (or all) report forms of the application. To do this, use common forms. In general, there are several use options:

- The same set of forms is used for all reports of the application. In this case you need to create the required forms and specify them in the configuration properties. In this case there is no need to develop forms in reports because the common forms will be used.
- You can specify sets of reports, for which the custom forms are acquired. In this case the common forms for operating reports are created, which are specified for each report in "their" group. This way you can create, for example, a special set of forms for accounting reports or for analytics reports.

NOTE. In case the same common form is set as the main form of several reports, you will only be able to open one such report in thick client (in regular mode).

If the Open new window standard command is used in the report form, remember, that the parameters transferred to the form, from which the standard command was used, are not transferred to the new window.

5.12.2. Externaldata processors and reports

5.12.2.1. General information

External data processor in 1C:Enterprise is the data processor stored outside of configuration in a separate external data processor file. External data processor performs the same tasks as the Report-type or Data processor-type configuration objects.

The external report in 1C:Enterprise is the report stored outside of configuration in a separate external report file. The external report performs the same tasks as the Report-type configuration object.

The main task of the external data processor (report) is to implement, provide, and update features without affecting the configuration.

External data processors and reports can be stores in two formats:

- Binary files with .epf and .erf extensions. These files can be used for editing in the Designer. They can also be used in 1C:Enterprise mode. You can develop and debug while operating the 1C:Enterprise. In this case the development and debugging of the data processor (report) is performed much faster: The external data processor (report) is edited and saved in the Designer mode, without saving the whole configuration, and launched - in 1C:Enterprise mode. To run the external data processor (report), you have to upload it with File - Open. It operates similar to any other application (report).

- XML file with extension that corresponds to files used in configuration export/import into files. When using this format you can only edit external data processor (report) in the Designer mode. To use the external data processor report in 1C:Enterprise, save it with .erf/.epf extension. The 2.0 format is used for its operation.

NOTE. External data processor (report) that is opened using the File - Open menu, are performed in safe mode if the user does not have administrative access rights.

Any configuration object of the Data processor-type or Report-type can be saved into the external data processor (report) file and vice versa - the existing configuration object can be replaced with the external data processor (report). A help information can be created for external data processor (report), just like for any other configuration object.

TIP. To ensure configuration integrity, it's recommended to use the external data processors (reports) for debugging. After debugging of the data processor (report) generation algorithm is over, include the external data processor into the configuration.

Saving the external data processor (report) into XML is required when you plan to store the data processor (report) into the external version control system. In this case the XML format is more convenient than .epf (.erf) extension. The Designer allows opening external data processor (reports) for editing in any format. In this case, by default the writing is performed in the format of data processor (reports) opened in the Designer. To change the storage format, run the File - Save as... command or File - Save copy... and choose the required saving format in the opened dialog box.

5.12.2.2. Creating external data processor (report)

To create the external data processor (report), choose File - New and then, in the displayed request, choose External data processor or External report.

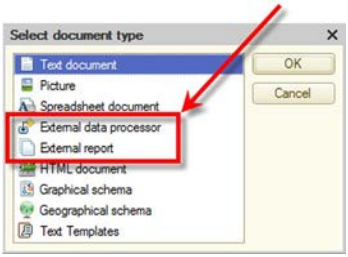


Fig. 119. Document type selection

The form editor for external data processor (report) development will be displayed. The editing window of external report will additionally include controls for creation, setting, and editing the data composition system.
The external data processor (report) is not a part of current configuration (although it is very closely connected with it), and it's saving procedure is different from their procedure for saving configuration changes. To save the external data processor (report) use File - Save or File - Save as... or File - Save copy. In the standard file saving dialog box, choose the required saving option and then choose a name of the saved external data processor (report).

5.12.2.3. Using external data processors (reports)

To use the external data processor (report) when operating 1C:Enterprise, it has to be opened the same way as in the Designer. However, you should remember that in 1C:Enterprise the external data processor can only be opened for execution: the user can't edit it. The attempt to open is followed by the verification of access rights. Then, the safe mode is enabled (if the user doesn't have the administrative rights). Besides, to run the external data processor (report) it must be saved in binary format (.epf, .erf).
The compilation of external data processor (report) module is performed when opening the external data processor (report). That's why when editing the external data processor (report) in the Designer and saving it you have to open this data processor in 1C:Enterprise mode once again.
The external data processors (reports) can also be operated on the 1C:Enterprise server. All limits for interactive objects (forms and so on) usage are the same.
To operate the external data processor (report) from 1C:Enterprise language, you must connect it using the `Connect()` method (only available on the 1C:Enterprise server).

```
// Place the data processor into temporary storage on the client
StorageAddress = "";
Result = PutFile(StorageAddress, "ExtDataProcessing.epf", , False);

...

// On the server, connect the data processor from previously created
// temporary storage.
DataProcessorName =
ExternalDataProcessors.Connect(TemporaryStorageAddress);
```

The name of external data processor will be placed to the `DataProcessorName` variable. This name will be used to call the connected data processor. Example, when opening the data processor form:

```
// Let's open the form of the connected external data processor
```

OpenForm(«ExternalDataProcessor."+ DataProcessorName +".Form»);
External data processor (report) can be connected in the safe mode. To do this, use `SafeMode` parameter of the `Connect()` and `Create()` connection

methods. The external data processor (report) connection mode will depend on the value of this parameter:

- the parameter is set to `True`: external data processor (report) will be connected in the safe mode;
- the parameter is set to `False`: external data processor (report) will be connected in the unsafe mode;
- The security profile name is set as the parameter value:
  - The name is not set: the external data processor (report) is connected in the safe mode;
  - The profile name is specified and the Safe mode security profile parameter is set to `True` for this profile: the external data processor report will be enabled but it will only operate when controlled by the security profile with the specified name;
  - The profile name is specified and the Safe mode security profile parameter is set to `False` for this profile: the external data processor report will be enabled but it will only operate when controlled by the default security profile;
  - There is no security profile with the specified name on the 1C:Enterprise server: the external data processor (report) is connected in the safe mode.

The external data processor (report) might contain malicious code. That is why, in case of an attempt to open the external data processor (report) interactively, they permission request will be displayed to the user.

If you answer Yes in this dialog box, this choice will be remembered and further openings of this data processor (report) will not display any questions. If you answer No, the data processor (report) will not be opened. This question will be displayed again with the next opening.

For programmatical operation the external data processor (report) can be located:

- In a file located in configuration (for example, in a template);
- In infobase data;
- In the temporary storage.

The names of external data processors (reports) must be unique within the session. In case of problematical or interactive connection of the external data processor (report) with the same name as the external data processor (report) that was already loaded during this session, the old data processor will be disconnected and the new one will be connected.

NOTE. When operating the external data processors reports in thick client, remember that when operating in the managed application mode, you can only open managed forms, and in the normal mode - only regular forms.

When operating the external data processors (reports), remember:

- If you connect the new data processor (report) with the same name as the already connected data processor (report), the opened form from the old data processor (report) are disabled (the error is displayed).
- When connecting the external data processor (report), in case the external data processor (report) is binary identical to the already connected one and the safe mode flag is the same, there is no actual reconnection and the error is not displayed. Otherwise, the previously connected external data processor (report) is disabled and the new one is connected.
- When executing the `Create()` method, the `SafeMode` method parameter is ignored, if the external data processor (report) was previously connected with the `Connect()` method. If the connection is the first one, the external data processor (report) will be connected using the `SafeMode` parameter from the `Create()` method.
- When the external data processor (report) is connected programmatically with the `Connect()` method you can specify, whether the control of unsafe operation is required. For this use the `UnsafeActionProtection` method parameter.
- When getting the form for external data processor (report) the open form will be found. It doesn't matter, whether it was open for the currently connected data processor or for the previously connected one (with the same name).
- When opening the data processor (report) with Service and Settings - File -Open menu command, the data processor (report) form is opened with the `OpenForm()` method with `Uniqueness` parameter set to `True` to enable opening of the new data processor form in case of changes.
- If not specified explicitly, all of the above features are equally applied to both interactively and programmatically opened external data processor (report).

5.12.2.4. Editing external data processor (report)

The external data processor (report) is edited in the Designer. To open the existing external data processor (report), choose File - Open. In the displayed standard dialog box, choose the file type corresponding to the format of the saved external data processor (report) and specify the name of the file opened.
When opening the external data processor (report) in the Designer the object editing window is opened automatically. Unlike other configuration objects, the debugging of external data processor (report) can be performed without restarting the 1C:Enterprise. You only need to call the data processor report again in 1C:Enterprise mode after saving it by the Designer (only for the data processor (report) saved in .epf/.erf).
You can copy the external data processor (report) in the editing dialog box without changing the object edited. To do this, run Actions - Export to files commands. Apart from copying the external data processor (report) you can choose

the export format. By default, during this operation the 1C:Enterprise offers the export format that is different from the current format. It means that if the edited external data processor (report) has the .epf/.erf extension, the XML format will be offered and vice versa. You can replace the edited external data processor (report) with another external data processor (report), which is located in the file (in any of the supported formats).
When exporting/importing the external data processor (report) to/from the file, you can change the object type without changing the object edited. For example, you can export the edited external data processor as the external report and vice versa, and the object edited will keep the type.

5.12.2.5. Help information

The external data processor (report) can be complemented with the user description. To edit the description, click Open link of the Help information property in the external data processor property palette.
To view the description of external data processor (report) in 1C:Enterprise mode, click F1.

5.12.2.6. External data processors (reports) and configuration objects

5.12.2.6.1. General information

The Report-type and Data processor-type object in the configuration can be modified into external reports and data processors and vice versa: external reports and data processors can replace the existing configuration object of Report-type or Data processor-type. External reports and data processors can also be added into configuration structure as new configuration objects of Report-type and Data processor-type.
When operating the external data processor (report) you can choose the format of the object saved. It can be done in the saving dialog box or by opening the file with the drop-down File type list. In defines, whether the data processor (report) will be saved as .epf/.erf or in XML format. In this section, the choice of external data processor (report) format will not be described in detail.

5.12.2.6.2. Copying the data processor (report) into the external data processor (report)

The existing configuration object of the Report-type or Data processor-type can be copied into an external data processor or a report. To do this, select the configuration object in Configuration window and choose Save as external data processor/report in the context menu of the configuration object.

This will result in creating an external data processor report, which will be a copy of the selected configuration object. The configuration object itself will not change.
This operation only makes sense for further debugging the the data processor (report) created. After the debugging, the external data processor (report) can be entered in a configuration instead of the existing configuration object. Remember that to execute the data processor in 1C:Enterprise mode, you must save it as .epf/.erf and not in the XML format.

5.12.2.6.3. Replacing the data processor (report) with the external data processor (report)

External data processors (reports) can replace the configuration objects of Report-type and Data processor-type. To replace the configuration object with the external data processor (report), select its name in the Configuration window and use Replace with the external data processor/report item in the context menu of the configuration object.

5.12.2.6.4. Adding the external data processor (report) into the configuration structure

The existing external data processor (report) can be added into the configuration structure as the new Data processor-type or Report-type configuration object. The do this select any name of Data processor-type or Report-type configuration object into consideration structure and use the Insert external data processor, report context menu item of the configuration object. As a result the new data processor (report) will appear in the configuration tree.

5.12.2.7. Comparing and merging external data processors (reports)

External data processors (reports) can be compared and merged with data processors (reports) from configuration as well as other external data processor (reports).
To compare the report or data processor, open the Configuration, specify the acquired object and choose Compare, merge with data processor, report... in the object context menu. Then choose the external data processor (report) in the standard dialog box for choosing the file.

The Comparison and merging... window will be displayed. The operations in this window are the same as in case of configuration merging.

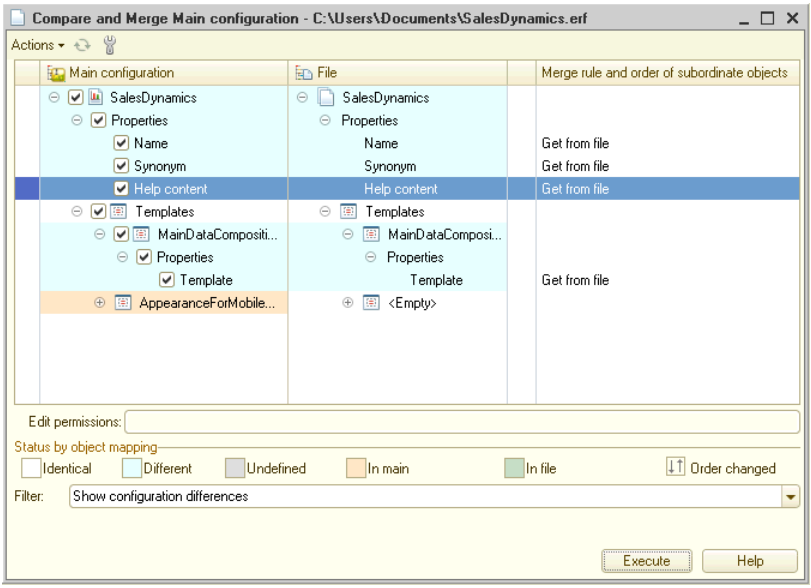


Fig. 120. Data processors merging

To compare or merge the external data processor (report) with another one, open the source external data processor (report) and click Actions in the editing window. Then choose Compare with external data processor, report in the drop-down menu. Choose the external data processor (report) in the standard file choose in dialog box.

5.13. Charts of characteristic types

5.13.1. General description

When developing applications you often need to store properties of the applied objects with the unknown content and type (on the application development stage).

Examples of such properties:

- For products or services it can be color, size, dimensions, and so on.
- For the contractor you can specify federal district, where their contractor is located, or the regular customer flag and so on.

- For a chart of accounts you can specify the quantity and type of profiles for analytical accounting (for the specific account). Then the specific analytical accounting profile values are specified for each posting.

When developing, the content and type of data stored is not specified for such properties (hereinafter **characteristics**). It is also possible that the list of such characteristics will be quite large and each application user will have to operate it. Please note, that the characteristics can be different for various objects groups, for example for one products or services groups you can specify color and size, while for other - only dimensions. It's also worth mentioning that characteristics can be entered during operation by the users.

There are two ways to do this:

- A separate attribute of the required type is created for each characteristic in the relevant object. This way has several disadvantages:
  - Each characteristic is the separate field in the database table, which will take up space in the database - it doesn't matter, whether the characteristic is used.
  - When there are a lot of characteristics, there will be a lot of "redundant" fields, and it's not convenient for both the developer and the user to operate them.
  - When you need to add the new characteristic, you will have to customize the application: add new attribute and account the new attribute everywhere in the application. This requires a lot of effort and is inconvenient.
- Using the special objects, which describe characteristic name, type and so on. This option doesn't have the disadvantages of the previous one: to add the new characteristic you don't have to change the application and it doesn't take up space in the database. The user can also add the new object characteristics. The 1C:Enterprise provides such object. It's called **chart of characteristic types**.

  However, there are some disadvantages. The characteristic organization and usage logic is more complex. You will only have to spend more because of the more complex way once - when developing the application. While the number of characteristics and types increases, the additional customization of an application is not required.

  The chart of characteristic types can be used to organize the analytical accountant. In this case each element (with the unique set of types) will be the extra dimension type and when editing the account in the chart of accounts you can specify the extra Dimension types used for the specific account. When generation the posting, the extra dimension components (of the relevant type) will be used for filling, irregardless of the extra dimension type set when editing the chart of accounts.

5.13.2. Configuration object characteristics

5.13.2.1. General information

The chart of characteristic types is the central element of characteristics mechanism.

The chart of characteristic types stores accept of types, which can be used for the chart of characteristic types elements (**characteristic type**) - Characteristic value type field. The chart of characteristic types also includes information on the opportunity to create characteristics, in case the infobase lacks the objects required to describe values of such characteristics. The Additional characteristic values field and specifying the required type in Characteristic value type list is used to do this.
When creating a characteristic type (the chart of characteristic types element) you can specify, which set of types will be the limit for values of this characteristic type. The set of types specified for the chart of characteristic types in general can be limited for the specific characteristic type but it can't be expanded.

For those metadata objects, for which you plan to implements that characteristic accounting, use Additional metadata object characteristics dialog box to specify is rules for filtering the characteristic types and rules for getting **characteristic values** what specific data objects. In fact, the following rules as specified:

- How to get the characteristic types for specific data object - filtering rules. The 1C:Enterprise uses the specified rules to create a request for getting the list of characteristic types. This request is generated using the `ALLOWED` keyboard. As a result, only those characteristic types will get to the list, for which the user has the right. The rights of a specified by the data access restriction set for the table with the characteristics list.
- How to get a characteristic value knowing the data object and characteristic type - rules for getting a value. The 1C:Enterprise uses the specified rules to create a request for getting the list of characteristic values. This request is not independent. It creates that table, which connects to the main request that gets the application data. That's why using or not using the `ALLOWED` keword is defined by the main request.

The object attribute properties used in the characteristic mechanism requires specification of some metadata properties that the 1C:Enterprise needs to automatically prepare the data that for the user.
After all settings are ready and the characteristic types list is generated (as a predefined chart of characteristic types elements in the Designer mode or as values

entered in 1C:Enterprise mode), you can edit the characteristic values for the specific data object.

If the form has the fields for the data object, characteristic type, and characteristic value the editing process is as follows:

- The user chooses the data object, for which the list of characteristics needs editing. For example, some catalog item.
- The 1C:Enterprise generates a list of characteristic types available for this object. When generating the list. the characteristic type filtering rules, specified for the metadata object, are applied.
- The 1C:Enterprise uses the information from the chart of characteristic types to define the types of the edited value and sets the form elements (choice parameter links, type link and so on).
- Now the user can choose the required characteristic types and set the characteristic values.

When using characteristics in reports and dynamic lists the similar operation scheme is used.

In some cases implementation of characteristic mechanism (examples of search implementation are described below) might require the shorter set of objects. However, using the chart of characteristic types is required with any set of objects used.

5.13.2.2. Editing the chart of characteristic types

The structure of chart of characteristic types is similar to the catalog. The chart of characteristic types can be hierarchical. It might have the attributes, tabular sections, and so on. The main difference between the chart of characteristic types and the catalog is the Characteristic value type property. This property describes the list of types, which can be later specified as the value type (one or several) of the specific characteristic type. The ability of type(s) selection doesn't require changing the structure and application code. Note that to set the type of attribute storing the characteristic value connected to the specific chart of characteristic types, there is a special `Characteristic.<ChartOfCharacteristicTypesName>`. The attribute, for which the type is specified, can't be composite.

If the Characteristic value type property includes `Day`, `Date`, `CatalogRef.Products` and `CatalogRef.Contractors` types, you can create `Size` characteristic type with `Number` type or `Supplier` characteristic type with `CatalogRef.Contractors` type. However, you can't create a characteristic type with the `Boolean` type because such type is not included into the Characteristic value type property. Thus, choose the types set carefully when creating the chart of characteristic types.

When choosing the `Number`, `Row` or `Date` primitive types in Characteristic value type property, specify type dimension or content that cover all possible values in the data type editing dialog box. Thus, if you don't specify the decimal part of a numeric type, it will be impossible to enter a fractional number.

IMPORTANT! Changing the list of type, characteristic types used, after entering custom values can lead to loss of user-entered values. For example, if you delete some type from the type list, the characteristic values associated with the delete the type will be cleared, too. In case the deleted type was the only one, the new type of this characteristic type will be all types specified in Characteristic value type property.

Creating a chart of characteristic types is not the only step to implement the characteristic accounting mechanism for metadata objects. Depending on the task, and you have to create additional metadata objects and configure them in a special way. The examples of characteristic mechanism implementation are described below.

5.13.2.3. Examples of characteristic mechanism implementation

1. The simplest example of characteristic accounting mechanism. The separate chart of characteristic types is created for each metadata object. This chart of characteristic types stores characteristic values.
2. In of the examples, the developer needs to create characteristic types, which are not described in the application. That features from the previous example are the same.
3. This example should be used when you need to utilize one chart of characteristic types to describe characteristic types of all metadata objects and one information register to store characteristic values.
4. The example should be used when you need to utilize the same characteristic type for different metadata objects (characteristic sets). The characteristic values are stored in an information register.
5. If you need to limit the access to characteristics and their values using the access restriction mechanism of data applied to the object itself, you can use this mechanism implementation example.

NOTE 1. The examples given in this section are not complete. They are intended to demonstrate various techniques of handling characteristics.
NOTE 2. The example of characteristic accounting above is not complete. It is intended to demonstrate the basic examples of such accountant.

5.13.2.3.1. Example 1

Task

Characteristic types are stored in chart of characteristic types, and each chart of characteristic types "serves" one metadata object. It means that for `Products` catalog the `ProductsCharacteristics` chart of characteristic types is created, and for `Contractors` - `ContractorsCharacteristics` and so on. Characteristic values for specific objects will be stored in information registers. Each information register will store information on data, characteristics types and characteristic values of one "metadata object - chart of characteristic types" pair. To simplify, we will review one metadata object, one chart of characteristic types, and one information register.

Implementation

Create two dimensions and one resource in the information register designed to store characteristic values. Dimensions will be responsible for the object, for which the characteristic value and type are store, and the resource will store the characteristic value itself.

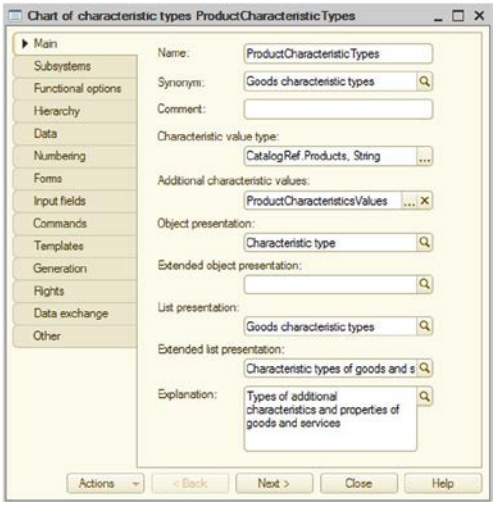


Fig. 121. ProductsCharacteristicValues information register

Let's have a closer look at the register (`ProductsCharacteristicValues`):

- `Object` - dimension. It contains a link to data object, for which is a characteristic value is stored. A dimension will have the type of metadata object, for which the characteristics will be stored. In the current example this value is `CatalogRef.Products`.
  The `Master` property must be set for the dimension. This will allow you to automatically delete object characteristic value when object is deleted and will allow to open the list of characteristics directly from the objects form (the link is located in navigation panel form).
- `CharacteristicType` - dimension. Contains a link to characteristic type (specify `ChartOfCharacteristicTypesRef.ProductsCharacteristics` as the value type). Values from chart of characteristic types will be added to this dimension.
- `CharacteristicValue` - resource. It stores value of one of the types of chosen as **value type** of **characteristic** type specified in the `CharacteristicType` dimension . The `Characteristic.ProductsCharacteristics` will be used as the value type. For this resource you can't specify the compound type that consists of several `Characteristic` types. That's why to store characteristic values (in this particular example) several information registers must be used (one for each "data object - chart of characteristic types" pair).
  Add the `Filter.Owner` parameter connected to the `CharacteristicType` dimension and `Clear` usage mode to the Choice parameter links property for the resource. This setting during the editing in the form will clear characteristic value when changing characteristic type.

  You also have to fill the Type link resource property. The `CharacteristicType` dimension must be specified in this property. After specifying the connection one of the features of the Chart of characteristic types will be enabled: depending on the characteristic type specified in the `CharacteristicType` dimension the system will automatically change the list of available types for the `CharacteristicValue` resource.

The next step is to describe the connection between an object, for which the characteristics are accounted, the chart of characteristic types, and information register to the 1C:Enterprise. This must be done using the Additional characteristics of metadata object window. It can be called from the metadata object context menu (Characteristics item).

In this window you describe, which object provides characteristic types and how it does it (left part of the window) and where are the characteristic values stored (right part of the window). In the current example:

- Characteristic types are located in `ProductsCharacteristics` chart of characteristic types (Characteristic types field).
- Characteristic type and its presentation will be retrieved from the chart of characteristic types item. The `Link` standard attribute attribute will be used as a key field (Key field property) for searching for the required chart of characteristic types item.

Storage description will be used by 1C:Enterprise when it needs the list of characteristic values for some metadata objects.

For example to get the list a filter values in dynamic list. In this example the following scheme for characteristic value storage is described:

- The `ProductsCharacteristics` information register is used for storage (Characteristic values property).
- To store the object, for which the characteristic values are specified, the `Object` dimension is used (`CatalogRef.Products` type). This is set in Object field property.
- To store the link to object characteristic, the `CharacteristicType` dimension is used (Field of type selection property).
- Characteristic value is stored in `CharacteristicValue` resource (Value field property).

In general, any object with the attributes of the required type can be used as an object for characteristic value storage:

- The attribute of the same type as the object, for which the characteristic value is stored - object field.
- The attribute of the same type as the chart of characteristic types, which stores the characteristic - field of type selection.
- The attribute of the same type as the characteristic value (the characteristic value itself) - value field.

When you edit the fields of Characteristic values dialog box, 1C:enterprise will give you a choice from objects that meet requirements stated above (the Characteristic types field must be filled in the string, where the editing is made). Editing the characteristics is available for the following metadata objects:

- Catalogs
- Documents
- Enumerations,
- Charts of characteristic types
- Charts of accounts
- Charts of calculation types
- Exchange plans
- Business processes
- Tasks

Result

Imlementation of this example will allow to:

- Create characteristic types required for the element of "their" metadata objects in the chart of characteristic types.
- Specify the separate set of characteristic for each `Products` item in the catalog and specify the required characteristic value for each characteristic type.

5.13.2.3.2. Example 2

Task

Let's look at the situation, where the user has to add a characteristic if the application doesn't have the type for this characteristic. For example, you need to consider color for the products item, but there's not such catalog in the application. To complete such task, you have to create a special catalog, which will be the additional characteristic values storage. This catalog will be subordinate to the chart of characteristic types. Then you need to specify this catalog in the special property of the chart of characteristic types - Additional characteristic values. This catalog should also be specified as one of the possible Characteristic value type property type.

Implementation

Let's create a catalog, which will store additional characteristic values (it will be called `CharacteristicValues`). All other application objects will be similar to example 1. Then, perform the following actions with the catalog:

- Specify the `CharacteristicValues` chart of characteristic types as the catalog owner. As a result, the only values set available for the characteristic type is the one specified for the particular characteristic type.

- Specify the `CharacteristicValues` catalog is one of the compound values of the Characteristic value type property of the `ProductsCharacteristics` chart of characteristic types.
- Choose the `CharacteristicValues` as a values of Additional characteristic values chart of characteristic types property.

The settings in the Additional characteristics of metadata object window for `Products` catalog are the same.

Result

Imlementation of this example will allow to:

- Create characteristic types if the application doesn't have the types for storing its values (additional characteristic values).

- Specify additional characteristic values as characteristic values.

5.13.2.3.3. Example 3

Task

This example is applicable when you need to use one chart of characteristic types to describe characteristic types of all metadata objects and one information register to store all characteristic values. Distribution of characteristic types between metadata objects will be performed using the chart of characteristic types attribute. The value of this attribute will define an attachment to the specific object (attribute contains value) or all metadata objects (attribute is not set).

Implementation

You need to create a chart of characteristic types (named `ObjectCharacteristics`). Create `TypeOfCharacteristicType` attribute for the object created and set `Enumeration.ObjectTypes` as an attribute type. The content of enumeration will define the metadata objects, for which characteristic types will be stored.

NOTE. To increase the speed of the filtering by required characteristics type, it's recommended to set the Index property of `TypeOfCharacteristicType` attribute to Index with additional order.

To store attribute values, create information register called `ObjectCharacteristicValues`. Take its content and settings from Example 1 (`ProductsCharacteristicValues` register). The difference from Example 1 is that the type of `Object` dimension will be compound and the types should include all reference types, for which the characteristic accounting is planned.
Similar to Example 2, you should create a catalog for additional characteristic values accounting.

In this example, it's supposed that characteristic will only be accounted for `Products` and `Contractors` catalogs.

Now we have to "explain" the way to filter characteristic types for this or that data object to 1C:Enterprise (it must be done for each object with characteristics):

- In which object is the characteristic list located.
- What key field will be used to get the specific characteristic to identify the value type (characteristic type) and characteristic presentation.
- Which characteristic attribute (and filter value) will be used to define the list of characteristics associated with a specific metadata the object.

Below is an example of characteristic filter rule specification for Products catalog.

The setting in is the same Additional metadata object characteristics window as in the Example 1. Additionally, you should specify the values of two fields:

- In Types filter field property you should specify, which chart of characteristic types attribute is the key field that will be used to filter characteristic type for this or that data object. In this example it's TypeOfCharacteristicType chart of characteristic types attribute.
- The Types filter value property contains a value that should be included into an attribute specified in Types filter field property. This make the chart of characteristic types item (specified in Characteristic types property) "appropriate" for the required metadata object. In this example the Products catalog is configured, so for the Types filter value property the ObjectTypes.Products enumeration value is set.

If the application includes characteristic types that can be used with all objects, for which the characteristics are accounted, 1C:Enterprise can be configured in such way that the list of characteristic will include the characteristic types "attached" to the metadata object as well as common characteristic types. To enable this, add another row in the characteristic settings window.

In this case the "general" characteristic types on the ones, for which the TypeOfCharacteristicType attribute value is not filled. Characteristic value storage is described similar to Example 1.

Extension

In the current example, if you need the characteristics support for new metadata object, do the following:

- Add the required value to the ObjectTypes enumeration.
- Add the type that corresponds to the data object, for which the characteristics accounting is configured, to the Object dimension of the ObjectCharacteristicValues register.
- The properties in Additional metadata object characteristics are the same as in the examples about.

Result

Imlementation of this example will allow to:

- Use the same chart of characteristic to store characteristic types of all objects.
- Divide characteristic types into "attached" to the specific objects and common for all objects.

5.13.2.3.4. Example 4

Task

This example can be used when one characteristic type has to be used for several metadata objects. For example, the Delivery date characteristic type can be used for contractor (for example, it could mean the date of products delivery by the contractor) and for goods (in this case in will mean the date of some product delivery). To implement this task using the previous example, you will have to create several characteristic types attached to specific metadata objects and with the same names (to simplify identification).
The scheme that meets the above requirements looks as follows:

- The chart of characteristic types will keep the characteristic type list.
- That sets of characteristics are stored in predefined data of an object (catalog). It is the list characteristics, which can be set to metadata objects and for which the characteristic specification is enabled. The list of available characteristics is set in 1C:Enterprise not the Designer.
- The characteristic values are stored in an information register.

Implementation

Let's have a closer look at this scheme. We'll take a `Products` catalog as an example. The chart of characteristic types is the same as in the previous examples. To store the acceptable characteristics sets. you should create a dedicated catalog (named `CharacteristicSets`). In this catalog, create a `SetContent` tabular section, in which you should only create one attribute, `CharacteristicType` (a link to `ChartOfCharacteristicTypesRef.ObjectCharacteristics` chart of characteristic types), and as many predefined items as there are metadata objects, for which the characteristics accounting is planned. In the current case, this catalog must include one item for `Products` catalog.

Information register, in which the characteristics values of objects are stored, is the same as the register suggested in the Example 3.
The next step is establishing connection between metadata object, the object storing characteristics list, and the object storing values of the selected characteristics.

As a result, the following scheme is created:

- The presentation and characteristic value types are generated using the indirect addressing. The Key field property of the characteristic setting window is used for this.
- To get the list of available characteristics (for `Products` catalog), use the predefined `Products` item of the `CharacteristicSets` catalog. To

enable this, it's specified in the Types filtering field property that the required item from CharacteristicSets catalog will be sorted by Link key field. The filter value (Types filter value property) is the link to predefined item.

- Storage of characteristics values is not the same as in the above examples.

The discussed scheme for storing characteristic list doesn't provide automatic generation of characteristic choice list when adding a new characteristic to an object (from the list of available ones). To generate such list, you should generate a choice form for a metadata object from the chart of characteristic types and you should do it in a specific manner.

To do this, use the wizard to generate choice form for the metadata object from the chart of characteristic types. In the created form, set the Custom query check box for the main form attribute (List attribute of the DynamicList type) and set the following request as a source of the dynamic list:

```
SELECT
  ChartOfCharacteristicTypes.Link,
  ChartOfCharacteristicTypes.Code,
  ChartOfCharacteristicTypes.Name,
  ChartOfCharacteristicTypes.ValueType
FROM
  ChartOfCharacteristicTypes.ObjectCharacteristics AS
ChartOfCharacteristicTypes
    {INNER JOIN Catalog.CharacteristicSets.SetContent AS SetContent
    BY ChartOfCharacteristicTypes.Link =
CharacteristicsList.CharacteristicType}
{WHERE
  CharacteristicsList.Link.* HOW CharacteristicsSet}
```

You should also specify ChartOfCharacteristicTypes.ObjectCharacteristics as the main table of the dynamic list and set the Dynamic data reading check box.

In the form module, specify the OnCreateAtServer event handler:

```
&OnServer
The OnCreateAtServer(Denial, StandardProcessing) procedure

  Predefined = New Map;
  Predefined.Insert(Type("CatalogRef.Products"),
Catalogs.CharacteristicSets.Products);
  If Not Parameters.Property("CharacteristicTypesFilterObject") Then
    Return;
  Else
    FilterObject = Parameters.CharacteristicTypesFilterObject;
    FilterObjectType = TypeOf(FilterObject);
    If Predefined.Get(FilterObjectType) <> Undefined Then
      NewItem =
List.Filter.Items.Add(Type("DataCompositionFilterItem"));
      NewItem.LeftValue = New
DataCompositionField("CharacteristicSet");
      NewItem.ComparisonType = DataCompositionComparisonType.Equal;
      НовыйЭлемент.ПравоеЗначение = Предопределенные[ТипОбъектаОтбора];
      NewItem. Use = True;
    EndIf;
  EndIf;

EndProcedure
```

Let's review the features of choice form implementation in more detail. To specify, which item of `CharacteristicSet` catalog should be used to get the list of available available characteristics, use the `CharacteristicTypesFilterObject` parameter of the form. This parameter is present in extension of the managed form for the chart of characteristic types. If the form parameter is specified (as it was done when creating `CharacteristicType` attribute of the `Characteristics` tabular section of the `Products` catalog), the type of this parameter is analyzed and it's verified, whether there is a predefined item of the `CharacteristicSets` catalog for this type (in accordance with the `Predefined`, filled "manually"). If the required type is set in the map (meaning that there is a required predefined item) - the `CharacteristicSet` dynamic list filter is set. If the `CharacteristicTypesFilterObject` form parameter is not set or the predefined value is not set for the value type specified in the parameter - the filter is not set and the whole list is displayed.

Please pay attention to the way the request, which is used as a source for the dynamic list, is generated to implement the described behavior. Such behavior is possible because the filter and join are connected as a constructions for expanding request language of data composition system. In case the filter is not set, the data composition system (used to operate dynamic list) will not include the join or filter in the resulting request. This means that all characteristic types will get to the choice list. Such behavior will be used to add the characteristic type to list of available characteristics.

Extension

If the characteristics support is required for one more object in the example above, do the following:

- Create new predefined object in `CharacteristicsSet` catalog.
- For a "new" metadata object, a connection between the configuration object and `CharacteristicsSet` catalog (similar to the example in this section) must be set up.
- In the chart of characteristic types choice form you should add the required filling code of the `Predefined` map. For example, for `Contractors` catalog, the added row will look as follows (predefined item of the `CharacteristicsSet` catalog will be called `Contractors`):

```
Predefined.Insert(Type("CatalogRef.Contractors"),
Catalogs.CharacteristicSets.Contractors);
```

- Extend the `Object` dimension type of `ObjectCharacteristicValues` information register by adding the type of the "new" metadata object.
- In 1C:Enterprise mode, fill the tabular section of the new predefined item with the acceptable characteristic types.

Result

Imlementation of this example will allow to:

- Keep the single characteristic types list for all metadata objects, which require characteristics record keeping.
- In 1C:Enterprise mode, fenerate the list of acceptable characteristic types for each metadata object.

5.13.2.3.5. Example 5

Task

If you need to limit access to characteristics and their values using the data access restriction applied to the object itself (for example, a user can change the product characteristics only if he or she can change the product), you can store characteristics (and their values) in tabular sections of catalogs, documents and other objects, for which the tabular section can be created.

There are several features of this storage method:

- If you need to use the same characteristic type to operate several metadata objects (for example, `Supplier` characteristic can be assigned to a product and a contractor), then storing the characteristics in the tabular section will allow to use the same item of the chart of characteristic types in both cases. Otherwise, you will have to create two characteristic types and particularly specify, which metadata object is associated with the particular characteristic type.

- The full-text search will find the specific object value when searching by characteristic value. If the characteristic values are stored in the information register, you will find the information register record and you will only be able to access the required object from this record.

Implementation

The scheme that meets the above requirements looks as follows:

- The chart of characteristic types will keep the characteristic type list.
- That sets of characteristics are stored in predefined data of an object. It is the list characteristics, which can be set to metadata objects and for which the characteristic specification is enabled. The list of available characteristics is set in 1C:Enterprise mode not the Designer mode.
- In the object, for which you can specify characteristics, there's a specific tabular section that stores characteristic type (the the list of acceptable ones) and the value of the characteristic.

Let's review the scheme in more detail. We'll use `Products` catalog as an example. The chart of characteristic types is the same as in the previous examples. To store characteristic values, you must create a tabular section (for example, called `Characteristics`) in `Products` catalog. In this case the tabular section is similar to information register, which was used in the above examples to store the characteristic values. Create two attributes in the tabular section:

- `CharacteristicType` - link to the chart of characteristic types used (`ChartOfCharacteristicTypesRef.ObjectCharacteristics` type). This attribute will include link to the characteristic.
- `CharacteristicValue` - value of characteristic (`Characteristic.ObjectCharacteristics` type). It stores value of one of the types of chosen as value type of characteristic type specified in the `CharacteristicType` attribute.

The setting of tabular section attribute properties is similar to setting dimensions and register resources used to store characteristic values.
For the `CharacteristicType` attribute, add the `CharacteristicTypeFilterObject` parameter connected to the `Link` dimension and `Clear` usage mode to the Choice parameter links property. Search setting when choosing the characteristic type will provide display of characteristics specified in relevant predefined item of `CharacteristicsSet` catalog.

For `CharacteristicValue` attribute, do the following:

- Add the `Filter.Owner` parameter connected to the `Characteristics.CharacteristicType` attribute and `Clear` usage mode to the Choice parameter links property. This setting during the editing in the form will clear characteristic value when changing characteristic type.
- The Type link property must be set to `Characteristics.CharacteristicType`.

Organization of storage of the used characteristics list for each metadata object (as well as the dedicated chart of characteristic types choice form) is the same as in the Example 4.

Extension

If the characteristics support is required for one more object in the example above, do the following:

- Add the `Characteristics` tabular section to the "new" object similar to the example in this section (specify all the required attribute properties).
- Create new predefined object in `CharacteristicsSet` catalog.
- For a "new" metadata object, a connection between the configuration object and `CharacteristicsSet` catalog (similar to the example in this section) must be set up.
- In the chart of characteristic types choice form you should add the required filling code of the `Predefined` map. For example, for `Contractors` catalog, the added row will look as follows (predefined item of the `CharacteristicsSet` catalog will be called `Contractors`):

```
Predefined.Insert(Type("CatalogRef.Contractors"),
Catalogs.CharacteristicSets.Contractors);
```

Result

Imlementation of this example will allow to:

- Keep the single characteristic types list for all metadata objects, which require characteristics record keeping.
- In 1C:Enterprise mode, generate the lists of acceptable characteristic types for each metadata object.
- Apply data access restrictions set for the data objects to restrict access to characteristic values.

5.13.3. Extra dimension of analytical accounting

The chart of characteristic types allows to organize analytical accounting based on the chart of accounts. To do this, specify, which chart of characteristic types

contains extra dimension types of accounting. Then to set the specific extra dimension type only the values from the specified chart of characteristic types will be chosen. To set the extra dimension types on the development stage - specify predefined values in the chart of characteristic types. If the extra dimension types are not set on the development stage, specifying predefined values is not obligatory.

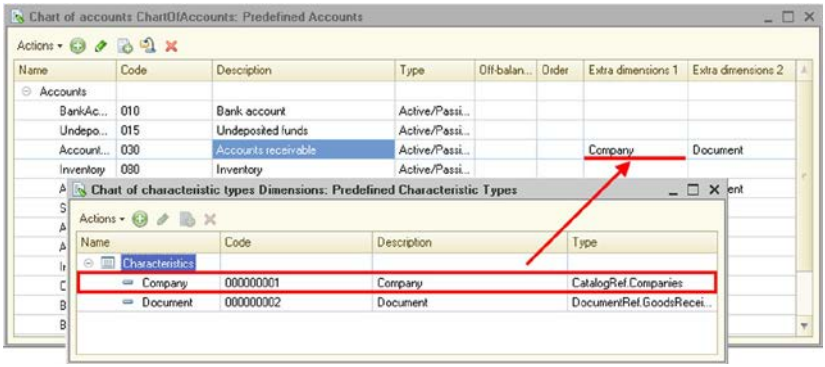


Fig. 122. Using extra dimension types

On the picture you can see that account 40 called Products output (operations, services) has two extra dimensions - `ProductGroups` and `PriceTypes`, which are chosen from the list of predefined characteristic type defined in this chart of characteristic types.

5.13.4. Using characteristics in application

5.13.4.1. Automatically generated forms

In case metadata objects participating in characteristics operation are set with all recommendations above in mind, you don't need to create additional forms to view and edit characteristic types and values (except Examples 4 and 5 with manually created forms). Automatically generated forms will have the required features:

- In object form on the navigation panel there will be a command for jumping to the list of its characteristics;
- Displaying characteristic types list;
- Displaying characteristic values list;
- Editing characteristic values (with automated filters and links of the edited values).

NOTE. Choosing tabular section of the catalog (document and so on) as the storage object will not result in enabling automated choice list generation when adding characteristics for the specific object. Such list must be generated programmatically.

5.13.4.2. Using characteristics in reports

Characteristics can be used in reports for:

- Display in report (chosen field),
- Using in filter,
- Using for sorting,
- Using for grouping,
- Using in conditional appearance.

To use metadata object characteristics in the report, the list of available data composition scheme field lust include the field with the type of link to that metadata object and usage of field attributes **must not** be **denied** for this field.

Let's look at the simple report with `Goods` field of `CatalogRef.Products` type. The `Products` catalog item has attributes and standard attributes.

In 1C:Enterprise mode, several characteristic types were created for this catalog. When you need to use some characteristics (for example, Bold and Size) as the report field, the attribute list of the Goods catalog will include attributes, standard attributes, and the list of characteristic types created for the `Products` catalog.

Find the required characteristics type in the attributes list and add it into the list of chosen fields.
The characteristics are used in other components of the conditional appearance in the same manner (filtering, sorting, grouping, conditional appearance).

5.13.4.3. Using characteristics in dynamic lists

5.13.4.3.1. General information

Characteristics can be used in dynamic fields for:

- Using in filter,
- Using for sorting,
- Using for grouping,
- Using in conditional appearance,
- Displaying in the separate list column.

The setting of dynamic list column display is performed in form setting dialog box (All actions - Edit form), while filter, sorting, grouping and conditional appearance are configured in list settings dialog box (All actions - Configure list).
Remember that the main way to use the characteristics in dynamic lists is "dot separated" usage (i.e. in filters, groupings and so on). However, in case of need you can include display characteristics in the main table of dynamic list.

5.13.4.3.2. Using in filters an so on.

Using characteristics to configure the list requires reference-type column for the required metadata object in the available dynamic list fields.

To use the characteristic value when setting the dynamic list (filter, sort and so on), do the following (in 1C:Enterprise mode):

- Open dynamic list settings dialog box;
- Expand the reference field (in the available fields list);
- Choose the required characteristic to be used in the setting.

5.13.4.3.3. Including in columns

To display characteristics in dynamic list columns, include column (can be invisible) connected to reference-type data of the required metadata object.

To add a field displaying a characteristic value, do the following (in 1C:Enterprise mode):

- Open form setting dialog box;
- Put a pointer on a reference-type hidden field;
- Click Add fields;
- Choose the required characteristic type.

Now the added characteristic types will be displayed as the dynamic list columns.

5.14. Registers

5.14.1. General information

1C:Enterprise registers are designed to store and process data on business or administrative activities of the enterprise.

Document and Catalog infobase objects store information on objects of subject domain, such as employees, goods, resources, and currencies. Each database object corresponds to the subject domain object.
Registers usually store information about changes in the states of objects or other information that does not reflect the objects of the subject domain directly. For example, the registers can store information about exchange rates or information about the receipt and consumption of goods.
A database object exists independent of its attribute values and is valued by itself. For example, an employee can change last name, ID number or any other attribute. However, it will be the same person.
After deleting, the object can't be recreated. Even if you enter the same attributes as in the deleted object, it will be a new object. 1C:Enterprise stores an internal object identified - reference. The reference is unique within the infobase. There can't be two objects with the same reference during the infobase lifetime. The references of deleted objects are reassigned to the new objects. 1C:Enterprise allows to store links to database object in database fields.
The information storage unit in registers is a record. The payload of the register record is defined solely by data stored in it. For example, the record on the exchange rate by itself is worthless. It doesn't correspond to any object in the subject area. The important part is that it contains currency, date, and exchange rate for that date. You can remove the record and enter the same one - this will not corrupt the logic. Register records don't have links and links to register records can't be stored in database fields.
This section describes information registers and accumulation registers.

5.14.2. Information registers

5.14.2.1. General information in information registers

An information register is mainly intended to store data - that is relevant to a business task - includes a specific combination of values, and might change with time. For example, if you want to store competitor's prices on the goods you sell, the collected prices data is expanded by goods and competitors. If you want to track price changes and enter data occasionally, the store information will also expand by time.
In 1C:Enterprise there a mechanism to store such data and operate them - information register.
Information register is generally a multidimensional data array, required to implement function, which can display information using the set of arguments. The function arguments are called **dimensions** and function results - **resource**. In the example above the two-dimensional `CompetitorsPrices` register will contain two dimensions, `Competitor` and `Good`, and a `Price` resource. There can be several resources: for example, you can store retail and wholesale prices.

Apart from dimensions and resources, a set of attributes can be created for the information register. Attributes allow to include various information into the register record. Attributes don't affect register resource values and can be used to analyze register records.
Information registers storing data that changes with time are referred to as **periodic** information registers. For periodic information registers, the following standard operations are supported: getting the latest or the earliest value (for example, getting the last entered price for a specific product and a specific competitor), as well as getting a slice of the latest or the earliest values. For example, all the recently entered prices for all products and competitors can be retrieved.
To expand information by time, a `Period` register field is used. It's not entered as a dimension, but added automatically by the 1C:Enterprise when creating periodic register.
Information registers doesn't require dimensions generation. In this case the register will provide a set of periodic data. Such registers can be used, for example, to store last names of officials, who sign documents. During business activities, documents are created and signed by officials, who have the right to sign document at the specific moment. The disadvantage of such method in case of constant value change is that when opening the archive document, the new last name of the official will used that was chosen from the constant. In such cases, instead of constant use periodic information register, which stores information in changes, and in documents the values are chosen from the information register by document date.
The most representative example of one-dimensional periodic value - exchange rate. When performing calculations (for example, when calculating price in rubles by converting from another currency) it's important to know the value at the moment of calculation.
It's crucial to know the exchange rate when calculating something with the past date - in this case, an exchange rate for the past date must be "recalled".
To get such data, you need to create a table with the following columns: currency name, exchange rate date and the exchange rate itself. The rows of such table contain exchange rates of several currencies at the specific date.

Date	**Currency**	**Exchange rate**
31.10.2008	USD	26.5430
31.10.2008	EUR	35.0447
01.11.2008	USD	27.0981
01.11.2008	EUR	34.4092
02.11.2008	USD	27.0793
02.11.2008	EUR	34.4828

When calling such table, remember that the `Exchange rate` column contains specific exchange rate at the date and it's supposed that the rate doesn't change until the new exchange rate. That's why to get an exchange rate for intermediate

date you need to take a rate for the closes later date with the recorded exchange rate.
Also, keep in mind that various value in the Currency column practically means that several exchange rates are tracked. In other words, the table above can be presented in different way.

Date	USD exchange rate	EUR exchange rate
31.10.2008	26.5430	35.0447
01.11.2008	27.0981	34.4092
02.11.2008	27.0793	34.4828

There can be several exchange rate columns in the table (as many as the number of currencies to be tracked).
If the register is not periodic, the Period field is not created. In the example above the CompetitorsPrices register can be non-periodical, if you want to have the current prices, not just store the price changes log. In this case the register function will be able answer: "What is the competitor's price for some product", but it won't be able to answer: "What was the competitor's price for the product at the beginning of the year".
From the described information register operation principles it's clear that there can only be one record with the specific set and period of changes. Indeed, one product of one competitor can only have one price. If for some reason you can't get several prices and want to enter this information into the database, create another dimension to store the value, by which the prices may differ. For example, you can create InformationSources dimension. Then you will be able to enter competitors prices broken down by sources.
The uniqueness of records by dimension set is fundamentally differs information registers from accumulation registers that allow to enter several values with the same dimension and period values.
If when operating the register the most often usage is getting data at the first or the last (current) point of time, allow the system to maintain totals for the last ones (Allow totals: slice last property) or the first ones (Allow totals: slice first). As an example of such usage we can refer to a register with sale prices. The price may change with time but retrieving a current price is the most frequent request to the register.
The totals of information register will be used in case the following conditions are met:

- Usage of totals is allowed for the register in configuration;
- Usage of totals is allowed for the register in 1C:Enterprise mode;
- You get the data for the first date (slice first) of the last (current) date (slice last) without specifying the period;
- Conditions of SliceFirst and SliceLast virtual tables are only set for dimension and separator values in Independently and simultaneously mode;

- In data access restrictions you can only use dimensions and separators in Independently and simultaneously mode.

If all conditions are met, then to get the information form the register the regular request will be used.

5.14.2.2. Information register records

Information register rows containing information on resource values for dimension values and specific period are called **records**. The **record key** is used to identify information register records. You can enter records in information register:

- manually,
- using documents.

These two options affect the information entering method, not the main register operation logic.

The document used to enter a record into the information register is called recorder.

Recorders that are recorded independently can be edited manually or with 1C:Enterprise language means. If the dimension of such register is set as "leading" and the dimension value is the link to database object, it's considered that the register record only makes sense while this object exists. For example, if the `Competitor` is set as the leading dimension, it's considered that the record only makes sense as the information in this competitor. When the competitor data is delete, all records about it will be deleted automatically.

If the register is recorded by the recorder it means that records will be strictly subordinate to the recorders - documents. Usually it means that the records will be generating when the documents are posted. Hence, when the document is deleted, records are deleted automatically. As opposed to the leading dimensions, there can only be one recorder.

Remember that information register record key, created with `EmptyKey()` method is not the same as the information register record key with the default dimension values for "their" types.

5.14.2.3. Editing information registers

The Information registers branch of the configuration tree is used to operate information registers.

When the register is recorded, its properties are defined and register structure is developed: the sets of dimensions, resources and attributes of the register, screen forms for viewing and editing register records, and register print forms (if required) are created.

This section includes description of specific information register properties that are different from the general configuration object properties.

The register is edited in Information register editing window. The register properties are located on the tabs.

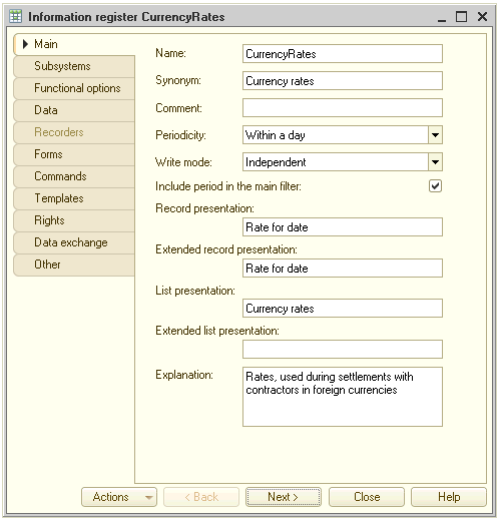


Fig. 123. Information register editor

Periodicity - this property specifies how frequently the register can save resource values that change with time.
This property determines whether values of the register resources can be retrieved using 1C:Enterprise script methods. For non-periodic register you can only retrieve the last entered register resources value - there is no information on the past periods in such register. For periodic registers, it is impossible to retrieve a value with periodicity less than that of the register.
Periodicity does not depend on register editing method.
Write mode - the property defines the way the records are written: independently (for example, manually) or subordinate to the recorder (for example, documents).
If the register is set as the periodic one and independent write mode is specified, the Main filter by period property becomes available. If the check box is checked, then apart from the main register dimensions and attributes, participating in record filtering when registering changes, you can also specify filter by Period field.
On the Data tab, the register data structure is generated. Dimensions, resources, and attributes are created.

On the Recorders tab, the recorders list is managed. The tab is available if the Write mode property is set to Recorder subordinate.

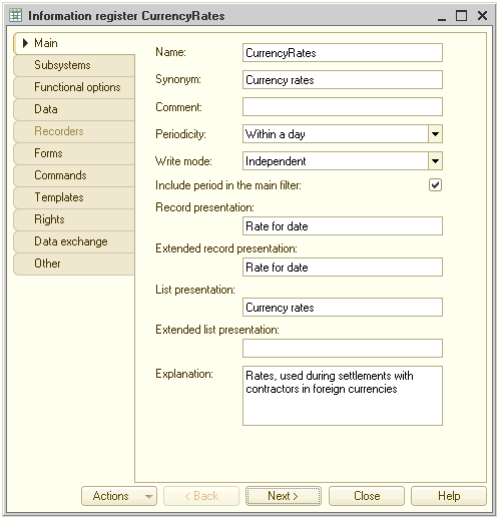


Fig. 124. Setting information register recorders

In the upper list, you can manage recorders list (set or unset the check boxes), and in the lower list, the objects that are recorders are listed.
The recorders list for other types of registers is managed similar to the described information register recorder list management mechanism.
The distinctive feature of information register resources is broader data typification. Resources in other types of registers can only be numerical.

5.14.2.4. Information register structure development

The register structure development includes creation of dimension, resource, and attribute sets.
To manage the list of dimensions, resources, and attributes of the register and edit their properties, use the controls of Dimensions, Resources, and Attribute groups of the Register editing window. In terms of settings, the controls of this group are the same.

5.14.2.4.1. Information register dimension (resource, attribute) properties

Properties of dimensions, resources and attributes are editable in the property palette. In general they are the same as configuration object properties. Further in this section, the uniqie properties of dimensions, resources and attributes will be described.

Leading - set this property for dimensions with specific data type - link to configuration object. In this case, the information register record only makes sense when this object exists. When deleting the object, the associated records will be automatically deleted from the object.
Deny empty values - setting this check box enables mechanism that denies register record with empty dimension value.
Index - this dimension property can be edited if the dimension is not leading. For dimensions, resources, and attributes with Index property set, the dedicated index is created, which improves performance when operating the register. The index is always created for leading dimensions.
When viewing the register in 1C:Enterprise mode, you can sort register records by indexed dimensions, resources, and attributes. The number of forms required to view and edit the register must be generated during configuration development.

5.14.2.4.2. Information register dimension list sorting

The information register dimensions order is important. The dimensions, which require fast access must be located at the top of the list.
It affects the ability to apply 1C:Enterprise language methods that use position-based access to dimensions.
Also, remember that changing dimensions order required restructuring of the infobase.

5.14.3. Accumulation registers

1C:Enterprise registers are used to accumulate information on the number and changes of some values - tangible, financial, and other. All information on business activities entered using documents or generated with calculations must be accumulated in the registers. Then this information can be retrieved, analyzed, and presented to the user in report forms.
This section describes the accumulation register definition and gives information of the basics of accumulation register operation.

5.14.3.1. General information in accumulation registers

Accumulation register - a configuration object designed to store register and totals changes.
The issue that usually arises when creating a "storage" of accumulated information is defining its structure: where to accumulate summary data to extract it without too much processing. 1C:Enterprise uses simple and flexible solutions to create accumulation registers: you only set, in which sections and which data

to store in the register, and 1C:Enterprise will provide writing and retrieval of the required data with the simple language methods.
1C:Enterprise language methods allow to retrieve balances of accumulation register at the specified point of time. There are filters by dimension values and balances broken down by other dimensions.
Consider an example. Suppose that data on number and price of each good in each warehouse will be stored in the created inventory accounting software. The information to be retrived is: "balance of the good in the warehouse", "balance of the good in all warehouses", "price of all goods in the warehouse".
In 1C:Enterprise methodology, accumulation register of such type is the Cartesian coordinate system with warehouses on one axis and goods - on the other one. The crossing of the warehouse and the product contains numbers: quantity and prices.

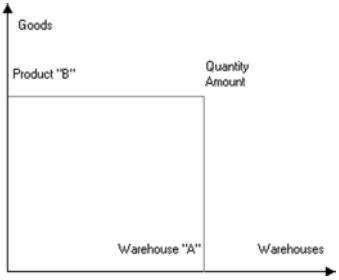


Fig. 125. Accumulation register

The physical point of accumulation register is quite hard to describe and, most likely, accumulation register doesn't have the tangible equivalent.

IMPORTANT! So, let's define: accumulation register - n-dimensional coordination system, with summary data in its nodes. The axes of such coordination system will be called register dimensions and the stored data - register resources.

5.14.3.2. Accumulation register records

The accumulation register state usually changes with document posting. The document posting procedure is located in the document module and includes the algorithm for generating information on changes in the registers that has to be made when posting a document. This information is called register records. The totals calculation mechanism uses register records to make direct changes in accumulation register. So, register records only include increments (positive or negative) of register resource values, not the value totals.
The specialist configuring the system can provide the end-user with means to view register records. The Designer allows to create screen and print forms to view and analyze register records.

When developing the configuration, you can create unlimited number of accumulation registers. However, remember that writing changes in a large number of register.
Apart from dimensions and resources, a set of attributes can be created for the accumulation register. Attributes allow to include various information into the register record. Attributes don't affect register resources values and can be used to analyze register records.

5.14.3.3. Accumulation register totals

As described above, the register changes are made with register **records**. Register records affect its totals. **Totals** - summary register information that is retrieved by totaling values entered by register records.
The accumulation register totals can be presented as a table with as many columns as the total of dimensions and resources of accumulation register. The number of rows in accumulation register totals table will depend on the number of unique dimension values and will not depend on the number of unique resource values.

Product	Warehouse	Count	Sum
Table	Retail	10	5000
Table	Wholesale	5	2500
Cabinet	Temporary	7	10500
Cabinet	Wholesale	2	3000
Cabinet	Retail	10	a

From the table it's cleat that the `Good` dimension takes Table and Cabinet values and `Warehouse` dimension - Temporary, Wholesale and Retail values. `Count` and `Sum` columns displaying accumulation register resources contain quantity and total of each good in each warehouse.
In contrast to register records, you can't directly view the accumulation register totals. To call the totals, a number of reports can be created in configuration that will call the totals and display them as product reports, bin cards, invoices and so on.

5.14.3.4. Balance registers and turnover registers

There are two types of accumulation register in 1C:Enterprise: balance registers and turnover registers.
For balance register, the 1C:Enterprise language methods allow to retrieve the accumulation register totals at the specified point of time. There are filters by dimension values and balances broken down by other dimensions.
Turnover registers are designed to store information, for which the concept of balance is pointless, for example, totals of sales broken down by customers.

Let's take an example: settlements with customers buying goods that the enterprise produces or sells (to service customers and so on). You can say that implementing such accounting - a must for general accounting in any enterprise.
To quickly retrieve current data on the mutual debts of the enterprise and customer, the Settlements register should be used, which stores sum of debt for each customer. With each business activity, the registry state will change accordingly reflecting the current settlements status. The Settlements register - a balance register.
However, you can't retrieve data on some customer's purchase volume in a period of time from Settlements register - it doesn't store such information. That's why you will need to put in an additional efforts to get this: for example, you can include Contractor attribute into the register structure and them sort the register records by the required contractor and find out the overall purchase sum. But when you need to retrieve these data quickly (for example, when the customer reached specific purchase total, a discount should be applied), such method can't be used, obviously.
In this case, the turnover register can be the right solution. In this register - let's call it Purchase volume - the data on purchase volume for each customer is stored (the customer turnover).
From now on, each business activity will change the status of Settlements register as well as Purchase volume register. When the customer buys a product, the information of the purchase total will be added to this register. As a result, the Purchase volume register will constantly accumulate information on the overall purchase volume of the customer.
The above information will help to conclude on advantages of registers.
First of all, registers are used to store information when the quick access is required. The "promptness" and practicability of registers usage must be defined by the specialist configuring 1C:Enterprise based on the users demands.
Registers also allow to retrieve more accurate information on the state of funds. The processes for document saving and register records writing are separated (the document can be saved without posting), so the document data and information in registers might differ. The register, as opposed to the document, is the storage for final information, so the register records are used as a proof of business activity.

5.14.3.5. Main accumulation register properties

The Accumulation registers branch of the configuration tree is used to operate accumulation registers.
Editing the properties of Accumulation register-type objects and creation of the subordinate objects is performed in the editing window.
When the accumulation register is recorded, its type is defined and register structure is developed:

- the register dimensions, resources, and attributes are created;

- if needed, screen forms and print forms are created for viewing register records.

NOTE. The values of `UUID`, `BinaryData` types and rows of unlimited length can't be used as accumulation register dimension types.

This section describes unique properties of the accumulation register that are different from the general object properties.

Register type - in case the register is designed to:

- store balances (choose `Balances` value from the list);
- store turnovers - `Turnover` value.

Default list form - several forms can be created to view register records. If there are several forms for input and selection, specify the form that will be used as the default in Default list form property.

Enable totals splitting - if this check box is set to `True` (default value), the totals splitting mechanism will be used that provides better performance of concurrent register writing. When several sessions write records at the same time, 1C:Enterprise will not update the same register records. It will write totals changes separately. When retrieving totals, the values will be added. This maintains and provides accurate totals (for example, to retrieve reports quickly) and concurrent records writing. This mode requires additional resources (for example, more data in totals tables are required). That why there are properties in configuration and in the language managing this mode.

The records will be "multiplicated" only when the transaction are performed simultaneously. Their number in each record combination will depend on the maximum number of simultaneous transactions. When recalculating the totals, some records are collapsed.

The totals splitting mode can be changed by the used in 1C:Enterprise mode. The property is enabled by default.

Use in totals - if the check box is set to `False`, the dimension is excluded from the stored register totals.

Several forms can be created to view register records. If there are several forms for input and selection, specify the form that will be used as the default in Default list form property.

5.14.3.6. Turnover accumulation register aggregates

The **aggregates** mechanism is designed to increase system performance when operating the turnover accumulation registers. Aggregates - dedicated storages designed to be used in request mechanism of 1C:Enterprise.

5.14.3.6.1. Basic concepts

To further describe aggregates, we'll use several terms explained below.

Aggregate - physical database table, storing summary turnovers of all register resources by the chosen dimensions with the specified periodicity and in the specific period. The register, for which the aggregates are generated, might contain no more than 30 dimensions.

The aggregates have the following parameters:

- **Aggregate size** - a size of the aggregate table. **Estimated figure**.
- **Effect** - the expected reduction of the average request execution time with the help of aggregates. For example, if the aggregate effect is 90 %, it means that the average request execution time with the help of this aggregate is 90 % less than the average request execution time but using the totals. **Estimated figure**.

Aggregate period - a dates interval, for which the data are stored in the aggregate.

Aggregate periodicity - periodicity of data storage in the aggregate.

Aggregates list - the set of aggregates specified on the configuration stage. The list can be created manually or uploaded form the file generated by calculating the optimal aggregates.

Usage statistics - information on the requests (dimensions, period, periodicity) to the register. Used to rebuild the aggregate and get the optimal aggregate list.

Optimal aggregate list - list of aggregates with optimal size to effect ratio for the current register state (its records and usage statistics).

Aggregates/totals mode - if the aggregates mode is set the aggregate data will be used to execute requests. If the totals mode is set, the totals data will be used to execute requests..

Aggregate usage - disabling aggregates means that when register records are changed, no operations will be made with the aggregates. Disabling aggregates makes sense while the data are massively uploaded to the registers. However, further aggregates enabling might result in resource-demanding process of aggregates updating (in case the changed data fall into the aggregate period).

5.14.3.6.2. General aggregate operation scheme

Let's review the general aggregate operation scheme:

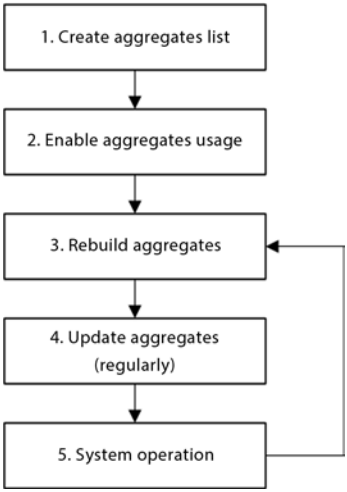


Fig. 126. General aggregate operation scheme (client/server mode)

NOTE. This scheme describes operation of one register. If you need to operate several registers, repeat each step for every register.

Let's review the said operation scheme in more detail:

1. Create the list of the required aggregates in the Designer. This step is optional. The aggregates can be create in the Designer in case you need to ensure constant use of some aggregate in the infobase.
   If you decide to create aggregates in the Designer, this can be done in the following ways:
   - Calculate (for example, in totals mode) and upload optimal aggregates (`DetermineOptimalAggregates()` method). If the aggregates were not used before, the list of optimal aggregates will only be retrieved based on the register records table. If the aggregates were used before, the list of optimal aggregates will be generated based on the records table and usage statistics.
   - Create cusotm aggregate list based on the register requests analysis.
2. Enable aggregate operation mode (`SetAggregatesMode()` method) after updating database configuration.
3. After this you need to rebuild aggregates from time to time (`RebuildAggregatesUsing()` method). When executing this operation, 1C:Enterprise adds the required aggregates and deletes the unused ones. This operation is executed is the current aggregate list is not optimal. Only those

aggregate are deleted that were created during rebuilding. The aggregates created in the Designer are not deleted automatically.

If the aggregates are created in the Designer, the following actions will be made when rebuilding:

- The aggregate will be enabled if the aggregate usage mode is set to `Always`. Assessing aggregate performance doesn't affect its usage.
- The aggregate will be enabled based on the performance assessment in case the aggregate usage mode is set to `Auto`. Using this aggregate is assessed along with aggregate created automatically by the 1C:Enterprise. If the automatically created aggregate is more efficient than the aggregate created in the Designer in `Auto` mode, the automatically created aggregate will be used.

4. Then, update the aggregates (`UpdateAggregates()` mode). Updating aggregates causes data to be transferred from the records table of the chosen register to the aggregate tables. The records created after the previous aggregate update are transferred.
5. Then, accumulate the usage statistics for the created aggregates. To do this, perform typical tasks using data from register with aggregate modes enabled, for some time (for example, 1 month). The aggregates must be updated regularly during operation. After this period, rebuild the aggregates (Step 3) and repeat Steps 3 - 5 on a regular basis.

The scheme above describes approach to register aggregates operation in client/server mode.

In file mode, the aggregates are operated differently:

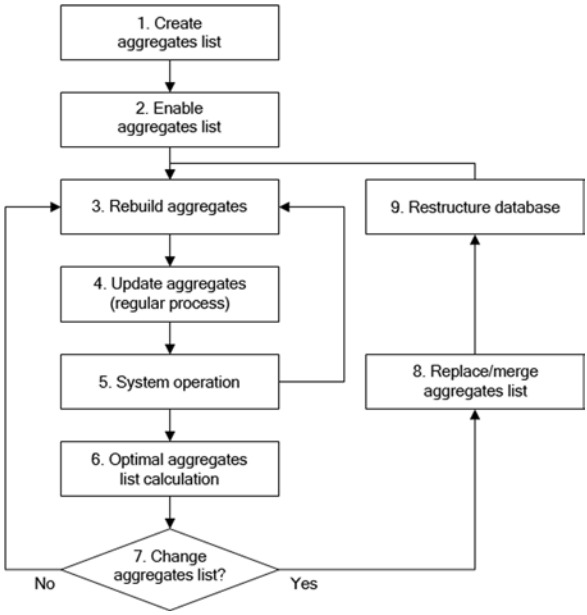


Fig. 127. General aggregate operation scheme (file mode)

The main difference of this scheme form the general aggregate operation scheme (see fig. 127) is the following (only differences are described):

- Creating the list of aggregates in the Designer on Step 1 is compulsory.
- Step 6. Recalculation of the optimal aggregates (`DetermineOptimalAggregates()` method) should be done regularly (for example, once per month). During this period (1 month), perform typical tasks using data from register with aggregate modes enabled. The aggregates must be updated regularly during operation.
- Step 7. After getting the new list of optimal aggregates, you need to define, whether changing aggregate list in configuration metadata is required. If the change is not required, then continue operation and go to Step 3.
- Step 8. If the aggregate list requires updating, upload the required (or all) aggregates (from the list of optimal aggregates). Then update database configuration (the infobase will be restructured) and continue with Step 3.

Recommendations on every step are described in the next section.

5.14.3.6.3. Recommendations on the use of aggregates

Generating aggregates list

The aggregate list can be generated in several ways. If you plan to enable aggregates for the register that already exists in the configuration in case the aggregates usage is not enabled for it, there are several ways to do this:

1. If there are no register records or there are a few of them (no more than 2 or 3 thousand records), analyze the requests, which use the chosen register and retrieve a list of frequently used combinations of dimensions, filter, request periods and periodicity of data retrieval. Use the retrieved information to create aggregate list and try to minimize a number of aggregates used.
2. If there are a large number of records in the register (more than 3 thousand), calculate the optimal aggregates using the register data with `DetermineOptimalAggregates()` method and upload the resulting aggregate list.

If the aggregate usage and aggregate mode are enabled for the register used (i.e. the initial aggregate list was somehow generated), you can retrieve the list of optimal aggregates and upload it in the Designer. The difference between this method and the second method is that in this case to calculate optimal aggregates, the system statistics will be used. That's why the retrieved aggregates list will be more efficient than retrieving the list of optimal aggregate for the register without aggregate modes and aggregate usage enabled.

NOTE. Retrieving optimal aggregates only makes sense if there are 3 to 5 thousand records in the register. If there are less records, the generated aggregate list might be inefficient.

Rebuilding

The aggregate rebuilding operation must be done as frequently as the optimal aggregate list calculation. This operation doesn't imply changes in aggregate list and only uses the current aggregates.

Remember that the rebuilding operation is productive if the volume of accumulated statistical data used is large enough.

It's hard to give general recommendation in the rebuilding period, but we can distinguish several aspects:

- the register data behavior might change;
- there's a suspicion that request nature will change (this will result in changes in accumulated statistics).

Two parameters must be specified when executing this operation:

- Maximum Relative Size - sets limits for the generated aggregate list (in percents of the records table). If the parameter is equal to 0, there are no limitations of aggregates size.

- The minimum effect of rebuilding - a required number of percents in increase resulted from old list rebuilding. If the effect of the new list is increased by the set value, it means that the method actually rebuilds the list. If the parameter is not set or set to 0, it means that there were no minimal effect requirements.

Also, the current aggregate list is rebuilt is it's greater that the Maximum relative size parameter or the list was rebuilt with efficiency that's larger by at least Minimum effect parameter value. Otherwise, the aggregate list is not rebuilt.

NOTE. The rebuilding operation takes quite a lot of time and resources. It's not recommended to complete it during intense operation of the infobase of other users.

Aggregate uodating operations

It's recommended to perform this operation more often than the aggregate rebuilding.

The are two ways to update aggregates:

- Update all aggregates marked as used in full. This might take a significant period of time.
- The "partial" filling. In this case, with each call the period of 1 month is updated in 10 aggregates.

When performing this operation, specify Maximum relative size parameter. It limits the size of the generated aggregate list (in percentage from the records table). If the parameter is equal to 0, there are no limitations of aggregates size.

NOTE. It's better to perform this operation when the infobase load is low.

TIP. We recommend to use totals splitting mode when using the aggregates, especially if the aggregates are updated using the scheduled job while posting many documents associated with the register, for which the aggregates are updated.

Using aggregates

To set the aggregate used for request, the below described algorithm is used.

The used dimensions of the register are defined in the request. Then the list of aggregates, which contain all dimensions used and correspond to the request in periodicity and period, is generated. The aggregate with minimum size is chosen from this list. This is the aggregate that will be used.

A request may contain the aggregate with lesser periodicity than the request requires. In this case the data will be retrieved by combining all data from lesser periods.

It's possible that the period set in the request will not correspond to the aggregates period. In this case two aggregates can be used to complete the request. Consider an example. There are two aggregates closing the whole request period in the infobase:

- With monthly periodicity;

- With daily periodicity.

The request is performed from September 15 to November 15. In this case, two aggregates will be used:

- The aggregate with monthly periodicity will be used to retrieve data from October 1 to October 31;
- The aggregate with daily periodicity will be used to retrieve data from September 15 to September 30 and November 1 to November 15;
- The data of different aggregates will be combined to get the final result.

The virtual table with turnovers of the register, for which the aggregates mode is enabled, always contains current data.

Calculating optimal aggregates

This operation may be performed when required and not on a regular basis. Below is the list of scenarios, which require the calculation of optimal aggregates:

- when the **initial** aggregate list generation period has expired;
- in case of **significant** decrease of performance in the current aggregate list;
- in case of **significant** change of the data nature;
- when the register requests content changes;
- if there is a probability that the current aggregate list is no longer optimal.

NOTE 1. This operation is not compulsory for client/server mode. Generating optimal aggregates (when required) is made automatically during aggregates rebuilding.

NOTE 2. This operation requires the most resources and time. It's highly recommended to perform it when other users don't operate the system.

Using scheduled jobs when operating aggregates

Aggregate rebuilding and updating operations can be performed using the scheduled jobs. When creating a schedule for scheduled jobs, adhere to the following recommendations:

- For a scheduled job performing rebuilding and updating it's recommended to fill the `Key` property of the scheduled job.
- It's recommended to set the updating and rebuilding schedule for scheduled jobs in a way to avoid simultaneous updating and rebuilding.

Keep the recommendations above to avoid concurrent rebuilding and updating and enhance performance.

5.14.3.6.4. Aggregate editing

It's only possible to create and edit aggregate list for turnover accumulation register (Register type property is set to Turnovers). To call the aggregate wizard, use Open aggregates command in the context menu of the register.

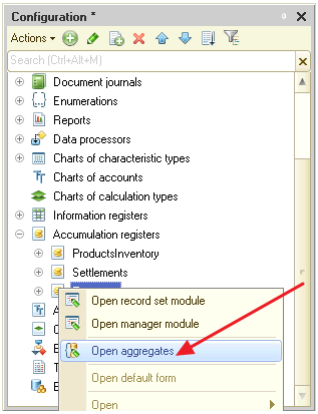


Fig. 128. Calling aggregate wizard

The aggregate wizard window will open, where you will be able to manage turnover accumulation register aggregates. The aggregates can be managed manually or they can be uploaded (there's a dedicated command bar button for this) from the predefined list of optimal aggregates.

Fig. 129. Aggregate wizard

When creating aggregates, a usage mode can be set. If the usage mode is set to Auto (by default), 1C:Enterprise will automatically define, whether the aggregate should be used when rebuilding aggregates. If it's set to Always, 1C:Enterprise will always use the aggregate.

The Periodicity column defines the minimum period of time for storing totals of the chosen dimensions. There can be several aggregates with the same set of dimensions but different periodicity. Don't generate too much aggregates. A large number of aggregates can result in excessive database size, but it won't improve requests performance.

In the right part of the window you can set, which dimensions are included into the edited aggregate. An aggregate may include any number of dimensions (no more than 30) or don't include any. In this case 1C:Enterprise stores free register turnovers with the set periodicity.
If there's an XML file with the list of optimal aggregates, it can be uploaded. To do this, use the aggregate wizard command (see fig. 129) and choose the file.

Fig. 130. Uploading optimal aggregates list

1C:Enterprise will compare the aggregates from the list and the current list of aggregates. It will highlight the aggregates that are recommended for adding (from the Optimal aggregates list) and aggregates that are recommended for deleting (Aggregates list). The recommendations are not mandatory and can be followed partially.

5.14.3.7. Accumulation register structure development

The structure development is creating sets of dimensions, resources and attributes on the Data tab.

5.14.3.7.1. Accumulation register dimension (resource, attribute) properties

Properties of dimensions, resources and attributes are editable in the property palette. In general they are the same as object properties. Further in this section, the uniqie properties of dimensions, resources and attributes will be described.
Type. As opposed to the dimensions and attributes, Resource-type objects can only store one type of data - Number.
Prohibit unfilled values - setting this check box for the dimension enables the mechanism that prohibits writing the register record with an empty dimension value.
Index - this property can only be set for dimensions. Setting this property allows to speed up the operations that call the register data, for example, is the records for specific dimension value are chosen. Such operations include requests with condition of equality of the current dimension and the specified value, as well as temporary calculation and 1C:Enterprise language methods to cycle

`AccumulationRegister` object records that use filtering by the specific dimension value.

Use in totals - if the property is not set, the dimensions are excluded from the stored register totals (the property is only used for the stored turnover register dimensions). In case such dimension is used in request of virtual table condition, the virtual table will not use the stored totals and will calculate data using the records table.

5.14.3.7.2. Accumulation register dimensions list sorting

The accumulation register dimensions sorting affects optimixation of access to register totals: the dimensions, which require fast access must be located at the top of the list.

Chapter 6. Command interface

Command interface - this is the main way of user access to the application features. It allows to move between the forms and perform actions. Configuration developer doesn't write the global interface and form command bar commands in detail for each role implemented in the application (or combination of such roles), but writes the rules, under which the command interface will be automatically generated for the user.
Command interface is generated depending on the user rights, application functional options and user settings.
Such declarative description of the command interface helps to align commands available for the user and user rights. It also helps to modify application command interface upon partial implementation of application abilities.

6.1. General command interface structure

6.1.1. Sections and subsections of the main application window

All global main application window commands are divided into the sections. Navigations between sections is enabled by sections panel. The sections content is unambiguously defined by the content of upper level subsystems, for which the Include in command interface flag is set.
The current section commands are displayed in the navigation panel and actions panel of the main window.
When displaying section commands one or several subsection may appear in the navigation panel. Subsection - a complex of commands that correspond to the subordinate subsystem (also with the Include in command interface flag). For example, in Sales records section may include Retail and Wholesale subsections because there are such subordinate subsystems.

6.1.2. Types of commands

That user command interface can be divided into:

- independent global commands,
- parametizable global commands,

- local form commands.

In this section we will only describe global commands. From one hand, the commands are divided into:

- standard commands (automatically added by 1C:Enterprise to the command interface);
- navigation commands;
- action commands;
- commands created in the configuration.

6.1.2.1. Standard commands

For most configuration objects, 1C:Enterprise provides standard commands, which are being automatically located in the comment interface. Such objects and standard commands provided by 1C:Enterprise are listed below:

- The common form:
  - Open common form;
- Constants:
  - Open constant editing form;
- Catalogs:
  - Open list form;
  - Open new form;
  - Open new group form;
  - Generation;
  - Go to the list with filtering by the owner;
- Documents:
  - Open list form;
  - Open new form;
  - Generation;
- Documents journals
  - Open list form;
- Reports:
  - Open new form;
- Data processors:
  - Open new form;
- Chart of characteristic types:
  - Open list form;
  - Open new form;
  - Open new group form;
  - Generation;
- Charts of accounts:
  - Open list form;
  - Open new form;
  - Generation;
- Charts of calculation types:

    - Open list form;
    - Open new form;
    - Generation;
- Enumerations:
    - Open list form;
- Information registers:
    - Open list form;
    - Open new form;
    - Go to the list with filtering by the recorder;
- Accumulation registers:
    - Open list form;
    - Go to the list with filtering by the recorder;
- Accounting registers:
    - Open list form;
    - Go to the list with filtering by the recorder;
- Calculation registers:
    - Open list form;
    - Go to the list with filtering by the recorder;
- Business processes:
    - Open list form;
    - Open new form;
    - Generation;
    - Go to the list with task filtering by business process;
- Tasks:
    - Open list form;
    - Open new form;
    - Generation;
- Exchange plans:
    - Open list form;
    - Open new form;
    - Generation;
- Filter criteria:
    - Open list form.

6.1.2.1.1. Generating and posting the standard commands

Standard command for opening the list form and creating new item are always generated if the Use standard commands property is not disabled.
The standard report command is generated if the report has the main ER diagram or main/additional form specified.
The standard processing command is generated if the processing has main or additional form specified.

The standard generation commands are created by 1C:Enterprise if the Generation property is properly set. For example, if the Goods catalog and Goods batches catalog are specified as the possible basis for Arrival invoice and Expenditure invoice documents generation, then the standard generation command will appear for these catalogs.
Standard command for opening a form of the list filtered by owner are generated when the `Owners` catalog property is set in particular way, and for information register there is one or several dimensions with `Leading` property.
The standard command of the list form filtered by the recorder is formed for registers subordinate to the recorder.
The standard commands are located in the following way:

Panel	**Commands located**
Navigation panel	List opening commands
Actions Panel	• The command for opening a constant editing form. • Commands for opening new object forms. • Commands for opening report forms and processing forms
Form navigation panel	Commands for opening filtered list forms
Form command bar	Generation commands

6.1.2.1.2. Parametizable standard commands

Some standard commans are parametized, i.e. they can be executed in the context of this or that form after receiving some value as a parameter.

Command	**Parameter type**
Generation	Link to basic object
Opening the list filtered by owners	Link to owner object
Opening the list filtered by the recorder	Link to recorder document
Opening the filter list	Link to filter value

If it is said that Goods and Goods batches catalog can be used for generation of Arrival invoice, then the generation command parameter type will be compound: `CatalogRef.Goods` and `CatalogRef.GoodsBatches`. This is why the Arrival invoice generation standard command will be automatically located in goods and goods batches catalog item forms.

6.1.2.2. Independent and parametizable global commands

Independent global commands are designed for the user to select the functions for the whole application. Running such command doesn't require additional information (parameters). Examples of such commands:

- opening the catalog list,
- opening the document journal,
- opening the report form,
- opening new catalog item form and so on.

Parametizable global commands depend on execution context and cannot be executed without getting additional information (command execution parameter). Examples of such commands:

- Opening list of subordinate catalog (parameter - link to owner catalog item);
- Opening list of records of the register subordinate to another recorder (parameter - link to recorder document);
- Generation of one object based on another (parameter - object that serves as a basis).

Parametizable commands can be displayed in the navigation panel of the auxiliary window and directly in the command bar form.

Global commands are displayed in a form command bar before the Open help commands. The commands from Importantgroup are located directly in the command bar and other command groups (Generate standard command as well as other groups from Form command bar category) are located in the menu section. Execution of global commands located in the form command bar results in opening new auxiliary window of the application.

Remember that global commands don't process `Mark` property. This property is designed to be used by form commands only.

6.1.2.3. Navigation commands and action commands

Navigation commands are designed to forward the user to another application form without leaving the current (main or auxiliary) application window. Navigation command opens another form in the same application window, where the user has called the command.

Both independent commands of a global command interface and parametizable global commands can be navigation commands. Navigation commands are located in the navigation panel of the main or auxiliary application window.
The examples of navigation commands in main application window: commands for jumping to the catalog list or document journal. For example, Currencies command opens currency list form in the same window, but Financial documents command will open the document list form in the same window.
The examples of navigation commands in auxiliary application window: commands for jumping to lists that are logically subordinate to the object edited in this window. For example, in the currency catalog item editing form there can be a command for jumping to information register with exchange rate change history. The document form can include a command for jumping to its moves in this or that register.

NOTE. Discarding for opening doesn't stop the navigation command execution. Discarding form opening will result in opening of an empty form.

The **action** commands are the ones that open new auxiliary application window. Such commands switch the user from executing another command for some time, i.e. they change the activity context significantly. For example, new document creation command switch the user from the task of navigating the application, which is executed in the main application window, to the task of editing new document.
Such commands are located in the actions panel of the main application window or in the command bar of the form displayed in the auxiliary application window.

6.1.2.4. Commands created in the configuration

Apart from standard commands, the developer can create his or her own commands, define their locations (command group), describe in 1C:Enterprise language the actions performed by them and so on.

6.1.3. Command groups

All global commands are divided into four categories by the location and nature.

Category	Description
Navigation panel	For placing independent navigation commands
Form navigation panel	For placing parametizable navigation commands that are called from the form
Actions Panel	For placing commands that result in displaying new auxiliary application window
Form command bar	For placing in the form the parametizable commands that result in displaying new auxiliary application window

The above-mentioned categories are the list of application interface locations, where global commands can be displayed.
1C:Enterprise uses standard command groups to group global commands.

Location	Standard group commands
Navigation panel	• Important, • Normal, • See also
Actions Panel	• Create, • Reports • Service
Form navigation panel	• Important, • Go to, • See also
Form command bar	• Important, • Generate

Moreover, when developing a configuration, the developer can create custom command groups (configuration object is General - Command groups), which can be classified into one of the above-mentioned categories (Category property of a command group). These groups, along with predefined groups, can be used as a location for the developed commands.

6.2. Building global command interface

In this section we review configuration objects and object properties that affect the building of a configuration command interface.

6.2.1. Subsystems

The basis for creating a global command interface of the main application window- configuration subsystems structure. It is the subsystems that form the user's impression of the application's overall functions. The subsystems structure describes the general functions to the user. Thus, building the global command interface of the main application window based on the subsystems structure imposes some responsibility during the subsystem development. In fact, the subsystem structure - the first thing that users see when getting familiar with the application.

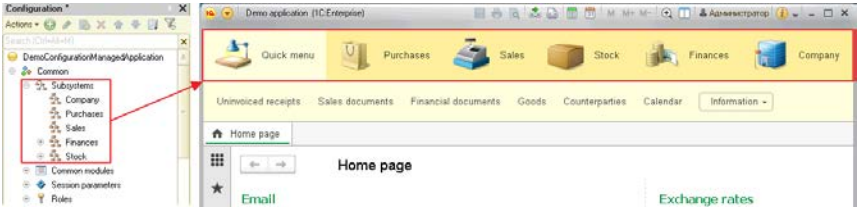


Fig. 131. Displaying subsystems in sections

The command interface structure is impacted by the subsystems that has Include in command interface property set. This property is set for subsystems by default so the subsystems are supposed to describe the global command interface in the first place.

NOTE. If there's no subsystems with the Include in command interface property set, then the sections panel is not displayed in the main application window.

First-level subsystems lead to creating **sections** - a set of specifically oriented global commands. Sections are displayed in sections panel of the main application window. Choosing a section changed command set of the panel and actions panel.

Subsystems of lower levels lead to creating navigation panel **subsections**, which include commands of a respective subsystem.

The object's attachment to the subsystems set defines, which object commands are displayed in the respective sections of a configuration command interface. However, new objects are not attached to any subsystem by default. This means that object commands are not present in the command interface. This is why, to place standard and custom object commands into a respective command interface sections, you must specify the subsystems, to which the object is attached.

The object's attachment to subsystems is independent, i.e. an object can be attached to the parent and subordinate subsystems at the same time. It can also be attached to subordinate subsystem only, if this is reasonable for building a command interface.

NOTE. If a configuration doesn't have any subsystem with the Include into command interface property set, the desktop includes all commands that could be placed there manually. In this case, the desktop command interface editor appearance changes: there's no command tree and there's no option to delete the command from the desktop.

As soon as the new subsystem (included into command interface) appears, the commands are not placed in a home page and they must be placed explicitly.

NOTE. The automated home page placement mode is not enabled for all commands, if the Ordinary application startup mode is set for the configuration.

When building a sections panel, all first-level subsystems are used that meet the requirements below:

1. Include into command interface property is set to `True`;
2. Access rights provide subsystem availability for the current user;
3. The subsystem is not disabled by the functional options;
4. The subsystem is not disabled by the user via settings dialog box of a sections panel.

In such case, if all commands in this section are not available for the user (because of insufficient access rights or if the commands are disabled by the functional options/user settings) such section is still displayed in the sections panel.

6.2.2. Command

As mentioned above, there are standard commands provided by the system automatically and custom commands created by the developer.

`Command` configuration object can be created as a subordinate object for the following configuration objects:

- Catalogs
- Documents
- Document journal,
- Reports
- Data processors
- Charts of characteristic types
- Enumerations,
- Charts of accounts
- Charts of calculation types
- Information registers
- Accumulation registers
- Accounting registers
- Calculation registers
- Business processes
- Tasks

- Exchange plans
- Filter criteria.

Apart from this, you can create common commands (General - Common commands configuration tree branch).

6.2.3. Command parametization

If a command is parametizable, its execution includes getting some value as a parameter. Such value can only root from form data.

IMPORTANT! Setting a command parameter value type is only rational for the commands that are located in forms (they are related to Command bar of the form or Navigation panel of the form category groups).

In fact, Command parameter type property sets the content of the forms, in whose context the command execution is possible. For example, if a command has `CatalogRef.Goods` parameter type, it can be executed:

- in Goods catalog item form, where the parameter is a link to the object editable in the form;
- Goods catalog list form, where the parameter is the link from the current list row;
- in arrival invoice form, where the parameter is the link to the good from the current row of a document tabular section and so on.

The number of values of the selected type that will be sent to the command as the parameter values define the Parameter use mode property. If this property is set to Single, then one value of the specified type is sent to the command.

If the property is set to Multiple, then the values array is transferred as the parameter (even if one value is selected). It's reasonable to use this mode when the data source for the command can be a table with multiple selection. In this case the first array item will be the current row (regardless of the sequence of table row selection). So if we have a list of goods: Mixer, Vacuum cleaner, Fridge, and Kettle, where Mixer, Fridge and Kettle are selected and the current row is Fridge the command will get an array containing the following (sorted by items): Fridge, Mixer, Kettle. The items order after the first one is not defined despite some order in the example.

NOTE. If the current row is not selected, it's impossible to identify the first array item unambiguously.

When placing several managed form tables, associated with one form attribute of `DynamicList` type, on the form it is not specified which table will be used as a source of data for command parametization.

Creating section of a command interface is automated and based on the main attribute form. When editing a form the form developer can add to the command interface section. For example, if a form is designed for editing a good, it will

automatically include all commands with the following parameter type - CatalogRef.Goods (for example, Product profile printing, Jump to price list and so on). If the good has a Manufacturer attribute, then when developing a form commands with CatalogRef.Contractor parameter type (for example, Contractor card, Contractor contract and so on) can be included, too, because you can get a value for parameters of such commands in such forms. Such commands (which are not associated with the main attribute) are not included into the command interface automatically; however, the developer can include them using the form editor.
The catalog group (Folder and item hierarchy type) and hierarchical chart of characteristic types are not included automatically into the command interface of the parametizable global commands group forms, however the developer can include such commands into a command interface using the form editor.

6.2.4. Creating command interface by default

The command location is affected by the Command parameter type and Group properties as well as attachment of the command (or parent object) to a subsystem.
Independent (i.e. not parametizable) commands are included into the global command interface sections that correspond to subsystems, which include owner object or the command itself (for common commands).
Parametizable commands are included into the global command interface sections that correspond to the objects with the types set in Type property of the command parameter.
In both cases, the command group - a peculiar specification of command location within one section. For example, command groups specify the panel (navigation or action), in which independent commands of a sections will be located. For parametizable commands the command group defines, whether the command will be located in the form navigation panel or the form command bar.
Thus, for independent commands (commands that don't require parameter, for example opening a list or opening new object form) an ability to be placed by default in a category other than navigation or action panel is rarely relevant. Placing parametizable commands in the navigation or actions panel is pointless, because in the main application window context such commands can't receive parameters required for their execution. The command placement table is shown below.

	Independent commands	Parametizable commands
Navigation commands	Navigation panel	Form navigation panel

Action commands	Actions panel	Form command bar

Remember that by default the global commands are not located in group form command interface of hierarchical catalogs and charts of characteristic types. To make the global command available in the command interface of such form, the command must be palced manually using the form editor.
When creating the command interface text presentations of configuration objects and various commands are used. By default, 1C:Enterprise generates presentations based on Synonym and Name properties. However, you can impact this presentation. List and object presentations are used for this.

6.2.5. Command interface property

Command interface (for subsystems), command interface (for configurations) and Main section command interface (for configuration) properties are used to edit global commands set of the respective section or command set of the main section. Edit these properties if the default command order, command groups, or command visibility do not meet user requirements.
In different ranges the same command can be displayed in different places on the navigation panel or actions panel and can have different visibility by default. For example, a command for opening Goods catalog list from the Trade section perspective is important and frequently used, and the developer can place it in Important group. However, from the Accounting perspective it's not that important, and developer can place it in See also group.

6.2.6. Editing command set

Apart from editing Command interface property, the editor of this property allows to edit the command set of all sections in All subsystems editor.
Editing the set of global commands, displayed in the form, is done in the form editor on the Command interface tab.

6.2.7. Role-based settings of command visibility by default

When creating section commands and form commands the following default visibility is set.
Visible by default:

- Commands for opening the list of catalog/document/document journal/chart of characteristic types/chart of calculation types/chart of accounts/independent information register/business processes/tasks/exchange plan
- Commands for opening report forms and processing forms
- Commands for opening a constant editing form
- Commands for jumping to the list of subordinate catalog
- Commands for jumping to the list of logically subordinate information register (i.e. a register with the leading dimensions)
- Generation commands
- Arbitrary commands created by the developer in the configuration.

Invisible by default:

- Commands for opening enumeration lists, information registers subordinate to the recorder, accumulation registers, accounting registers, calculation registers
- Commands for opening new object forms and new group forms
- Commands for jumping to subordinate register lists
- Commands for jumping to filters.

The visibility values created by 1C:Enterprise automatically can be changed for both parametizable and not independent commands. The default visibility value can be set broken down by roles used in a configuration. When displaying the commands in 1C:Enterprise mode, the command will be visible by default if the visibility of this command is set for at least one of the user roles.

When editing the role-based visibility remember that the command visibility for the role is only enabled when the command is available for this role. Therefore, when setting up visibility for a role that is highly restricted in rights, there is almost never any need to hide commands in addition, by unchecking their default visibility (besides, the total number of commands available in one section or another is rather small).

Editing of the role visibility by default- is a tool that allows to configure the initial "fullness" of the global command interface, primarily for users with broad access rights.

6.3. Service options for navigation

6.3.1. Using links

There is an option in "1C:Enterprise" to get a text link to any section of the command interface, report, processing, as well as to the objects of the infobase (documents, reference elements, etc.).

Fig. 132. Commands for getting and following a text link

The resulting link can be saved and used later, as a shortcut for easy navigation. So long as the links are textual, they can be sent to other users, for instance, via e-mail. You can not get a link to the tools that are part of the standard functions. There is an option to launch the web client simultaneously as you follow the link. To do this, you need to get an external link from a web client or a thin client connected to the infobase via a web server.

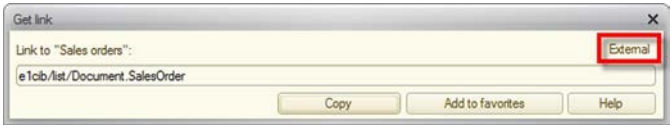


Fig. 133. Getting an external link

To get an external link, use a standard command for getting a link, making sure that the "External" button is pressed in the window for getting a link (see fig. 133). The resulting link can then be pasted in the browser's address bar, this will download an application as you follow the link (in the example in fig. 133, this opens the Stock Balance report form).

NOTE. In Microsoft Internet Explorer, the application does not load when following a link if this link refers to an application is currently loaded in a window of the web browser. It is recommended to paste a link in the address bar of a new blank web browser window.

6.3.2. User Notifications

The notification service is designed to inform users that the system has performed this or that action. In addition to standard notifications, applications can have custom ones. The system generates notifications when an object is interactively recorded or modified, and a developer does it -by calling the `ShowUserNotification()` method. Notifications can have a differing status and allow to implement an option to invoke an action when clicking on them. A notification is displayed in a window, with its location depending on the notification status, on the client application in use, and on the operating system settings. When a notification is programmed to support an on-click action, the notification will also become a hyperlink, while a click of the left mouse button within the notification area will invoke an action specified in design-time. If a notification is generated by the system once an object is interactively recorded or modified, then a click of the left mouse button on the notification area will open this object.
Notifications come in two statuses: important and informational. They differ in both appearance and behavior. The status of notification is set by the `UserNotificationStatus` parameter of the `ShowUserNotification()` method.

Fig. 134. The "Informational" Notification Type

Notification with `Information` status incorporates the following features:

- Such notification can be moved across the screen.
- There's an option to change the notification size.
- The system remembers position and size of notification window.
- Notification is displayed with no fading effect for 1 second and disappears completely from the screen in 5 seconds.
- In the web client, notification can be displayed only within the main application window and only if the application is active.
- Missed notifications can not be "restored" (the fact of them being displayed is not saved anywhere).

Notification with `Important` status incorporates the following features:

- Such notification has the fixed size and position. Sizes and positions of notification windows cannot be changed.
- Notifications are displayed even if the main application window is minimized.
- The notification is displayed for 10 seconds, and for -4 seconds when using the Microsoft Edge web browser. After that, the notification window disappears with no fading effect.
- Message Center is intended to handle notifications. F an notification of the `"Important"` type has been skipped by the user,- then the fact of displaying such an alert will be saved in the message center.
- The number of notifications displayed at a time depends on the client application and the operating system in use:
  - thin client, fat client, and Microsoft Internet Explorer- web browser permit up to 3 notifications at once;
  - Google Chrome- web browser permits up to 3 notifications at once;
  - Mozilla Firefox- web browser permits unlimited number of notifications at once;
  - Microsoft Edge- web browser displays only the latest notification;
  - Safari- web browser shows the latest notification when the Banner type is set up in the notification settings or an unlimited number of notifications, when the Warning type is selected in the notification settings;
  - if a user has disabled notifications in their web browser for the website with an application, notifications are displayed within the main application window. In this case, no more than 3 alerts are displayed simultaneously.

In order for the left mouse button to trigger an action when clicking on a notification, the `ActionOnClick` parameter must be specified when calling the `ShowUserNotification()` method. Values of two different types can be assigned to this parameter. The system behaviour will vary depending on the value type:

- The `String`- type in this case, it is expected that the parameter should contain a navigation link to the infobase object.
- The `NotifyDescription` object -in this case, a handler will be called with an action (i.e., with a left mouse button click) which is a link that is specified at the `NotifyDescription` object instantiation stage. The developer themselves can implement a desired response to the action intended for such notification.

Depending on the notification status, an action is invoked when clicking on different parts of notification:

- The `Important` Status: to invoke an action, simply click on any part of such notification.
- The `Information` Status: to invoke an action, click on the displayed hyperlink only.

An option is provided to update previously displayed notifications. For instance, a notification displays the number of tasks that the user needs to coordinate. In this case, it is practicable to update the information of an existing notification rather than to instantiate a new notification for each new task. To implement this behavior, the `UniquenessKey` parameter is to be used. If at the time of calling the `ShowUserNotification()` method there exists a notification with its unique key equal to the corresponding parameter of the method, then the existing notification will be closed and a new notification will be generated with an updated information and the selfsame unique key. In such event, a new notification may not be displayed on screen since a limited number of notifications can be shown at a time, so instead, a new notification may be placed in the notification queue pending to be displayed. If there are no notification with this unique key,- a new notification will be created and displayed. System notifications have the `Information` status, and their unique key value contains a navigation link of the object edited by the form. The `UniquenessKey` parameter is only used when working with the `Taxi` interface.
Text formatting is not available for the text displayed in notifications.

6.3.3. Displaying the status of time-consuming processes

In the process of configuration development, there are situations when it is necessary to inform the system user about the progress of time-consuming processes (such as a payroll calculation for a business unit). That's when the status bar comes in handy.

The status bar is displayed by calling the `State()` ("Состояние") method and is not available on the back-end. When it is necessary to display on the front-end the progress of a time-consuming process running on the back-end, such process should be implemented in such a way that it could be executed on the back-end in "chunks", for their further initiation from the front-end. In such fashion, it becomes possible to display the change of a process execution simultaneously with the start of the next "chunk" on the back-end. The status bar is displayed in a window that is located by default in the lower right corner of the working area of the screen.

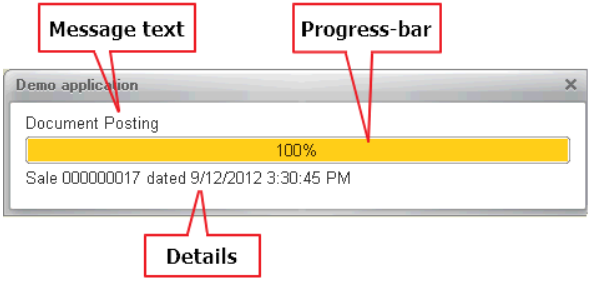


Fig. 135. The status display window

If it is desirable to display the progress of the process execution by using a progress bar indicator (for example, we know the total number of employees and wish to display how many of them have already been calculated in the payroll), then we must specify the second parameter of the method that sets an absolute value of the progress indicator when calling the `State` method(). The minimum value of the progress indicator is always set to 0, and the maximum value is -100.

```
State ("Documents are being processed (" +
PeriodPresentation(StartDate, EndDate) + ")", ForProgress, "Documents
are being processed for" + ProcessingDate + ". Total: "+ Counter,
PictureLib.Process);
```

If the second parameter is not specified when the method is called, the progress indicator will not be displayed on the status bar. In this case, an explanation note will be placed directly below the text of the main description:

State (“Processing is done, ,"Total processed documents: " + NumberOfProcessedDocuments + Symbols.LF + "ProcessedDocuments: "+ NumberOfProcessedDocuments, PictureLib.Processed);

Fig. 136. State Without Progress

The status panel automatically disappears from the screen shortly after the execution of the program routine that caused the appearance of the status panel is over. Nonetheless, if the mouse cursor is being hovered over the status panel window, it will be displayed until it is forcibly closed or the mouse cursor is moved from the status bar area.

NOTE. The status bar is located on top of all windows of the current application, except for modal windows. A modal window is displayed on top of the status bar.

In the web client, the status bar is implemented as a separate web browser window. The way it is refreshed depends on a particular web browser:

- Microsoft Internet Explorer. The status is updated each time the `State()` ("Состояние") method is called.
- Mozilla Firefox. The window is refreshed on each server call, as well as following the execution of the 1C:Enterprise script code. Therefore, in the absence of the server method calls during the execution of the 1C:Enterprise script code the status bar will not be refreshed.
- Google Chrome and Safari. The window state is refreshed only following the execution of the 1C:Enterprise script code.

6.3.4. Messages

In the application, most messages are naturally related to the data. For instance, in case when the stock is lacking certain amount of a certain item while the document is being processed, the developer should notify the user about it.
The message tool allows developer to create an error message that enables to specify which of the object's properties have cast the error. When displayed in a client application, a message can be automatically bound to a form element that edits this property, and the specified message will be displayed next to it.
Messages are displayed in the message panel that is located in the lower part of the work area. The height of the message panel does not exceed 5 lines. If the panel contains more messages it could display at once,- a scroll bar appears, but the panel itself does not increase in height. The height of the message panel can't be changed manually, either.

To enable this tool's functioning across the platform, there is a `UserMessage` object.

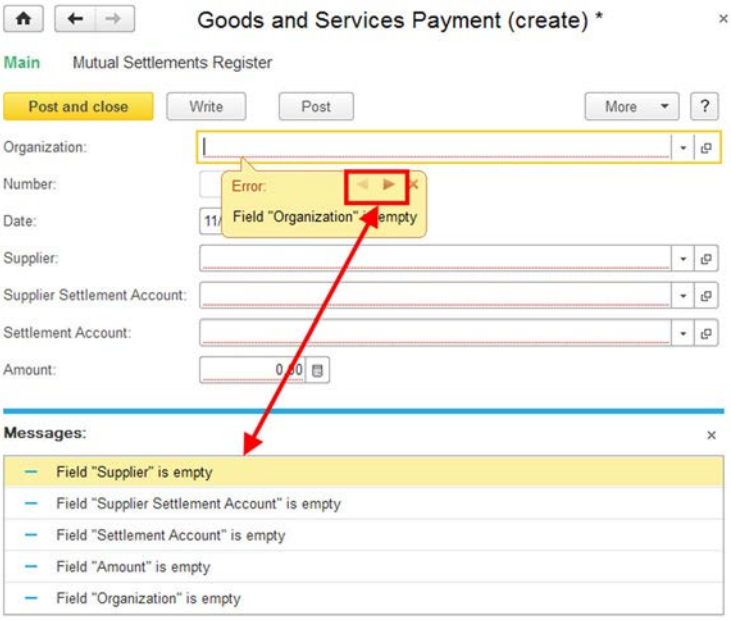


Fig. 137. Messages

The message tool is aimed to inform the user that the requested action has failed. For instance, the processing of a document is not possible due to the shortfall of certain items in the stock.

If a message is linked to a field of a tabular section, then the reordering of the tabular section rows should not affect the correct way of displaying the message.

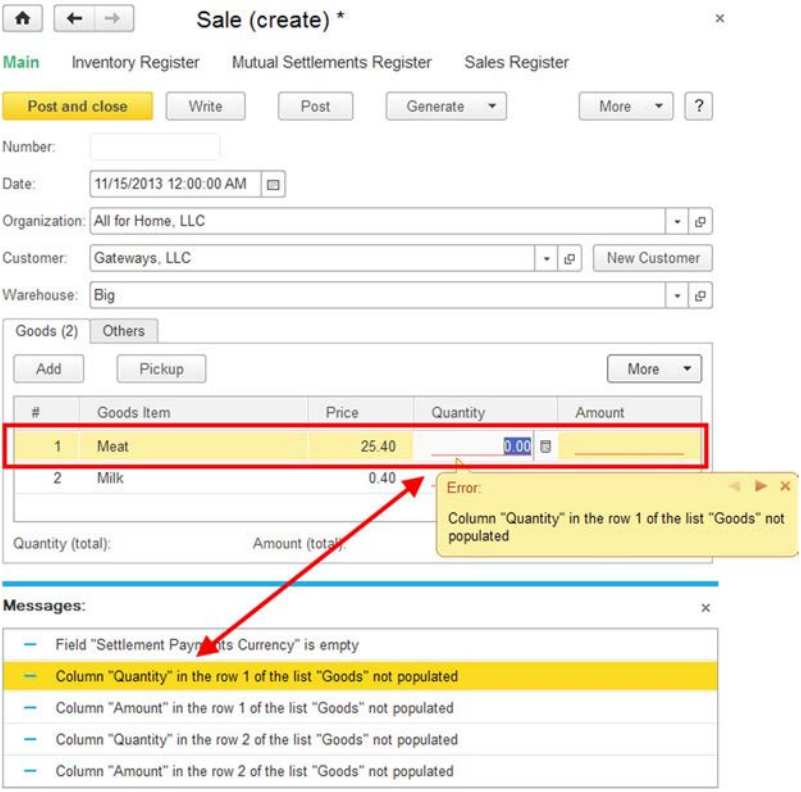


Fig. 138. Messages in a tabular section

If a row to which a message is linked has been deleted, the message will be re-linked to the form element that displays the tabular section.

In case when a message is linked to a table row associated with a form property such as `FormDataCollection`, `FormDataTree`, or `FormDataStructureAndCollection`, and a user has interactively performed a search in this table (by using an appropriate standard command), and due to the search result the string linked to the message is found hidden, then the search will be automatically canceled, with the message displayed in correct

place. In case when a program filter is set for the table (the `RowFilter` property), then such filtering is not canceled when the message is displayed, with the message being positioned on the table as a whole.

Messages generated by the global context `Report()` method or by using the `UserMessage` object are recommended to be used only to inform about erroneous actions. It is also recommended to supply the output of such messages by disabling the closing function for the form in which messages are being displayed, if messages are generated during the execution of an action that implicates closing the form. If the form closing function is not disabled, there may occur situations when the messages are displayed in the message panel of the form that is being closed, and therefore, this panel will be closed along with this form. In this case, a user will not see the displayed messages. If it is necessary to inform a user about an event, then it is recommended to use the `ShowUserNotification()` method.

In order for an error message to be linked to the desired form element, the system needs to know the following information: what property of which infobase object is to be displayed, and by what form element. Filling in the above properties allows you to answer all these questions:

- By setting the value of the `DataPath` property, we indicate which of the form properties store the data of the desired infobase object.
- By setting the `Field` property, we specify the object property which the message will be displayed for. It can be said that a message will unfold the `DataPath.Field` data identified in the database by the `DataKey` value. Next, the system will automatically (drawing on the knowledge of the form structure) determine the form element in which the required information is displayed. And finally, a message linked to this form element will be displayed.

  The `Field` property may contain a data description in the following format:
  - For properties, -a string formatted as`AttributeName`, for example, `Supplier, Supplier.Parent`.
  - For the properties of tabular sections - a string formatted as `TabularSectionName[ИндексСтроки].PropertyName`, for example, `StockList[10].Quantity`.
  - For the record sets' lines - a string formatted as `[ИндексСтроки].AttributeName`, for example, `[10].Currency`.
  - It should be remembered that if a field to which the message is linked will be moved by a user from a table to a form (for tabular sections and record sets), the message will no longer remain linked to this field.
- The `TargetID` property describes which form the message should be linked to. For instance, if a message is generated from a form displayed on the home page, specifying the `TargetID` property will let display the

message linked to that particular form from which the message was generated.

If the DataKey property is filled in the message and it does not match the key value of the current form's main property (be it an object reference or information register record key), then a new object form corresponding to the DataKey property value is being opened.

- If the DataPath property of a message is empty, the name of the main property of the open object form is assigned to it, and the message is displayed in the newly opened form.

Besides, all messages with the DataKey property equal to the value of this property in the current message are being transferred from the current form window to the open form. At the same time, the DataPath property for all transferred messages is assigned in a similar way, if it is not set.

To fill the data path, a special system tool is being used that saves information regarding the mapping of the form elements and objects, and then uses it when the SetData() method is called. Thus, before executing the procedures of the object module, in which messages are formed, the match between the form properties and objects must be established, and in the module itself, developer must call the SetData() method.

When performing standard actions with an object in a form, for example, when posting a document, the extension of the form itself creates an object and finds a match. Therefore, in these cases developer does not have to do anything.

NOTE. If at the point of writing a new object (exchange plan, catalog, document, element of chart of accounts, element of a chart of calculation types, element of a chart of characteristic types, business process, task) messages have been formed to the user in a transaction, then once the transaction is canceled, the links to this object are cleared in the messages (DataKey property).

The following describes how to set a mapping match if an object is being created from the form data programmatically.

The match between an object and a form can be specified explicitly, by calling a method, or implicitly. To establish the match explicitly, use the SetObjectAndFormConformity() method. For the second argument, the method receives a pointer to the form in which an object is displayed (the FormIdentifier property) and to which property of the specified form the data object is stored (the Properties property), which enables the subsequent setting of the correct mapping of the messages and form controls:

```
Parameters = New Structure;
Parameters.Insert ("FormIdentifier", UUID);
Parameters.Insert ("Property", "Object");
SetObjectAndFormConformity(Object, Parameters);
```

The established match will exist so long as the object that the match is set for exists.

In order for the message to use the object and the property name mapping, you should associate the message with the data with the `SetData()` method of the `UserMessage` object.

```
CatalogObject = CatalogLink.GetObject();
Parameters = New Structure;
Parameters.Insert ("FormIdentifier", FormIdentifier);
Parameters.Insert ("Property", "Object");
SetObjectAndFormConformity(CatalogObject, Parameters);
CatalogObject.CheckFilling
```

It should be realized that in the `SetObjectAndFormConformity()` method the property name is specified for the form that will be opened by default to display the element of the catalog in use (in the example, `CatalogObject`). Then the messages that are created in the filling check handler (`CatalogObject.CheckFilling()`) will be correctly linked to the properties of the new form. If `FormID` of the `Parameters` structure (in the example above) contains a unique identifier that does not match to any instance of the form, then a new form will be created and opened to display a message.
To get a match, the `GetObjectAndFormConformity()` method can be used. You can get a match so long as there is an object for which it is set.

```
MatchDescription = GetObjectAndFormConformity(Object)
```

If there is a match with a property for the passed object, the function will return the name of the property. Besides, the match can be ascertained through the form by using the `FormAttributeToValue()` method. In most cases, it is recommended to use this method.

```
// Code in a document module.
&OnServer
Posting() Procedure
  ...
  Message = New UserMessage();
  Message.Text = "In line 10 of the tabular section" + "The number of
lacking product item(s) is " + MissingNumber + "" + Unit of
Measurement of the Product;
  Message.Field = "Products[9].Quantity";
  Message.SetData(ThisObject);
  Message.Message();

  // The message will be displayed in the form and linked to
  // a control mapped to the field
  // Quantity in the line 10 of the tabular section
  // Products.
  ...
EndProcedure;
```

If the generation of user messages to a user (by using the `UserMessage` object) is performed during a context or non-context server call from a form, or during a call from a common module or from a common command module, the output

of messages is blocked. The generated messages will be displayed to user after returning control to the frontend. To get a list of not displayed messages, use the `GetUserMessages()` method.
If there is an invalid value in the input field which the message is linked with, you can switch to the message window in the web client. In other frontend applications, such a switch cannot be performed.

6.3.5. Working with keyboard shortcuts

In an independent auxiliary window, the following shortcut keys are triggered:

- shortcut keys for the commands of the main window navigation panel;
- key shortcuts for the commands of the main window action console;
- key shortcuts for commands of the auxiliary window itself (both form commands and navigation panel commands).

Simultaneously, the key combinations of the main window commands are triggered, even if the command called by any key combination is invisible, for example, was hidden by user. In the auxiliary windows, the shortcut keys of the form navigation bar hidden commands, as well as those of the hidden form elements, do not function.
In the blocking windows, only the keyboard shortcut of these windows work, and the shortcut keys of the main window commands are not supported.

6.3.6. Working with the List of Forms

The system allows you to get a list of forms that are currently displayed in the system, including forms located on the home page (desktop). This list can be retrieved by using the `GetWindows()` global context function. Then the contents of the resulting `ClientApplicationWindows`collection (that consists of objects of the `ClientApplicationWindow` type) should be analyzed:

- The window with the `Main` property set as true describes the main application window. The `Content` property for such window always contains an empty array.
- A window with the `HomePage`property set to true describes the home page (desktop). In this case, the `Content` property contains a list of forms that are currently placed on the home page (desktop).
- The remaining windows describe the auxiliary windows of a client application. The `Content` property for such a window contains a list of forms opened in this window. The main form of such a window will be the first item of the array.

It should be remembered that the order of the items in the `ClientApplicationWindows` collection is not defined. In applications, it is not recommended to rely upon any assumptions regarding the order of this collection items.
The following example displays all windows of the client application, outputs their properties and a list of forms that are located in a particular window of the client application (if any).

An example of displaying the windows list:

```
Windows = GetWindows();
For Each Window Of Window Do
  Report("Header: " + Window.Header);
  Report("Main Application Window: " + Window.Main);
  Report("Home Page: " + Window.HomePage);
  Report("Number of forms in the window: " + Window.Content.Number()
);
  If Window.Content.Number() Then
    For Each Form of Window.Content Do
      Report(" Form : " + Form.FormName);
    EndDo;
  EndIf;
  Report("-----");
EndDo
```

6.3.7. Using the Jump List

The operating system Microsoft Windows 7 and later (OS version 6.1 and later), as well as the Linux operating system with the Unity desktop shell, allow users to use the so-called **jump lists**. This feature allows the user to quickly navigate to the objects of the program being used (files, folders, etc.) by right-clicking on the command panel button associated with this application. Some
objects (at the user's choice) can be fixed in this menu with the help of a special button.

A detailed description of the jump list functioning can be found in the manuals to the corresponding operating system.
The 1C:Enterprise system also provides a jump list. Infobase acts as an object in this list. Jump list is synchronized with the list of recently selected infobases of the launcher window, even if the display of this list in the launcher is disabled

When using jump lists, keep in mind the following peculiarities of the system behavior:

- The launch of the infobase from the jump list always takes place in 1C:Enterprise mode.

- To display the jump list, docking should be performed by using the Dock to the Taskbar command , for the 1C Enterprise desktop shortcut created by the 1C:Enterprise installation program.

6.3.8. Installing preview window buttons for the Windows taskbar

In the Microsoft Windows 7 operating system and later OS versions, Aero Peak function is available to users. This function enables to preview the thumbnails of all windows associated with the selected program by hovering the mouse over the corresponding taskbar button. Some programs place special buttons on these thumbnails that facilitate program management. For instance, Windows Media Player places buttons for navigating between tracks and stopping / starting play-back in the thumbnail window.

The 1C:Enterprise system provides an option to place similar buttons on thumbnails of running infobases. To do this, use the special global context `OSTaskbar` property. Let us now look at the example of using this property.

As an example, add two buttons to the preview window:

1. Place the Customer Order;
2. Checkout New Sale.

We will create buttons at the beginning of the system start-up:

```
Procedure BeforeStart(Failure)

  Buttons = New Array();
  Button1 = New Structure();
  Button1.Insert("Image", PictureLib.ClientOrder);
  Button1.Insert("Title", "Place the customer order");
  Notification = New NotifyDescription("ButtonActions",
ButtonNotifications, "Place the order");
  Button1.Insert("Action", Notification);
  Button2 = New Structure();
  Button2.Insert("Image", PictureLib.NewSale);
  Button2.Insert("Title", "Checkout New Sale");
  Notification = New NotifyDescription("ButtonActions",
ButtonNotifications, "Checkout New Sale");
  Button2.Insert("Action", Notification);
  Buttons.Add(Button1);
  Buttons.Add(Button2);
  OSTaskbar.SetButtons(Buttons);

EndProcedure
```

When clicking on any of the created buttons, the handler will be triggered, in which the desired action will be performed. However, a managed application

module cannot contain notification handlers; therefore, you should create (or use) a front-end common module that will host the handler. The example uses the common module `ButtonNotifications` and the `ButtonActions()` handler.

```
Procedure ButtonActions(AdditionalParameters) Export

  If AdditionalParameters = "PlaceOrder" Then
    // an order document to be created here
  ElseIf AdditionalParameters = "CheckoutSale" Then
    // a sale document to be created here
  EndIf;

EndProcedure
```

Now, to create an order and checkout a sale there is no need to maximize the main application window and search for the necessary commands in the command interface.
When using this tool, the following features and limitations should be kept in mind:

- The maximum number of buttons that can be placed in the preview window is -7.
- after the first buttons are created, you cannot increase the number of buttons in the preview window. It is possible to change and hide previously created buttons.
- Any button picture is allowed, but it is recommended to use the `.ico` format that contains pictures of 16x16, 20x20 and 24x24 sizes (for 96, 120 and 144 DPI respectively).
- in the structure that describes each button all three properties are required. A property cannot be omitted.

When working under the Linux OS, as well as on the Windows XP OS, creating buttons in the thumbnail window does not produce any visible result.

6.3.9. Taskbar Icon Overlay Management

With "1C:Enterprise" running, there can occur situations when something happens in the system, and you wish to attract user's attention to it. For example, the built-in mail system has received another batch of e-mail, and hense, the counter of unread letters has increased, or some time-consuming action has completed, and the user's attention is required.
In the event that the system is running Windows 7 (or a later version of OS Windows), the application developer can implement the following options:

1. to place an overlay on the infobase icon displayed on the command bar. As such overlay, a picture or a number can be used.
2. make the command bar button associated with the infobase flash.

Let us consider some examples in detail.

Placing an Overlay Icon on Taskbar Button

Suppose that within the framework of the information system there is a special service for a task exchange, and it is required to display to user how many tasks there are for them running or pending in the system at the moment. To implement this, let us use an idle handler (timer) that will check every 60 seconds (for example) how many tasks are assigned to a user and display this number on the button representing the infobase on the command bar.

```
Procedure BeforeStart(Failure)
  AttachIdleHandler("WaitingProcedure", 60, False);
EndProcedure

Procedure WaitingProcedure() Export
  Counter = Messages.GetNumberOfTasks();
  If Counter = 0 Then
    OSTaskbar.RemoveBadge();
  Else
    OSTaskbar.SetBadge(Counter, False);
  EndIf;
EndProcedure
```

If there are no tasks assigned to a user, -the previously applied overlay number will be removed from the icon. If the second parameter of the `SetBadge()` method is set to `True`, the system will delete the overlay when the corresponding application is selected.

Task Bar Button Flashing

Displaying an overlay icon on the taskbar buttons is a good practice when an application processes some countable information. In this case, the display of a plain number could be sufficient. However, in those cases when the system completes a time-consuming process, such change cannot be indicated by placing an overlay icon on the taskbar button. Nonetheless, it is possible to signal this change (the end of process) by flashing the taskbar button.

To control such alert signals, use the `OSTaskbar.Flash()` method. The system behavior upon calling this method depends on the operating system in use.

Let us scrutinize an example in which the timer is set to an interval of 120 seconds, in order to simulate a time-consuming process. At the end of this time span, the taskbar button will flash 5 times.

```
Procedure BeforeStart(Failure)
  AttachIdleHandler("WaitingProcedure", 120, True);
EndProcedure

Procedure WaitingProcedure() Export
  OSTaskbar.Flash(5)
EndProcedure
```

120 seconds following the launch of the client application, the taskbar button associated with the application will flash 5 times in succession.

The icons on the Windows taskbar buttons are displayed only when the "Use small icons" checkbox in the taskbar settings is unchecked. On Unity user interface in Ubuntu, there may be no visual effects if the “1C:Enterprise” system is running under root privileges.

6.4. The Command Interface Development Order

When developing a command interface, the following sequence of actions is usually performed:

- at the beginning of the configuration design, its structure is determined regarding the application environment, that is, a tree of configuration subsystems is formed;
- the role list is compiled, whereby it is determined whom this configuration is earmarked for;
- when creating new configuration objects (catalogs, documents, etc.), as a rule, they immediately come to belong to certain subsystems, with permissions being assigned to the objects by roles;
- if a standard default arrangement of commands and their visibility appear unsatisfactory, the Command Interface property of the subsystems in question, as well as the composition of the commands of those forms in which the parameterized commands for the required objects are automatically placed (navigation to the subordinate list and object generation) are to be modified.

This minimum of actions alone will result in the emergence of the commands for opening forms of object lists, as well as the commands for generating new objects (if their visibility in the Command Interface Portion Editor is enabled), in the corresponding sections of the Command Interface. Besides, if it is defined so in the object properties, the forms of other configuration objects will be populated with the object-generating commands, commands of navigation to the list by recorder, commands of navigation to the list by owner, etc.

Let us consider an example of creating a Product Salesdocument. When creating it, we will do the following:

- We shall indicate that the document belongs to the Sales Records and Accounting subsystems;
- next, we shall attribute the rights to work with this document to the roles of the Sales Manager and Administrator;
- further on, we shall indicate that the document is generated on the basis of the Account and Customer's Order documents ;
- and finally, we shall indicate that it is the recorder for the Stock Balance at Depot register.

These actions will make the Product Sales command appear in the navigation bar of the Sales Records section if a user has the role of Sales Manager or administrative rights.

Besides, the Generate submenu command for generating the expenditure invoice will appear in the document forms Account and Customer's Order. We did not have to edit the interfaces for users with such roles and the forms of the Account and Customer's Order documents.

The standard representation of the command Navigate to the Invoice List (Product Sales) may look inaccurate. To override it, fill in the List View property of the `Product Sales` document and specify Sales there.

Suppose that we are not satisfied with the default presentation of an object (which participates in the presentation of commands), and we want to modify it so that the presentation of the Generate command would read as follows: Sales: generate. To do this, fill in the Presentation property of the Product Sales document object with the value Sale.

Now let's pay attention to the fact that with the standard positioning of the command, it is placed in the middle of the list of commands in the Sales Records section. Suppose the command is important enough for this section. We navigate to the All Subsystems editor (or to the editor of the command interface of the Sales Records subsystem) and move the command to the Important group. Meanwhile, we leave the Accounting section as it is, with the command being placed there by default.

Finally, let us pay attention to the visibility of the commands by default. In the form of the Product Sales document, a command for navigation to the list of the Stock Balance at Depot register's entries has appeared, yet it is invisible by default. Let us assume that this command is important for the system administrator, and by default it is really not too necessary for a manager. In order to reflect this, we shall create a form for the Product Sales document, proceed to editing the command interface and set the visibility of the command for navigation to the register's list for the Administrator role, and leave its visibility for the manager unchanged.

For the system administrator with their broad access rights, the number of commands appeared too large. Suppose further that they will but rarely use the command for navigation to the list of sales invoices. In order to make their command

interface more user-friendly, we once again proceed to editing commands in the All Subsystems editor, and uncheck the command's visibility for the role of Administrator.

It is also expedient to set the default visibility for the command for adding a Product Sales document in the actions panel for the sales manager so that they could trigger the generation of this document without addressing the list of documents.

In the course of setting up the configuration, a user can customize the visibility of commands at their own discretion.

Chapter 7. Forms

7.1. General information

A **form**- is an object created to input and view information, as well as to manage various processes. By using forms, an application asks the user to input a piece of information that is necessary for it to execute its function, or provides information to the user for viewing and editing.
The main purpose of the form- is to provide a user with a convenient means for entering and viewing information. Like a paper document, the form allows you to quickly enter the required information and save it for further processing, and if necessary, go back to the previously input data for viewing or correction.
The displayed part of the form (visible to users) is described as a tree that contains **form controls**. The form controls can be input fields, checkboxes, radio buttons, press buttons, and so on. Besides, a control can be a group that contains other controls. A group can be represented as a framed panel, a panel with pages (tabs), a page itself, or a command panel. Moreover, a control can be a table that also contains controls (columns). The structure of controls describes how the form will look. Whereas all the functions of a form are described by dint of properties and commands. **Properties**- are the data that a form operates on, and **commands**- are the actions it performs. Thus, a developer, by using the form editor, should include the desired properties and commands in the form, create the controls that display them and, if necessary, arrange the controls into groups.
The system can automatically create the form of an applied object, but developers are able to create a form themselves and to determine the composition of its properties, commands, and displayed controls. Drawing on this logical description, the system automatically shapes an appearance of a form, to display it for users. That said, the system takes into account various properties of the displayed data (for example, their type), in order to arrange the form controls as user-friendly as possible. Developer can fine-tune the controls' placement by means of various settings. They are able to define the order of controls, set their desired width and height. However, this is just some extra information that helps the system display a form.
In forms, a developer can use not only the commands of the form itself, but also the global commands used in the command interface of the entire configuration. Besides, an option to create configurable commands that will open other forms based on the specific data of the current form is implemented. For example, it may be a call of a stock balance report for a particular warehouse that is at the moment selected in the sales invoice form.

There is a feature in forms for attaching user messages to the form data. This allows the system to visually mark and activate form controls in which the user made a mistake when filling them in.
Also, the form automatically takes into account the role availability of data. For example, if a particular attribute of the displayed object is not available for viewing to a specific user, the system will automatically remove the form control associated with this attribute and then rebuild the form.
The form is designed in the form editor that is available in the Designer mode. In this editor, the following features can be defined:

- **Form Attributes** -are intended for storing data that a form works with.
- **Form Settings** - are intended for arranging relations between the forms and controlling the functionality of a form when opening it.
- **Form Commands** -are intended for performing various actions within a form.
- **Form Module** -contains the program code associated with the form functioning.
- **Form Controls** - are intended for displaying and editing form attributes, as well as for displaying and executing commands.
- **Command Inteface** - contains commands that can be executed in a form and are provided with both the global command interface and the form commands.

IMPORTANT! Forms are present both on the back-end and the front-end at once. This should always be remembered when designing a form.

7.2. Form Attributes

7.2.1. General information

A set of form attributes describes the composition of the data that is displayed, edited, and stored in a form. All the same, the form attributes themselves do not provide options to display and edit data. Rather, the form controls associated with the corresponding form attributes are intended for displaying and editing the data. The set of all form attributes will be called **form data**.

IMPORTANT! It should be remembered that all form data must be described in the form of form attributes. It is not allowed to use the form module variables as data sources for the form controls.

In the process of the form development, you can explicitly set an option to view and edit specific form attributes, in the context of roles, by using the View and Edit properties. Besides, the availability of a certain attribute in a form itself can be configured by using the functional options.

There is an option to assign the form main attribute, that is an attribute that defines the standard form functionality (the form extension).

IMPORTANT! It should be remembered that a form can have only one main attribute.

The form **extension** -comprises the additional properties, methods and form parameters of the `ClientApplicationForm` object that are characteristic of the main form attribute type.

7.2.2. The Form Data Types

The form operates with a limited set of data types:

- Types that are directly used in a form -are those types that are supported on both the thin client and web-client (for example, `Number`, `CatalogRef.Items`, `GraphicalSchema`, `SpreadsheetDocument`).
- Types to be converted to special data types, -form data types. Such types are displayed in the list of form attributes in parentheses, for example, (CatalogObject.Items).
- Dynamic list- is a special data type that allows to organize the display of arbitrary information retieved from database tables in the form. To do this, you should specify the displayed table or describe the resulting selection by means of the query language. The tool is based on a data composition system and provides a set of options for sorting, selecting, searching, grouping, and conditional formatting of the received data.

Some applied types (such as `CatalogObject`, etc.) are not present on either the thin client or the web client end. Therefore, for presentation of such applied types in the form, some special data types designed to work in forms have been introduced in the platform. This peculiarity necessitates the conversion of applied objects into form data (and vice versa).

The following data types are used:

- `FormDataStructure` - contains a set of attributes of arbitrary type. Properties can be other structures, collections, or structures with collections. For instance, the `CatalogObject` in form is represented by such a type.
- `FormDataCollection`- is a list of typed values that resembles an array. Access to a collection item is done by an index or by an ID. In collections generated on the basis of registry record sets or tabular sections of objects, the collection item field `LineNumber` does not correspond to the real index of the collection item. In some cases, there may be no access by ID, owing to the type of an applied object that is represented by this collection. Any integer can act as an ID. A tabular section in form is being presented by such type.

- `FormDataStructureAndCollection` - is an object that is represented in the form of a structure and collection at once. It can be handled like either of these entities. For one, a record set is represented in form by such type.
- `FormDataTree`- is an object intended for storing hierarchically arranged data.

IMPORTANT! It is not recommended to use form data as parameters of procedures and functions that transfer control from front-end to the back-end, and also as return values of functions if it transfers control to the front-end.
IMPORTANT! When calling back-end procedures and functions and returning control to a client, only modifications in the form data are transferred. When the Arbitrary type is selected as the form attribute type and the mutable value is selected as the attribute value, then in order to transfer the data between the front-end and the back-end (and back) correctly, it is necessary to assign it again to the form attribute after the mutable value was changed.

An applied object is represented by either one or several form data items. On the whole, the hierarchy and composition of the form data depends on the complexity and interrelation of the form's applied objects.
For example, a document containing a tabular section will be represented by an object of the `FormDataStructure` type (the document itself) to which the object of the `FormDataCollection` type (the tabular section of the document) is subordinate.

IMPORTANT! During the configuration design, it is essential to remember that applied objects are only available on the back-end, while the form data objects can be used both on the front-end and on the back-end.

As you form the form attributes, it is necessary to consider a series of restrictions:

- Only those types of the 1C:Enterprise system can be stored in the form data, which the XDTO serialization option is indicated for.
- It is not permited to assign the values of the `Array` and `Map` types to the form attributes.
- For the attributes of the Arbitrary type of objects, `such as` `FormDataStructure`, `FormDataCollectionItem`, `FormDataTreeItem`, `FormDataStructureAndCollection`, the assignment of values of `Array` and `Map types is not allowed`.
- It is not recommended to use the values of the `Array` and `Map` types as items of the collections of the `Structure` or the `ValueList` types. In the above cases it is recommended to use fixed collections: `FixedArray`, `FixedMap`.
- It is not recommended to use the `Structure` and `ValueList` types in the form attribute data (i.e., in the attributes of the attributes).

- If a form attribute is of the `Arbitrary` type or a collection (array, map, etc.) is used as a value of such attribute, then such attributes (or collection properties) should not be assigned values containing the following applied objects:
  - `DocumentObject`
  - `CatalogObject`
  - `BusinessProcessObject`
  - `TaskObject`
  - `ChartOfCharacteristicTypesObject`
  - `ChartOfCalculationTypesObject`
  - `ExchangePlanObject`
  - `ChartOfAccountsObject`
  - `ExternalDataSourceCubeDimensionTableObject`,
  - `ExternalDataSourceTableObject`.

In fact, it can be said that form data- is a unified presentation of data from various applied objects on which the form operates in a uniform manner, and which are present both on the back-end and on the front-end. That is, a form contains some kind of projection of these applied objects in the form of its own data types and performs conversion between them if needs be. However, if a configuration developer implements their own data processing algorithm, then they must perform the data conversion (from specific types to applied types and back) independently.

When editing form attributes, it is possible to influence the transfer of data between the front-end and the back-end while the form is running. That's what the Attributes Editor's column Use Always is designed for. Action of this property differs for the three types of attributes:

- For an attribute subordinate to the dynamic list (dynamic list column):
  - The property is enabled- an attribute is always read from the database and included in the form data.
  - The property is disabled- an attribute is read from the database and is included in the form data only when there is a currently visible form control associated with the attribute or its child attribute.
- For an attribute subordinate to a register records collection:
  - The property is enabled -the document register records are read from the database and will be present in the form data.
  - The property is disabled -the document register records will neither be read from the database, nor will they get in the form data (if there is no form control that refers to the document register records).
- For the rest of form attributes:
  - The property is enabled- an attribute will be present in the form data, regardless of whether or not there is a form control associated with the attribute or its child attribute.

- The property is disabled -an attribute will be present in the form data only if there is a form control associated with the attribute or its child attribute. Unlike the attributes of a dynamic list, the visibility of the control associated with the attribute does not matter in this case.

NOTE 1. It should be remembered that a property set for the parent attribute affects all its child attributes. For example, if the Use Always property is always left unchecked for the tabular section of a document, then the system assumes that this property is likewise unchecked from all its child attributes (despite the actual state of the property).
NOTE 2. For form attributes of an object type (`CatalogObject`, `DocumentObject`, etc.), the `URL`, `DeletionMark`, and `IsFolder` attributes are always present in the form data (if the object has a corresponding attribute), regardless of the Use Always property state.
NOTE 3. It is not allowed to assign the values of the other form data objects to the attributes.

If in the attributes editor (in the Type column) the type description is displayed in parentheses, for example (CatalogObject.Items), this means that the applied type will be converted to the form data types.
If the form data contains an invalid type (such as an attribute of a form attribute), then an warning will be issued in the form attributes editor, next to the name of the attribute type, "Not available in the form data". In 1C:Enterprise mode, the form field corresponding to the given attribute will not be generated. No errors will be fired, though. When copying and editing objects containing such data, you should independently implement the program code that provides copying and working with such attributes within the form.
Next, we shall look at some features of the form attributes belonging to various types:

- For form attributes such as `ValuesTable` and `ValueTree`, there's an option to add columns (the Add Attribute Column command). This defines the data structure of the collection being created.
- For attributes of the `FormDataCollection` type (for example, tabular sections of objects) and `FormDataStructureAndCollection` (for example, sets of register records), it is possible to specify additional columns for attributes (the Add Attribute Column command) that have no association with the data stored in the infobase.
  The system will create these columns as the form data are being generated. Addressing such attributes is available both on the front-end and on the back-end.

An example of filling in a column of attribute not associated with the form data:

```
&OnServer
Procedure OnReadAtServer(CurrentObject)

  ForEach Row of Object.Item Do
    // TypeOfItem is an attribute not associated with the form data
    Row.TypeOfItem = Row.Item.Type;
  EndDo

EndProcedure
```

- For an attribute of the `ValueList` type, it is possible to set the value type (property Value Type) that will store the list. At the same time, the system will automatically limit the type of data to be added when adding interactively. Also, programmatic addition is not prohibited, yet an attempt will be made to cast the value of the added type to the limited type (or to a composite type). The Value Type property can also be associated with a form control and can interactively limit the data types that can be added to the value list.
- For an attribute of the `DynamicList` type, the list parameters can be set, such as the main table, settings, etc.

IMPORTANT! Details columns that are not associated with the data do not participate in the conversion of values between the form data and the infobase objects, and the other way round.

Form data modifications are displayed by the form controls as soon as the execution of the code in the 1C:Enterprise script language is completed or following a forced call of the `RefreshDataRepresentation()` method. Consider an example.

Suppose there is a `Counter` attribute of the `Number` type in the form. This attribute is displayed on the form by a field of the Indicator Field type . Let there be some operation, the progress of which should be displayed by the indicator. The operation is invoked by pressing a button on the form:

```
Procedure FormCommandHandler()

  For Counter=1 to 100 Do
    ExecuteOperation();
  EndDo

EndProcedure
```

If you click the button that initiates this command in 1C:Enterprise mode, the indicator will first be in the leftmost position, and then- immediately in the rightmost one (we assume that the `ExecuteOperation()` method runs for a while, which should allow us to see the indicator changing). That is, the indicator will be refreshed only when the execution of the command handler is over.

In order for the indicator to reflect the progress of the operation, the handler code must be replaced with the following:

```
Procedure FormCommandHandler()

  For Counter=1 to 100 Do
    ExecuteOperation();
    RefreshDataRepresentation();
  EndDo

EndProcedure
```

After this modification, the indicator will start changing during the execution of the code placed in the command handler.

NOTE. In the web client, calling the `RefreshDataRepresentation()` method will refresh the form controls only after the execution of the 1C:Enterprise script code piece.

7.2.3. Converting the applied objects data to form data and back

There is a set of global methods for converting the applied objects data to form data and back:

- `ValueToFormData()`,
- `FormDataToValue()`,
- `CopyFormData()`.

Methods that work with the applied objects are available only in the back-end procedures. The method for copying values between the form data is available both on the back-end and the frontend, since it does not request applied objects as parameters.

When converting form data to an applied object, their compatibility should be considered.

- `ValueToFormData()` - converts an applied object to form data.
- `FormDataToValue()` - converts form data to an applied object.
- `CopyFormData()`- makes a copy of form data with a compatible structure. It returns `True` if the copying has not failed, or `False` if the object structures are incompatible.

When converting form data to applied objects and back, objects are cached, with the relevance of the cached object version being checked.

NOTE. When executing standard actions (opening a form, invoking a standard `Record` command, etc.) in a form with the main attribute, the conversion is performed automatically.

Let's provide an example of how to use data conversion in one's own algorithms.

```
&OnServer
The OnCreateAtServer(Denial, StandardProcessing) procedure

  ObjectItem = Items.FindByDescription("Coffee-pot").GetObject();
  ValueToFormData(ObjectItem, Object);

EndProcedure

&OnClient
Procedure Write();

  WriteOnBackEnd();

EndProcedure

&OnServer
Procedure WriteOnBackEnd()

  ObjectItem = FormDataToValue(Object, Type("CatalogObject.Items"));
  ObjectItem.Write();

EndProcedure
```

Also, the `ClientApplicationForm` object has methods available on the back-end:

- `ValueToFormAttribute()` - converts an applied object to the specified form attribute.
- `FormAttributeToValue()` - converts data form attribute to an applied object.

Using these methods is usually more convenient, since they carry, for example, an information about the type of form attribute. Besides, the `FormAttributeToValue()` method sets the mapping between the form data and the object that is used when generating messages.

It should also be remembered that when converting the objects of the `ValuesTable` or `ValueTree` types to the form data (by using either the `ValueToFormData()` method, or the `ValueToFormAttribute()` method), the following specificity should be taken into account: All columns that exist in the form data should be present in the object under conversion.

IMPORTANT! Attribute columns that are not associated with the data do not participate in the conversion of values between the form data and infobase objects, or vice versa. Columns that are missing in the object data are cleared when converting to the form data.

When an object is being converted to the form data by the platform, or by calling `ValueToFormData()`, `ValueToFormAttribute()` methods, only the object data is transferred. The object inner state is not transferred to the form data. For example, the New Object Ref value that is set to an object by using the `SetNewObjectRef()` method will be lost during the process of converting an object to the form data and back.

In capacity of the first parameter of the `FormAttributeToValue()` and `FormDataToValue()` methods, only the form attributes of the following types can be utilized: `FormDataStructure`, `FormDataCollection`, `FormDataStructureAndCollection`, `FormDataTree`. Let us provide an example of using these methods.

```
&OnServer
Procedure RecomputeOnBackEnd()

  // Converts an Object attribute to an applied object.
  Document = FormAttributeToValue("Object");

  // Performs recomputing by the method defined in the document
module.
  Document.Recompute();

  // Converts an applied object back to an attribute.
  ValueToFormAttribute("Document", "Object");

EndProcedure
```

7.2.4. Properties of Attributes

This section contains descriptions of some properties of the form attributes.

Title- a text that is used as the title of the form control associated with the given attribute, if the Title property of the form control is not specified explicitly.

Main Attribute- determines that the given form attribute is the main one and thus determines the form extension.

Stored Data-if this property is set to an attribute, then an interactive modification of such attribute will result in:

- An attempt to block the associated form attribute.
- Setting a sign of change in the form (`Modified` check-box).
- If an attribute has the Stored Data property enabled and the form is in View Only mode, then all form controls associated with this attribute will also be in View Only state.

Fill Check-determines whether it is necessary to check this attribute for filling in (the property value is equal to Cast Error). The Fill Check is available only for the attributes of the following types:

- primitive data types (`Number`, `Row`, `Boolean`, `Date`, any type of link, standard period),
- value list,
- value tree,
- value table.

NOTE. Attributes filling check is performed similar to the `ValueIsFilled()` function. The filling check of tabular sections implies that the tabular section is considered to be filled when it contains at least one row.

7.3. Special types of form attributes

7.3.1. Dynamic list

7.3.1.1. General information

A dynamic list is a special data type that allows you to display arbitrary information from database tables on a form. To do this, you must specify a table you want to display the data from, or describe the resulting selection in the 1C:Enterprise query language.

This tool is based on a data composition system and provides options for sorting, selecting, searching, grouping, and conditional formatting of the received data. In this case, the source of data is a query which is either generated automatically by the system (based on the specified data), or is written by developer manually.

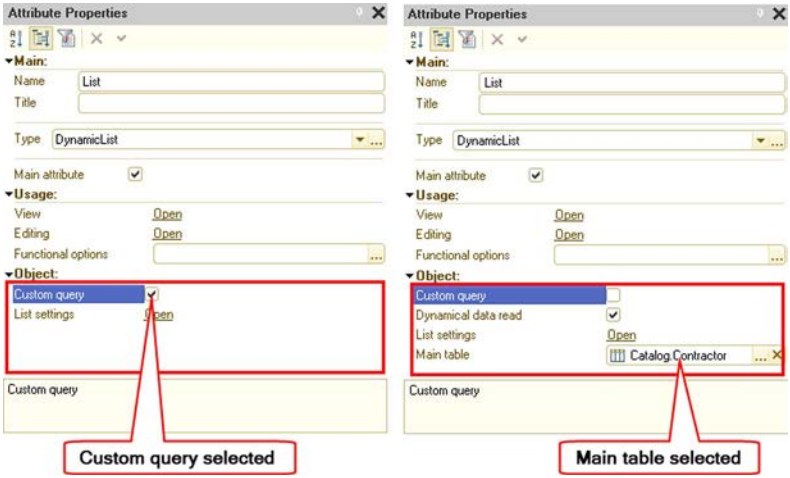


Fig. 139. Options for generating a dynamic list

When creating a form attribute of the DynamicList type, a developer can choose either of two ways to generating a data query:

- By specifying the main table- in this case, it is enough to simply point to the table (the Main Table property) you need to receive data from, and the system will automatically generate a data query (see the right part on fig. 139).
- By manual query input -for this you need to set the property Arbitrary Query (see the left partfig. 139). After that, a manual query input for receiving data from the infobase will become available.

By using a query, one can fetch data from several tables, so the main table can be specified. This is necessary so that the dynamic list can sort out which data are primary and which are -secondary, and hence, can correctly select and display information, as well as provide standard commands. However, if it is not possible to point out the main table in a query, then it may be left unspecified, but then the dynamic list will not provide the commands associated with the main table. Besides, in this case (without specifying the main table), the efficiency of retrieving data with a dynamic list will be significantly reduced.

In order to improve performance, it is recommended that all connections used in an arbitrary query only for additional data be made optional by extending the query language of the data composition system.

For a dynamic list, which is the main attribute of a form, there is an option to set the selection values by using the Selection form parameter. For this, it is necessary that the name of the structure property, which is in the Selection parameter, coincides with the name of the dynamic list selection field. In this case, the value of the structure property will be set as the right-side value of the selection element. If an array, a fixed array, or a value list is passed as the Selection item value of the dynamic list form, a condition with an option In List is added to the selection, in the right-side value of which the value list of is placed (into which an array and a fixed array are converted).

An arbitrary query in a dynamic list can be a query in which a parameter is used to generate the value of a field, for example:

```
SELECT
  CASE
    WHEN Delivery.Index = 1
      THEN &Presentation
    ESLE Delivery. Index
  END AS Undex
FROM
  Document.ProductDelivery AS Delivery
```

At the same time, if the type of the parameter's value differs from the type of the object's attribute (for example, Attribute1 is of the Number type, and the value of the parameter- is the String type), then to display the field correctly, the parameter value should explicitly cast to the desired type:

```
SELECT
  CASE
    WHEN Delivery.Index = 1
      THEN CAST(&Presentation AS Row(100))
    ESLE Delivery. Index
  END AS Undex
FROM
  Document.ProductDelivery AS Delivery
```

If a field by which the filtering is made is disabled by using the functional options, then a filter by that field is not set, even though the filter value is passed as the form parameters or the links of the filtering parameters.

By using the Read Data Dynamically property, a dynamic list is set to read data in small portions. Regardless of this feature, the following conditions apply:

- If the view mode is set as a hierarchical list, only the data of the current group and the data of all parent items (without their child items) will be read.
- If the tree view mode is selected, only the data of the open nodes of the tree will be read.
- Loading of all the dynamic list data at once is not supported if the hierarchical view is selected (the Display property is set to Tree) and when the initial display of the tree is set to Expand All Levels. To retrieve the data, as many requests to the server will be made as there are nodes in the displayed list.

The property Get Views for Invisible Fields handles views for the list fields that are not displayed on a form or are associated with actually invisible form controls:

- The property is set to `True`- For the thin and web client, the dynamic list receives views for all fields of the reference type that are passed to the client.
- The property is set to `False`- For the thin and web client, the dynamic list receives views only for those fields of the reference type that are displayed with actually visible form controls. If in this case a table of the managed form is in the selection mode, then the representation for the key fields is received at all events.

TIP! For a dynamic list with a large number of reference fields that are not associated with table columns or form controls, but have the Use Always property checked, setting the value of the property Get Views for Invisible Fields to `False` can accelerate the retrieval of data with a dynamic list. However, if the values from such fields somehow get into the form controls or participate in the generation of text strings, -do not set the Get Views for Invisible Fields property to `False`, as this will lead to additional server calls, to retrieve the data.

As part of a single data retrieval, a dynamic list reuses previously created temporary tables if the following conditions are met:

- In the batch list query, there is no query following the main batch query.
- The content of temporary tables and fields in them is unchanged from the previous execution of the batch query.

The dynamic list uses the values of the following properties of the metadata object attributes:

- Format,
- Editing Format,
- a tooltip,
- a flag for allocation of negative values,
- Mask,
- a flag for multiline mode,
- a flag for extended editing,
- Password Mode.

When displaying and editing the filtering and the data composition system parameters, the editing format of the corresponding field is used.

7.3.1.2. Restrictions and specific features

When setting a filter in a dynamic list, remember that the filtering does not affect groups once a dynamic list is set to the Hierarchical List or Tree viewing mode. The "groups" refer to catalog items or charts of characteristic types, for which the property `IsFolder` is set to `True`.

The filtering that is automatically applied by the dynamic list to the standard attributes such as `Owner`, `Parent`, `Date`, `Period`, and `IsFolder`, are made by standard means of the data composition system. The filtering that is automatically applied by a dynamic list to key fields can be made either by standard means of the data composition system or by directly adding the `B` condition to the query text for the fields of the main table. As a result of applying filters by means of composition, they can be applied both in nested queries and in the parameters of virtual tables.

When developing dynamic lists, it is recommended to check all dynamic lists with arbitrary queries. In the course of verification, ensure that if there are nested queries or virtual tables in the list query, and there are fields available for filtering with aliases that match the standard attribute aliases such as `Owner`, `Parent`, `Date`, `Period`, `IsFolder` or key fields, that these fields actually correspond to the standard attributes which their aliases matches with. If this is not so,- the query should be modified so that they match or an alias differ.

If manual query generation is selected, then some restrictions are imposed on the query:

- The use of the `FIRST` instruction in a dynamic list query is not supported. If it is necessary to use a filter with a limited number of rows in a dynamic list, you should rework the dynamic list generation query in such a way that the actual content part of the query is placed in a nested subquery and limit

the number of the entries to be received in this subquery. Instead of a nested subquery, a temporary table can be utilized.

- The filtering, sorting, and grouping are not supported:
  - By attributes of tabular sections.
  - Presentation fields.
  - Field `DataVersion`.
  - Field `PredefinedDataName`.
  - Field`View` Tables of chart of accounts.
  - Field `RecordType` tables of accumulation register.
  - Field `ValueType` tables of chart of characteristic types.
  - Field of type `Type`;
  - Field of type `String` (of unlimited length).
  - Field of type `BinaryData`.
- Sorting and grouping by the fields `ExtDimention<ExtDimentionNumber>` and `ExtDimentionType <ExtDimentionNumber>` of the table `RecordsWithExtDimensions` of the accounting register are not supported.
- Grouping by fields that are query language expressions containing aggregate functions is not supported.
- With the main table selected, the dynamic list query has the following limitations:
  - Merge is not supported
  - The use of the `ORDER BY` section is not supported. One should use a query without the main table or set up the necessary ordering through the dynamic list settings.
- If a dynamic list is displayed as a hierarchical list or a tree, then an entry will not be displayed by the dynamic list if at least one parent of this entry is not displayed. In other words,- to display an item in a hierarchical list, a dynamic list must also display all parents of that item, up to the top of the list. In this case, the top of the list stands for either the root item of the hierarchical object displayed by the dynamic list or an item set as the `TopLevelParent` property of the form table extension for the dynamic list.

The use of the following tables as the main table of the dynamic list is not supported:

- A table with no key that uniquely identifies each record in the table (reference for object tables and record key for register tables). Nonetheless, the following tables can be set as the main table of a dynamic list (despite the lack of the ID key):
  - `ExtDimention` table of the accounting registry;

    - all virtual tables of the accounting registry, except for the `RecordsWithExtDimensions` table;
    - tables containing the constant values (including the `Constants` table);
    - tables of external data sources with no key fields;
    - cube dimension tables for external data sources;
    - tables of accumulation register:
        - turnover tables;
        - balance tables;
        - turnover and balance table.
    - calculation register tables:
        - actual action period table;
        - schedule table;
        - base data table.
- Tables of tabular sections of objects;
- Change registration tables (utilized in the data exchange algorithms).
- Tables of sequences;
- Tables of recalculations (used in algorithms of periodic calculations).
- A table that is used in a query only in an external connection.

In other words, a dynamic list with the specified main table will work correctly if, as a result of a query specified as a data source, the number of rows obtained from the main table (taking into account the imposed filtering) does not increase. If, as a result of the query, the number of rows received by the query from the main table increases, this will violate the key uniqueness of the table entries displayed by the list. In this case, the use of the main table of the dynamic list ought to be disabled.

When working with a dynamic list, you must take into account the access rights to the attributes displayed in the list:

- Dynamic list columns with the Use Always property checked that a current user has no right to `View` are not passed to the front-end. Access to the data of such columns (by using the `CurrentData` property and the `RowData()` method) is not possible on the front-end.
- If the current user does not have the `View` right to the key field of the dynamic list, obtaining this dynamic list results in an access violation error.

For a dynamic list that displays an enumeration, there is no option to interactively customize the list.

The contents of the columns and the settings of the dynamic list are linked with the query fields by aliases of the filtered fields. If an alias is not explicitly specified in the query for the filtered field and the field is systemic, then the field name in the English version of the 1C:Enterprise language is used as the alias. The specified link means that if you change an alias (or explicitly specify an alias for a field that has an automatic alias) of the query field that generates dynamic list data, the dynamic list attribute settings will be lost, the form controls will "lose"

the attributes to display, the dynamic list settings will become invalid, and so forth.

In the case when the dynamic list data source is a table (basic or virtual) that allows you to set a filter by a time period, then if a user filters such dynamic list by a time period (the Set Range of Dates... command), the specified dates will be set as the filtering values or as the virtual table parameters. If by means of expanding the query language for the data composition system, the names of the parameters of the virtual table were explicitly specified,- the parameters with the specified names will be set. Tables that allow to filter the time period for displaying or processing data:

- tables of registers (basic or virtual) that have an option to filter the time period (for calculation registers - by recording period);
- basic tables of documents, business processes, and tasks;
- basic tables of document journals;
- basic tables of sequences, tables of sequence boundaries.

An array or a value list can be used as a parameter of a dynamic list query. However, when the parameter is a value list, only the first value of the list will be used as a filter. If a dynamic list uses a query with parameters, the initial setting of the parameters' values should be done in the `OnCreateAtServer` handler.

When displaying data from a dynamic list, the following features should be borne in mind:

- When programmatically changing the properties of a dynamic list, the command bars associated with this dynamic list are not automatically refilled.
- If several fields are grouped into a group with the In Cell grouping mode, and there is a field that is displayed by a checkmark in the grouped fields, then this checkbox will always be displayed first in the resulting cell (to the left of the text).
- No appearance is applicable to grouping lines: conditional appearance of a dynamic list or form and appearance which depends on table column settings (`Format`, `Font` etc.).

In the dynamic list, when determining the data type for fields whose expression components include parameters, fields or literals, the resulting type is determined by the types of fields and literals. If the type of the parameter value is not included in the resulting data type,- its value will be truncated.

For instance, in the following example, the field will be of the type `Number`.

```
CASE
  WHEN False
    THEN 5
  ELSE
    &Parameter
END
```

If you assign a value of a different type to the `Parameter` parameter, it will get the value `0` in the dynamic list for this field (the default value for the `Number` type).

If in such a situation it is required to select a parameter of another type,- it is recommended to use the `CAST` query language construct. For example, if it is desired to pass a string no longer than 100 characters to the parameter in the foregoing example, then the simple indication of the parameter should be replaced with an expression with an explicit type cast:

```
CASE
  WHEN False
    THEN 5
  ELSE
    CAST(&Parameter AS String(100))
END
```

If in an arbitrary text of a dynamic list query, the parameters are used in the expressions of the filtered fields,- the type of parameters should be explicitly specified by using the `CAST` construct. For example, instead of`&Product AS Product` `CAST(&Product AS Catalog.Product) AS Product` should be used. Otherwise, the search through the search string may not work correctly or fire errors.

7.3.1.3. Methods of getting and caching data by a dynamic list

When retrieving data for display, the dynamic list uses one of the following three methods:

1. The reading from the database is performed in portions, with the number of data items slightly exceeding the number of rows simultaneously displayed in the list (but not less than 20). Data is not cached on the back-end.
2. Reading from the database is performed in pages of 1 000 data items. Data is cached on the back-end. Hierarchical data is cached: for each parent, no more than 2 pages of items are cached. No more than 20 pages of items are cached per one dynamic list. Caching will be enabled by a dynamic list for the following tables:
   - Filtering Criteria:
   - All tables of the accounting register, except for the main table and the `RecordsWithExtDimensions` table;
   - All tables of the accumulation register, except for the main table;
   - All tables of the information register, except for the main table;

- All tables of the calculation register, except for the main table;
- `TasksByPerformer` virtual table;
- External data source tables with no keys;
- Cube dimension tables for external data sources.

3. Reading from the database is carried out in pages of 1 000 items each. The first portion is one page is size. Each next portion increases by 1 page (when the end of the previous selection is reached). The closer the "viewing position" to the end of the displayed data, the larger the selection is read from the database, ultimately becoming equal to all the displayed data. Data is cached on the back-end. The maximum number of entries in the cache and dynamic list -is 1,000,000.

Depending on what is selected by the main table of the dynamic list and what value the Dynamic data reading property takes on, differing methods of reading data are being used:

- The value of the Main Table property is one of the following tables: exchange plan, catalog, document list, document journal, chart of characteristic types, chart of accounts, chart of calculation types, business process, task, table of the business-process points:
  - Key identifying a table row: `Link`.
  - Dynamic data reading property:
    - Set: method 1 is used (methods described above).
    - Reset: method 2 is used (methods described above).
- The value of the Main Table property is one of the following tables: the main table of the information register, the main table of the accumulation register, the main table of the accounting register, the main table of the calculation register, the `RecordsWithExtDimensions` virtual table of the accounting register:
  - Key identifying a table row: `RecordKey`.
  - Dynamic data reading property:
    - Set: method 1 is used (methods described above).
    - Reset: method 2 is used (methods described above).
- The filtering criteria table or a table of tasks by performer (`TasksByPerformer`) are specified as the Main Table property:
  - Key identifying a table row: `Link`.
  - The Dynamic date reading property is not applicable.
  - Method 2 is used (methods described above).
- The virtual table of the information register `SliceFirst` or `SliceLast` is specified as the Main Table property:
  - Key identifying a table row: `RecordKey`.
  - The Dynamic date reading property is not applicable.
  - Method 2 is used (methods described above).

- One of the virtual tables of registers except for those mentioned above is specified as the Main Table property:
  - Key identifying a table row: Number.
  - The Dynamic date reading property is not applicable.
  - Method 3 is used (methods described above).
- The Main Table property is not specified, an arbitrary query is to be used: the behavior depends on the properties Key Type, Key Field, and Dynamic Data Reading.

To be displayed, the data is transferred to the front-end in portions, with the size of each chunk similar to the size of a portion in the first method of reading data (described at the beginning of this section).
When creating a form containing a dynamic list, initially 45 data items of each visible dynamic list are transferred to the front-end (if there are more than 45 items in the list). If a dynamic list displays more than 45 rows, then once the form is opened, an additional server call will be made to get the rest of the data items.

7.3.1.4. Dynamic list data obtainment and randomizing methods setting without the main table

If the main table is used in the dynamic list the system 1C:Enterprise knows how this table is organized and selects the relative reading mode. In the better part of cases, it provides rather optimal (from the point of view of resources use) way to retrieve the data for such dynamic search. When a spontaneous request is used in the dynamic search and the main table is not specified 1C:Enterprise is unable to determine what data acts as the table key and how to efficiently retrieve the data for such list. In addition, the developer might need to specify own key for the data displayed in the dynamic list. For example, a link acts as the key in the reference table. But the developer is dissatisfied with such key.
To specify for the system 1C:Enterprise the parameters of the key for the data of a spontaneous request without the main table, the dynamic list request editor offers the following properties which become accessible if the property Main table is not filled in:

- Key Type -specifies how the dynamic search string key will be formed.
- Key Field -contains the list of fields to be included in the key. The property accessibility depends on the value of the field Key Type.

Making a decision on the type and content of the data included in the key, remember that the formed key shall provide a unique value of all lines which will return the dynamic research request. If the requirement is not observed -the dynamic search will not be displayed and the message of the nonunique key will be shown.

More details on the properties intended for the key formation.
Key Type property may have the following values:

- Filed Value -in this case, the data returned by the request have one key field. The field acting as the key is determined by the property Key Field.
  `CurrentLine` property value type: field value type selected by the key field.
- Line Key -in this case, the key includes several fields. One field is allowed as well. Key Field property is intended for the specified of the fields included in the key.
  Property value type `CurrentLine:` type value `DynamicSearchStringKey`.
- Line number -is used when no key can be determined for the data returned by the dynamic search request. In this case, the entry sequence number from the beginning of the fetch will be the key.
  `CurrentLine` property value type: `Number` type value.
- Auto -in this case, the platform tries to determine the key type based on the property Key Field and degree of filling of the property Main table:
  - The main table is not specified:
    - The key fields are not set. In this case, the platform will use Key Number as the key type.
    - One value is specified in the key fields. In this case, the platform will use Field Value as the key type.
    - Several values are specified in the key fields. In this case, the platform will use Line Key as the key type.
  - The main table is specified:
    - The used key type depends on the used table.

Key Field property allows selecting one or several fields which participate in the key record formation. All accessible fields of the dynamic list request which comply with the following restrictions are enumerated in the list of accessible fields:

- The field can be used for the display in the dynamic list and also for the display and ranking with the accessible types of comparison Equal and In the list.
- If the field is a key field of Field Value type the values of this field can't include `Undefined` and `NULL`.

The property Dynamic data reading can change when the values Field Value or Line Key are selected as the key type. The dynamic data reading method depends on the key type selected for the list and the state of the check box Dynamic data reading:

Key type	**Dynamic data reading is enabled**	**Dynamic data reading is disabled**
Field value	Method 1	Method 2
Row key	Method 1	Method 2

Key type	Dynamic data reading is enabled	Dynamic data reading is disabled
Row number	Not applicable	Method 3

7.3.1.5. Dynamic list settings

The List Setting property -Clicking on the Open link opens a form for setting the display of a dynamic list. Setting up the list is performed in the same fashion as the similar operations in the data composition system.

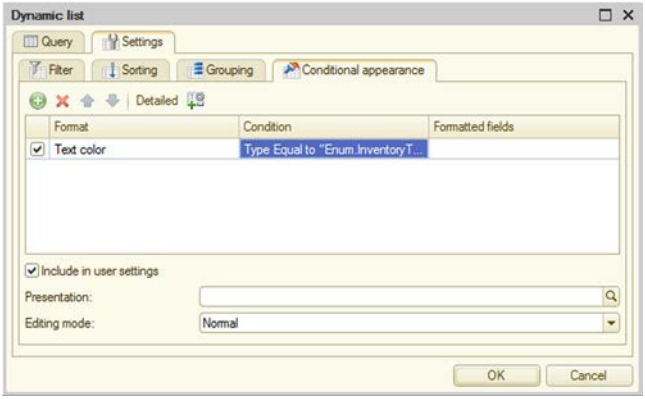


Fig. 140. Conditional appearance of dynamic list

While customizing a dynamic list in the configuration, the application developer has the following options:

- specify the fields desired for further ordering;
- describe the list filtering rules;
- specify the conditional appearance settings;
- Specify the fields desired for further data grouping.

It makes sense to specify the sorting order settings only if the default sorting order does not suit a developer's needs.

TIP. It should be remembered that an inapt selection of fields for sorting (as well as for the data filtering and grouping) adversely affects the efficiency of the dynamic list selective functions.

From the perspective of an application developer, the dynamic list settings consist of several parts that are interconnected. The main property through which the dynamic list settings can be managed is the `SettingsComposer`. This object contains three sets of settings that, when the system is running, determine the final settings that are applied to the dynamic list:

- `Settings` - settings made in Designer mode. The dynamic list property `Order` provides quick access to the `Settings.SettingsComposer`

dynamic list settings property. Thus, the following constructs are equivalent: `List.Order` and `List.SettingsComposer.Settings.Order`;

- `UserSettings` - are settings modified by a user in 1C:Enterprise mode;
- `FixedSettings` - these settings are specified from within the 1C:Enterprise script language. Also, this property embraces the filtering values that are passed to the form by using its parameters. The properties of the dynamic list `Filter`, `Parameters`, `ConditionalAppearance` provide quick access to the fixed settings of the dynamic list settings composer. In other words, the following access references are equivalent: `List.SettingsComposer.FixedSettings.Filter` and `List.Filter`.

When forming the final setting of the dynamic list, various types of settings are combined as follows:

- If any type of settings is entirely marked as custom, then the custom settings (`List.SettingsComposer.UserSettings`) are included into the resulting settings. Moreover, if any settings items are marked as inaccessible, then these settings will be placed into the resulting settings from the `List.SettingsComposer property. Settings`.
- If any type of settings is marked as custom not entirely but item-wise, then:
  - Items marked as custom will be placed in the resulting settings from the `List.SettingsComposer.UserSettings` property.
  - Items marked as inaccessible will come to the resulting settings from the `List.SettingsComposer.Settings` property.
- Fixed settings (`List.SettingsComposer.FixedSettings`) are added to the resulting settings "as is". Altogether, it is unacceptable when there are settings with the same name to be found in fixed and custom settings, for example, a filter with the same left-side value in its condition.

If there are setting items in the dynamic list settings that are disabled by using the functional options, these items will be removed from the list of available settings when retrieving the dynamic list data.

Management of which settings will be available to user, and which will -not, is performed in the dynamic list settings window.

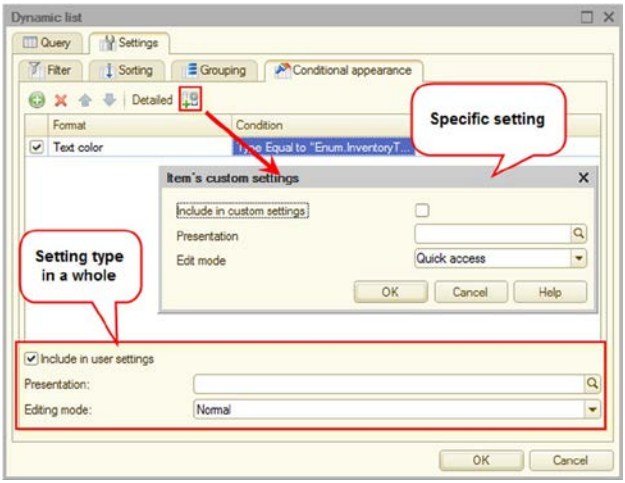


Fig. 141. Managing the inclusion in custom settings

The checkbox in the lower part of the window (seefig. 141) is responsible for placingall types of setting items in the settings (normal or quick ones). This option is available for filtering, ordering, grouping, and conditional appearance. If the settings are specified with the Quick Selection editing mode, then in the Custom Settings Group property of the form table that displays a dynamic list, one must specify an empty form group in which the items associated with the quick dynamic list custom settings will be placed. If such group is not specified,- quick custom settings will not be displayed on the form. It is also possible to explicitly generate the creation of custom settings in 1C:Enterprise language by using the `CreateUserSettingsFormItems()` method of the dynamic list extension.

It is also possible to choose an option for placing specific setting items in custom settings. This option is available for the filtering and conditional appearance items (see fig. 141).

If it is desired to load any special setting items when opening a dynamic list, this can be done in two ways:

- By using the `UserSettings` dynamic list form parameter The data contained in this parameter will be placed in the dynamic list custom settings.
- By using the `UserSettingsKey` dynamic list form parameter If this parameter is specified when the form is being opened, then the custom settings stored in the settings storage with the specified key will be loaded into the dynamic list, which is the main form attribute.

7.3.1.6. Search in dynamic list

A dynamic list located on the form provides an ability to perform an interactive search in the displayed data. Interactive search tools are managed using dedicated form item add-ons.

Let's review some specific features of using search in a dynamic list:

- In order for the search to be convenient to use (from the viewpoint of speed search, as well), the full-text search for all configuration objects that can be used as the main dynamic list table should be enabled. Also, all attributes of the configuration objects that can be displayed in a dynamic list and which may be required when searching should be included in the full-text search.

  If an object is excluded from full-text search, then the search mechanism in question will work, but the performance of such a search will be extremely low. It is not recommended to use search on objects that are not indexed by the full-text search.
- The application should have a routine task that regularly updates the full-text search index.
- The search is performed not on all columns of the dynamic list (and the configuration object), but only on those columns that are displayed in the table.
- Search in the dynamic list by fields of reference types with an arbitrary representation is performed by the fields that are used to form the representation. The fields that are included in the view are obtained with the `PresentationFieldsGetProcessing()` handler of the corresponding object.
- For dynamic lists with the main table specified, full-text search in the main table is applied. All non-indexed links from the main table will be appended to the full-text search results. The full-text search result for the main table is used as a filter by key fields. Also, a full-text search is performed by the fields displayed in the list from other tables (if a full-text search is used for the field and the configuration object). Without full-text search enabled, data can be found but the search itself will be very slow.

  If an error occurred while attempting to perform full-text search, the search will be performed without applying full-text search. For instance, this can happen when searching for one character and a large number of strings in the infobase starting with this character.
- In the event that a filter is used for the field of the main table of the dynamic list with the comparison type Equals, then the filtering value will be added to the search query for this table when performing a full-text search.
- The search string is split into words. The search string should not exceed 1,000 characters, and the number of words obtained during splitting should not be more than 20. The splitting is done in compliance with the following rules:

- A string is split by using the space, non-breakable space and tab whitespace characters for separators.
- Further, each fragment so generated is processed. If a fragment is in the date format (with or without time indication) with regard to the current regional settings of the session, then this fragment is considered to be a single word. Otherwise, each resulting substring is considered to represent a separate word.

- For each word, its own set of conditions is generated that are combined "by OR" This set of conditions is generated if either a full-text search for a given word in the table that the field was acquired from has returned at least one object, or the full-text search was not used for this field. Conditions are generated as follows:
  - For the field of the `String` type, the condition takes the form of `FieldName LIKE %Word%`.
  - For a field of type `Number`, the condition is given by `FieldName=Value`, where `Value`- is a word that is cast to the `Number` type. If the type cast is not possible,- a search by the field will not be performed.
  - The word is searched for as a substring in the default format for the `Boolean` type defined for the current session. If the matching word is found in a certain format, then the search is performed for the value corresponding to the form which the word is found in. In this case, the search does not use word formats that are set by using the `Format` form control property.
  - For the field of the `Date` type, the condition is given by `FieldName>= BegOfDay(Word) AND FieldName<=DayEnd(Word)`. If `Word` is similar to a date in which the year is indicated with one or two digits, the year will be converted to the current century, and this value will be inserted into the search condition.
  - For the fields of link type, search is performed on the fields used to generate the link format. For each of these fields, the search is performed according to the rules described above. The search does not use the fields used to generate an arbitrary data format.

  A set of conditions for each word is combined “by AND”.
- For values with leading zeros, you can perform a search on either the string with leading zeros, or on the string specified without them.
- If a dynamic list displays a list of documents or a journal of documents, the specified list viewing span is also displayed in the form area allocated for displaying the view status for the required dynamic list.
- The current value search command is not available if the main dynamic list table is a filtering criterion.
- Found pieces of strings are highlighted when displayed in the table.

- Only one search string is supported for a column. When a new search query for a column that has already been searched for is added, the new search expression will replace the previous one rather than add to it.
- If a form is lacking such form control extension as Search String Display associated with a table (the Source property of the form control extension) that displays a dynamic list, then pressing the Ctrl + F key shortcut opens the search dialog box.

  Otherwise, if a form features such form control extension as Search String Display associated with a table (the Source property of the form control extension) that displays a dynamic list, then the Extended Search command should be used, to open the search dialog box.
- When using the search dialog box, the following should be remembered:
  - Opening the search dialog box by using the key shortcut causes a value of the current cell to be copied into the What to Search For field, and the value of the Search Method to be set to By Exact Match.
  - Opening the search dialog box by typing directly in the search bar of the dynamic list causes the value of the Search Method switch to be set to By Substring, and the typed text is copied to the What To Search For field.

7.3.1.7. Retrieving data displayed in a dynamic list

When using dynamic lists, you may need to perform various actions on the data that is currently displayed in the dynamic list, taking into account the applied filters and search conditions. You can perform the following actions on a group of items: processing of the displayed information, for example, transferring selected documents or setting a certain attribute for selected objects, generating a list of available objects (with formatting, etc.), for instance, for further printing or saving to a tabular document.

To get the data that is displayed in a dynamic list, the `GetPerformingDataCompositionScheme()` and `GetPerformingDataCompositionSettings()` methods should be used.

An example of retrieving the data:

```
Scheme = Controls.List.GetPerformingDataCompositionScheme();
Settings = Controls.List.GetPerformingDataCompositionSettings();

LayoutComposer = New DataCompositionTemplateComposer();
CompositionLayout = LayoutComposer.Execute(Scheme, Settings);

CompositionProcessor = New DataCompositionProcessor;
CompositionProcessor.Initialize(CompositionLayout);
OutputProcessor = New
DataCompositionResultSpreadsheetDocumentOutputProcessor;

Return CompositionProcessor.Output(CompositionEngine);
```

Retrieving data to a collection of values (a table or a list of values) is done in a similar way.

Retrieving dynamic list data in this fashion has a number of specific features and limitations that should be borne in mind when developing applications:

- The following table formatting features are not supported:
  - Alternating row colors;
  - Table header image;
  - Table footer image;
  - Table footer background;
  - Table footer font color;
  - Table footer font;
  - Horizontal positioning in table footer;
  - Password mode.
- The conditional appearance specified for the managed form is not supported;
- The data acquired as a result of applying the `GetPerformingDataCompositionScheme()` and `GetPerformingDataCompositionSettings()` methods will not contain any modifications that can be made by using the `OnGetDataAtServer` event handler.
- When sorting a hierarchical table in ascending order by a field of the `Link` type, records containing a blank link are always placed first.

7.3.1.8. Additional processing of data retrieved by the dynamic list

Requirements for dynamic lists may include the display of some additional information that is difficult, inefficient or impossible to retrieve as part of a dynamic list request. In this case, the `OnGetDataAtServer` handler of the managed form table that displays the data of the dynamic list can be used. This is an out-of-context form handler that allows to perform the additional processing of data that a dynamic list gets for display.

This event is triggered after a dynamic list has received the data. The event is triggered if at least one row is received by the dynamic list.
Dynamic list settings (filtering (including search), sorting, grouping, conditional appearance, etc.) will be applied to the actual data received by the dynamic list request, before they are processed as part of the `OnGetDataAtServer` event handler.
The `OnGetDataAtServer` event handler receives three arguments:

- `ElementName`- contains the name of a table which this handler is executed for.
- `Settings`- contains a full copy of settings for a dynamic list. Special attention should be paid to the `AdditionalProperties` property for this argument. This property enables the transfer the requested data from a form to the out-of-context handler.
- `Rows`-this argument contains a collection of rows to be displayed by a dynamic list. The collection contains the actual data and the data formatting, but does not contain strings of groupings.

As an example, let's examine a list of sales items where the price of an item is displayed for the price category specified in a form attribute.

The dynamic list is located in the `List` attribute; it is displayed in the `List` table (a form control). Price type is configured by using the `PriceType` attribute of the `CatalogRef.PriceTypes` type.

NOTE. The examples below are not complete. They are intended to demonstrate various techniques of handling a dynamic list.

The request for a dynamic list itself is quite straightforward:

```
SELECT
  CatalogItems.Link,
  CatalogItems.DeletionMark,
  CatalogItems.Parent,
  CatalogItems.IsFolder,
  CatalogItems.Code,
  CatalogItems.Name,
  CatalogItems.SKU,
  CatalogItems.Supplier,
  CatalogItems.Type,
  CatalogItems.Description,
  CatalogItems.Predefined,
  CAST (NULL AS Number(10, 2)) AS Price
FROM
  Catalog.Items AS CatalogItems
```

This request displays the sales items catalog itself and “allocates a place” to put the product prices in (the `Price` column). The rest of the operations are executed in the `OnGetDataAtServer` event handler:

```
&OnServerWithoutContext
Procedure ListOnGetDataAtServer(ElementName, Settings, Rows)
  If Settings.AdditionalProperties.Property("PriceType") Then
    PriceType = Settings.AdditionalProperties.PriceType;
  EndIf;
  If NOT ValueIsFilled(PriceType) Then
    Return;
  EndIf;
  Query = New Query;
  Query.Text =
    "SELECT
    | Prices.Product,
    | Prices.Price
    |FROM
    | InformationRegister.ProductPrices.SliceLast(Product IN
(&Products) AND PriceType = &PriceType) AS Prices;
  Query.SetParameter("Products", Rows.GetKeys());
  Query.SetParameter("PriceType", PriceType);
  Selection = Query.Execute().Select();
  While Selection.Next() Loop
    СтрокаСписка = Строки[Выборка.Товар];
    ListRow.Data["Price"] = Selection.Price;
    ListRow.Formatting["Price"].SetParameterValue("Format", "ND=10;
NFD=2 ; NZ= ");
  EndDo;
EndProcedure
```

When considering the handler text one's attention should paid to the following points:

1. Due to the fact that the handler in question is out-of-context, the price category is retrieved in a special way, that is by using the additional properties of the `Settings` object passed by the handler's argument. The consequence of this is the fact that when changing the value of the form control displaying the `PriceType` attribute, one needs to set a new attribute value in the dynamic list settings, and then refresh the list:

```
List.SettingsComposer.Settings.AdditionalProperties.Insert("PriceType"
, PriceType);
Controls.List.Refresh();
```

2. Method for retrieving the key values for every record that will be displayed in a dynamic list. To do this, use the special `GetKeys()` method.
3. Data located in `Rows.Data` represents exactly that data set which will be passed to the front-end application to be displayed by a table of a managed form that is associated with a dynamic list. This transmission will be performed following the handler's execution.
4. Handling the dynamic list cells formatting (`Rows.Format`) does not differ from the handling the data composition system cells formatting.

If in the `OnGetDataAtServer` event handler the row data is being modified, it is recommended to exclude the fields in which the data is being modified from the list of fields used for grouping, sorting, and filtering. Failing to do this will

cause an unpredictable behavior when performing grouping, sorting, and filtering (including all types of search). The reason for such behavior is that all the specified operations are performed on the data that will be immediately returned by the dynamic list generation request. The result of the `OnGetDataAtServer` handler's output is not used for these operations. To set a restriction on the use of a field in a particular operation, the following dynamic list methods are used: `SetRestrictionsForUseInGroup()`, `SetRestrictionsForUseInOrder()`, and `SetRestrictionsForUseInFilter()`. The `GetRestrictionsForUse...()` methods (for the `DynamicList` attribute type) return the current list of fields that are not available for the execution of a particular operation, provided that the list of restrictions is specified by the `SetRestrictionsForUse...()` methods.

The data passed to the `OnGetDataAtServer` event handler has already been processed by the dynamic list conditional formatting. In this case, obviously, the data acquired from the query were used to compute the conditions. If it is required to change the dynamic list data formatting depending on the result of the `OnGetDataAtServer` handler's operation, then the formatting should be changed directly in the handler itself.

7.3.2. Formatted Document

In the process of developing applications, it becomes necessary to allow a user to perform text editing with formatting (hereinafter referred to as a formatted document). Writing e-mails, various memos or covering documents examplify such a need.

To work with a formatted document, a `FormattedDocument` object is intended with which you can perform program processing of a document, as well as a field of the Formatted Document Field type which is intended for interactively modifying a formatted document.

It is recommended to store the formatted document in the infobase as an attribute of the `ValueStorage` type, in which an object of the `FormattedDocument` type is placed

In order to allow user to interactively modify a document, the following steps should be taken:

- Create a form attribute of the `FormattedDocument` type and set the Saved Data property for it.
- Create a form control of the Input Field type in the form of Formatted Document Field and link it with the previously created attribute.
- When reading form data (`OnReadAtServer()` handler), load a document into the form attribute (`SetHTML()` method) that should be edited, after receiving it from the infobase.

- Prior to writing the form data (`BeforeWriteAtServer()` handler), get the modified document (`GetHTML()` method) and put it in the attribute stored in the infobase.

When processing a document, **bookmarks** can be used. A bookmark describes the position of a character in a document. When determining the position, the following should be considered:

- the document content is treated as a single sequence of characters;
- line break is treated as one character;
- an image is treated as one character.

If a character position which the bookmark is linked with changes, then the bookmark will trace the position of the linked character. If a character linked with the bookmark is deleted, the bookmark attempts to link with the next character, and if such character is not found,- the bookmark will point to the end of the document. Once an entire formatted document is replaced, all the bookmarks in this document will turn invalid.

By using bookmarks, one can get and set the text selection or cursor **position** in the editor, add and delete text elements in the document (the actual text, image, line feed).

If the start and end positions coincide when getting a selection, it means that there is no selection and the cursor position is received. The same applies to the setting of text selection: If the start and end positions coincide when setting a selection, the cursor position changes, with no text selection. When programmatically modifying the text, the actual display of a selection in the field of a formatted document will not change. If, prior to modifying a text, the selection borders have been retrieved, then after changing the text, the bookmarks can change their position while retaining their placement. In other words, if an insertion occurred inside the selection, then the position of a bookmark describing the end of the selection will be augmented by the size of the inserted text, etc.

Here is an example of getting and setting a selection in an editor for formatted documents.

```
// Content is a form attribute of the FormattedDocument type
// Editor is a form control of the Formatted Document Field type
// Start is a form attribute of the Number type. Points to the
position where the text selection starts.
// End is a form attribute of the Number type. Points to the position
where the text selection ends.

&OnClient
Procedure GetSelection()

  StartPosition = 0;
  EndPosition = 0;
  Controls.Editor.GetTextSelectionBounds(StartPosition,EndPosition);
  Start = Content.GetBookmarkPosition(StartPosition);
  End = Content.GetBookmarkPosition(EndPosition);

EndProcedure

&OnClient
Procedure SetSelection()

  StartPosition = Content.GetPositionBookmark(Start);
  EndPosition = Content.GetPositionBookmark(End);
  Controls.Editor.SetTextSelectionBounds(StartPosition,EndPosition);

EndProcedure
```

When performing operations of deletion, a bookmark describing the beginning of the deleted piece may turn out to be incorrect. If it is necessary in this case to retain the starting position of the deleted piece, the starting position of the deleted piece should be used. Thus, an example of replacing the selected text with another text would be as follows.

```
Procedure InsertStringIntoCurrentPosition(Editor, Content, String)
  Var Start, End;

  Editor.GetTextSelectionBounds(Start, End);
  Position = Content.GetBookmarkPosition(Start);

  Content.Detele(Start, End);

  Start = Content.GetPositionBookmark(Position);
  Content.Insert(Start, String);

  Bookmark = Content.GetPositionBookmark(Position + StrLen(String));

  Editor.SetTextSelectionBounds(Bookmark, Bookmark);

EndProcedure
```

If there are hyperlinks in the formatted document and the formatted document is displayed in View Only mode, then the hyperlinks are fully functional. They open in a new web browser window.

There is also an option for programmatic access to the formatted document text. The text itself consists of objects of the `FormattedDocumentParagraph` type accessible through the `Items` property of the `FormattedDocument` Object. Each paragraph in turn consists of the objects of the `FormattedDocumentText` type, the `FormattedDocumentLinefeed` type, and the `FormattedDocumentPicture` type (that is, the items of a formatted document). This collection is accessed through the `Items` property of the `FormattedDocumentParagraph` object. An item border is not necessarily a word border.

For instance, the following example demonstrates the enumeration of all paragraphs of a formatted document containing text and the separate processing of each paragraph.

Example:

```
For Each Paragraph In Document.Items Do
  If Paragraph.ParagraphType = ParagraphType.Regular Then
    ProcessRegularParagraph(Paragraph);
  ElseIf Paragraph.ParagraphType =ParagraphType.BulletedList Then
    ProcessList(Paragraph);
  ElseIf Paragraph.ParagraphType = ParagraphType.NumberedList Then
    ProcessList(Paragraph);
  Else
    Continue;
  EndIf;
EndDo;
```

Handling of a formatted document items incorporated in a paragraph may be useful if, for one, it is desired to remove the entire formatting in italic or bold font.

Example:

```
Bold = New Font(, , True);
Italic = New Font(, , True);
Regular = New Font;
For Each Item In Paragraph.Items Do
  If TypeOf(Item) = Type("FormattedDocumentLinefeed") Then
    Continue;
  EndIf;
  If Item.Font = Bold Or Item.Font = Italic Then
     Item.Font = Regular;
    EndIf;
EndDo;
```

In programmatic text processing, the `GetItems()` and `GenerateItems()` methods can also come in handy. These methods return an array of the formatted document items. To see the difference in the workings of these methods, consider an example of a formatted document containing the following string: 012 456 890. The string consists of 11 characters, where the characters 4 and 8 are spaces.

The task is to get items that are located between positions 2 and 9 of this document.

```
StartPosition = Document.GetPositionBookmark(2);
EndPosition = Document.GetPositionBookmark(9);

Output = Document.GetItems(StartPosition, EndPosition);
For Each Item In Output Do
  Report("Text is " + Item.Text);
EndDo;

Output = Document.GenerateItems(StartPosition, EndPosition);
For Each Item In Output Do
  Report("Text is " + Item.Text);
EndDo;
```

The difference in the output of these methods will be that the `GetItems()` method returns all the items that fall between the specified bookmarks (including all the border items), i.e. in this example, the following text will be output:

```
Text is 012 456 890
```

Whereas the `GenerateItems()` method will return the following:

```
Text is 2 456 8
```

In other words, when using the `GenerateItems()` method, the system generates a set of items in such a way that it includes only the contents of a formatted document which is delimited by bookmarks.
It should be remembered that the `GenerateItems()` method modifies the formatted document during its call, and if (in the given example) after calling the `GenerateItems()` method the `GetItems()` method is called once again, the output will be:

```
Text is 2 456 8
```

The output will be the same even if a set of items returned after the `GenerateItems()` method is called has not been modified.

7.3.3. Planner

7.3.3.1. General information

The `Planner` object (hereinafter referred to as the Planner) is designed for visualization of various date-and-time-based operations, with the possibility of repeating these events with a certain periodicity. The Planner is designed to display information and does not store any data in the infobase.
The general procedure is as follows:

- The data that should be displayed is retrieved from the infobase.
- This data is displayed in the Planner (as items).
- User edits these items.

- The data of the modified items is updated in the infobase.

The Planner provides an option to override the forms used to edit the items.

7.3.3.2. The Planner's Structure

7.3.3.2.1. Dimensions

General information

There is always one dimension in the Planner,- the **time scale**. Also, an application developer can create additional **dimensions** (the `Planner.Dimensions` collection) that will allow to specify the task of each item. Dimensions do not have an order of precedence, but on the form they are nested into each other by the order of addition. Each dimension is characterized by a value of an arbitrary type, which can then be used to search for the required dimension.

```
DimensionSalon = Planner.Dimensions.Add("Salon");
DimensionItem =
DimensionSalon.Items.Add(Enumerations.SalonType.ForMen);
DimensionItem =
DimensionSalon.Items.Add(Enumerations.SalonType.ForWomen);
DimensionStylist = Planner.Dimensions.Add("Stylist");
DimensionItem = DimensionStylist.Items.Add("Ivanov");
DimensionItem = DimensionStylist.Items.Add("Petrov");
DimensionItem = DimensionStylist.Items.Add("Sidorova");
```

In the sample code above, two dimensions are being added to the Planner:

- Salon- describes a type of some hairdresser's shop. Enumeration `SalonType` with two variables serves to determine the type of the salon: `ForMen` and `ForWomen`.
- Stylist- the family name of a stylist employed in a salon. A string with the stylist's family name will be the value type.

Columns/lines width and height

The form containing the planner field may change its sizes. The planner field tries to scale down the width of the columns and lines height so that all data of the planner are visible without scrolling. But such behavior not always positively affects the convenience of the result use.

Several properties managing the planner behavior while scaling down mitigate such problems:

- Auto minimum column width -if this box is checked the system will itself determine the minimal width of each column and will not allow the columns becoming narrower than the set minimal width.
- Minimum column width -if Auto minimum column width box is not checked the manual setting of the planer's column minimal width becomes

accessible. `0` means that the column will be compressed to the technical capability.

- Auto minimum column width and Minimum column width manage the height of the planner field line. The behavior is analogous to the properties of the column width management.

If the column (or line) reaches the set (manually or automatically) minimum the compression stops and the scroll bars appear in the planner.
The further decrease in the height and width of the planner field will lead to the appearance of a scroll bar in the relative coordinate.

Planner title freezing

When the scroll bars appear in the planner it becomes necessary to manage the behavior of the dimension titles (including the time scale). Title dock can be managed separately:

- The possibility to dock the dimension titles is managed with the help of the planner property Dock dimension title (`DockDimensionTitle`). Auto value is interpreted at Yes in all types of client applications.
- The possibility to dock the time scale title is managed with the help of the property Dock time scale title (`DockTimeScaleTitle`). Auto value is interpreted at Yes in all types of client applications.

When the web-client operates under the management of Microsoft Internet Explorer both described properties are always interpreted as No irrespective of the actual value.

7.3.3.2.2. Items

An **item** hence is to be found at the "intersection" of dimensions and the time scale. An items bears the following core specifications:

- Item description.
- The date and time of the start and the end of an action or event described by the item.
- Values of all dimensions specified for this instance of the Planner.
- Value of the item.
- The item reiteration options (for reiterating items).

In the simplest case, an item is created as follows:

```
Item = Planner.Items.Add(Date('20140606150000'),
Date('20140606154000'));
Item.Text = "Arrange Vasily's haircut";
Dimensions = New Map;
Dimensions.Add("Salon", Enumerations.SalonType.ForMen);
Dimensions.Insert("Stylist", "Ivanov");
Item.DimensionValues = New FixedMap(Dimensions);
```

In the sample code above, an item is created that describes an event that starts on June 6, 2014 at 3:00 pm and ends on the same day at 3:40 pm. The event will take place in the men's salon (with the value of the Salon dimension set to ForMen) and the assigned stylist will be Ivanov (with the Stylist dimension value set to Ivanov).

So far, the created item has a drawback, though: this item cannot be found in the Planner item collection (Planner.Items collection) by using the Find() method. In order to correct this drawback, it is necessary to assign some unique value at the point of creating an item as a value of the item (the Value property), in order to enable the unique identification of the items in the Planner.

One option for this unique identifier is the following:

- Create an Event catalog, with the composition of properties that enables full description of the saved items.
- Place the pointer to the Event catalog item in the Value property of the item being created.

Thus, each item in the Planner will be uniquely identified by a pointer to Event catalog item that the data for creating the item is received from.

If not all values of dimensions are specified at the creation of an item, then such item will not be displayed in the Planner.

7.3.3.2.3. Time Scale

Time scale placement

In some cases, it is convenient for the time scale to be located not vertically on the right, but along some other side of the form control that displays the Planner. This can be done by modifying the Planner.TimeScale.Position property.

Items behavior under low disc space

Several elements can be placed in one cell of the planner (dimension and time scale crossing).

The picture has enough space to place all elements by height. But there may be not enough space (for example, in case of scale-down of the vertical size of the form element of the type Planner Field) then the planner appearance will change.

The planner cell can't display all elements, therefore only the elements for which there is enough space are displayed and the remaining elements are "hidden" under the writing More X, where X is the -number of hidden elements in the cell. Such behavior is not always satisfactory.
The property of the form attribute Items behavior under low disc space (`ItemsBehaviorUnderLowDiscSpace`) is intended to solve this problem.

Specifying time scale settings

Apart from changing its position, the time scale can be adjusted in a number of ways. The first setting option is an ability to specify the units in which the time scale will be measured, and how many units will fit on the scale before moving to the next column (or row). For example, the time scale can be set to quantify time in hours, with 6 hours to be fit in one column (row). In such case, the display of one day in the Planner will consist of 4 columns (rows), depending on the time scale display orientation: 24 hours in a day and 6 hours in one column (row).

In 1C:Enterprise language, such behaviour is coded as follows:

```
Planner.PeriodicVariantUnit = TimeScaleUnitType.Hour;
Planner.PeriodicVariantRepetition = 6;
Planner.TimeScale.Position = TimeScalePosition.Top;
```

If the headers with the hour marks above every 6-hour sequence are unnecessary, then such transferred headers can be disabled by setting the `Planner.ShowWrappedTimeScaleHeaders` property to `False`.
The next option for setting the time scale is to change the current time scale or to set additional time scales (`Planner.TimeScale.Items` property). Changing the time scale is necessary in those cases when a standard time scale does not suit for some reason. For example, the unit of the time scale is day, the time scale is set to 1 (a week), and the Planner displays a time interval greater than one week (e.g. a month).

In order to visually mark the border, you need to add another time scale. The added time scale should represent a whole day:

```
TimeScaleItem = Planner.TimeScale.Items.Add();
TimeScaleItem.Unit = TimeScaleUnitType.Day;
TimeScaleItem.Frequency = 1;
```

As a result, the border between the two adjacent days will be more noticeable.

There can be made two improvements in the code snippet above: to put date before time and to change the format of the displayed date:

```
TimeScaleItem = Planner.TimeScale.Items.Add();
TimeScaleItem.Unit = TimeScaleUnitType.Day;
TimeScaleItem.Frequency = 1;
TimeScaleItem.Format = "L=; DLF=DD";
Planner.TimeScale.Items.Move(TimeScaleItem, 1);
```

The order of the time scales is determined by their order in the collection. In this example, the added scale follows the default scale, so the newly created scale will be shifted 1 position closer to the beginning of the collection. And by using the `Format` property, the manner in which the time of day start will be displayed is set. Now, the separation of time periods will become more clear and conspicuous.

It should also be remembered that it is impossible to remove an item from the `TimeScale` collection if it contains only one item. Therefore, one first need to add a new time scale, and only then the automatically added time scale can be removed.

7.3.3.2.4. Display periods

The scheduler can display different time periods. The time period may be one or more, the periods may be of an arbitrary duration. To set the display periods, the `Planner.CurrentRepresentationPeriods`collection should be used. The display periods should be specified based on time. If one sets the period by using the construction `Planner.CurrentRepresentationPeriods.Add(Date('20140601'), Date('20140601'));`, the first second of June 1, 2014, will be displayed. To display the day of the end of the period in its entirety, use the `EndOfDay()` function or explicitly specify the time of the end of the period.
The interaction of the specified display periods and the time scale frequency deserves special attention. All added periods are regarded as one continuous scale, and transfers of the time scale are processed by this scale. Suppose 7-day intervals should be displayed on the time scale for the next set of periods:

- June 1 - June 5;
- June 10 - June 19;
- June 21 - June 23;

The Planner is to be set as follows:

```
Planner.PeriodicVariantUnit = TimeScaleUnitType.Day;
Planner.PeriodicVariantRepetition = 7;

Planner.CurrentRepresentationPeriods.Clear();
Planner.CurrentRepresentationPeriods.Add(Date('20140601'),
EndOfDay(Date('20140605')));
Planner.CurrentRepresentationPeriods.Add(Date('20140610'),
EndOfDay(Date('20140619')));
Planner.CurrentRepresentationPeriods.Add(Date('20140621'),
EndOfDay(Date('20140623')));

TimeScaleItem = Planner.TimeScale.Items.Add();
TimeScaleItem.Unit = TimeScaleUnitType.Day;
TimeScaleItem.Frequency = 1;
TimeScaleItem.Format = "L=; DLF=DD";
Planner.TimeScale.Items.Remove(Planner.TimeScale.Items[0]);
```

The first 7 days completely covered the first interval (5 days, from June 1 to June 6) and 2 days from the second interval. The second 7 days covered 7 days of the second interval and the third 7 days covered the remainder of the second interval and the entire third interval.

Yet another way to set the display of discontinuous intervals is to set the `TimeScaleWrapBeginIndent` and `TimeScaleWrapEndIndent` properties. These properties are specified in the dimension of the `PeriodicVariantUnit` property. In other words, if the unit of the periodic option is day, then the indent from the beginning or end of the time scale will be calculated in days. For example, it is desirable to display June 2014 in the Planner in such a manner that only the first 5 (or less) days are displayed in each 7-day interval. To do this, use the following configuration code for the Planner (the sample code will not use the Stylist dimension from the examples above):

```
Planner.PeriodicVariantUnit = TimeScaleUnitType.Day;
Planner.PeriodicVariantRepetition = 7;
Planner.CurrentRepresentationPeriods.Clear();
Planner.CurrentRepresentationPeriods.Add(BegOfMonth('20140601'),
EndOfMonth('20140601'));
Planner.TimeScaleWrapEndIndent = 2;
TimeScaleItem = Planner.TimeScale.Items.Add();
TimeScaleItem.Unit = TimeScaleUnitType.Day;
TimeScaleItem.Frequency = 1;
TimeScaleItem.Format = "L=; DLF=DD";
Planner.TimeScale.Items.Remove(Planner.TimeScale.Items[0]);
```

In this case, the entire month of June is split into the following time periods:

1. June 1 - June 7;
2. June 8 - June 14;
3. June 15 - June 21;
4. June 22 - June 28;
5. June 29 - June 30;

Next, that number of days will be removed (from the beginning and end of the interval) from each interval which is specified in the `TimeScaleWrapEndIndent` and `TimeScaleWrapBeginIndent` properties (respectively), and the resulting intervals will be displayed. The last interval is taken up by the entire indentation, but the beginning of the interval will still be displayed (analogous to the setting the current display periods behavior).
The application developer has an option to independently change the displayed periods for navigation through the Planner (for example, by using the mouse wheel). To do this, the `OnCurrentRepresentationPeriodChange` event should be handled. If the `StandardProcessing` parameter in the event handler is set to `False`, the displayed periods specified in the `CurrentRepresentationPeriods` parameter will not be applied, and the displayed period will not change.

7.3.3.2.5. Overriding Item Editing Forms

There are two forms in the Planner, intended for editing items: Quick Edit form and Event Edit form. The following events are used, to override edit forms:

- `BeforeCreate`- in this event, you can override a form that will be used when creating a new event. Quick Event Edit form is used by default.
- `BeforeStartQuickEdit`- in this event, you can override a form that is used in the quick event edit. Quick Event Edit form is used by default.
- `BeforeStartEdit`- in this event, you can override a form that is used in the regular event edit. An event is triggered in the following cases:
  - A special button is pressed on the Quick control Edit system form. If the quick edit form is overridden, then the application developer needs to implement a feature for opening the regular event edit form. In this case, `BeforeStartEdit` event is not used.
  - The Edit command is selected in the context menu of an item;
  - An item displayed in the Planner is double-clicked.

In order to get the event editing parameters, the `SelectedItems` property on the form control that displays the Planner should be utilized.
In its essence, the form overriding is performed as follows:

1. Open the desirable form in an appropriate event handler;
2. Set the `StandardProcessing` parameter of the utilized handler to `False`.

7.3.3.2.6. Iterative Items

Items come as occurring once or iterative. Iterative items have a schedule (or rules for repeating the items). Schedule is specified by the `Schedule` property. Several elements grouped in the course of a schedule setting, form a series of

items. Within the series, it comes possible either to change the parameters of the entire series, or to change the parameters of one, specific, item of the series.
Iterative items are differently represented in an item collection and when displayed on a form. In the item collection, an iterative item will be given singularly, with the `Schedule` property being filled.
When being dicplayed on form, all iterative items will be present, however, the system does not provide an option to go through them. At the same time, when you try to edit any item that forms a series, the form control property `SelectedItems` will contain a collection of the Planner items that form a series for an item with a schedule. Such elements will not have the `Schedule` property set, but instead the `Parent` property will be filled.
The `Parent` property will refer to the item whose schedule has caused the creation of a series, part of which is meant to be edited.
If you manually edit one of the items of a series of iterative items in the Planner form, then the `ReplacementItems` property will get filled in the parent event of the series. This collection will show those items of the series the parameters of which have been modified (relative to the parent event). When creating a structure for item data storage, it should be remembered that not only parent items should be stored in this structure, but also the items that has some parameters differing from those specified for the series as a whole.
If a replacement item from a series is being edited, then the `ReplacementDate` property indicates the date and time of the creation of the series by the system.

7.3.3.3. General Model of the Planner Data Editing

As mentioned earlier, the Planner does not store data by itself. It is only a means of data display. Therefore, to store information about events in the infobase, an appropriate data structure should be created, for example, a catalog.
The general procedure is as follows:

1. The general parameters of the Planner, the number of dimentions, the time scale settings, etc. are defined.
2. The current display period of the Planner is determined;
3. In accordance with the established display period, data about the displayed information is retrieved from the infobase, and the required events are created in the form control. A reference to the catalog item that stores the data is specified as the `Value` property for the Planner item.
4. When editing or deleting the items, the `Value` property for events that have been modified or deleted is still retained. This can be done with the help of the form attributes that will store the values of the `Value` property of the modified or deleted items.
   New items can be determined by the blank `Value` property of the item. To do this, the entire `Planner.Items` collection should be iterated.

5. When changing the display period, the creation of items that belong to the new display period is performed.
6. Upon successful completion of work with the Planner, the planner data is synchronized with the data of the catalog in which the item data is being stored.

7.3.4. Charts

7.3.4.1. Overall information

A chart is a graphical representation of data. A simple kind of chart is a graph. The graph is used to display successive changes in values over time or depending on changes in other values. For example, with the help of a graph, a change in the price of goods during a year period can be demonstrated.

The vertical axis of the graph displays the values of the indicated value (price), with the graph time scale (months) -shown along its horizontal axis. If it is necessary to display various indicators on one graph, their values are shown as separate series.

In the example above, Apples and Eggs- represent series of the chart, while months -are the chart points.
For visual display of data of varying types different types of charts are being used. Three data types intended for the processing of charts are available:

- Chart. Chart is available for placement in a form, tabular document and data composition system.
- Gantt chart. Chart is available for placement in a form, and in a tabular document.
- Dendrogram. Dendrogram is available for placement in a form, and in a tabular document.

Further on, each of the main types of charts will be discussed in more detail.
The configuration containing the examples of construction of different types of diagrams is published on Information and Technology Support disk: https://its.1c.ru/db/metod8dev#content:5952:hdoc. This configuration is also available in the section of demo version on 1C firm web-site: http://demo.1c.ru/.

7.3.4.2. Interactive editing of chart properties

7.3.4.2.1. General information

The Designer allows to edit properties of the four main areas of a chart:

1. The entire chart.
2. Chart Plot Area -is the area where the chart itself is displayed.
3. Chart Title Area -is the area where the chart title is displayed.

4. Legend Area -is the area where the chart legend is displayed: that is how the displayed data and the way in which it is displayed is generally termed.

An example of the positioning of the chart areas is shown in **Ошибка! Источник ссылки не найден.**.

To edit chart properties, it is necessary to complete the following steps:

- On form:
  - Select a form attribute of the required type: `Chart`, `GanttChart` or `Dendrogram`.
  - Choose the GoTo command for the attribute.
  - In the dialog that opens, select the select object Data -Chart Type -Settings.
- In tabular document:
  - Select a figure of the required type (insertion of an icon is carried out by using the menu commands Service and Settings -Table -Figures).
  - Open the figure's properties palette.

In the properties palette, there is an option to choose which area of the chart the palette will display. To do this, use the drop-down list located below the search bar in the properties palette.

The drop-down box lists the four main chart areas described at the beginning of this section.

7.3.4.2.2. Chart properties editing

General information

The set of editable properties for the whole chart essentially depends on the type of the chart itself.

In a typical case, the availability of the chart itself, the chart title visibility, and the display parameters are being modified in the properties of a chart. In the properties palette, these parameters are arranged into the Basic, Specifications, and Formatting groups.

Also, there is an option to set up the series display parameters. These parameters depend on the chart type, yet there are three overall parameters:

- Display a graphical view of data in the chart- this parameter indicates how the selected series will be displayed in the chart:
  - Show - the selected series will always be displayed in the chart.
  - Don't show -the selected series will never be displayed in the chart, nor will it participate in the formation of a combined series.
  - Auto (default value) -the selected series will be displayed in the chart unless the delimitations for the number of series are enabled.
- Show in the legend -specifies how exactly the selected series will be displayed in the chart key:
  - Show - the series is displayed in the legend.

    - Don't show - the series is not displayed in the legend.
    - Auto (default value) -the display of the series in the legend is managed by the Display a graphical view of data in the chart property. A series will be displayed in the legend if it is displayed in the chart.
- Trendlines -allows to specify the parameters for the plotting of a series trendlines.

In case when the HideBaseValues property is set to True, the points with a base value are not taken into account in the calculation of trendlines. In stacked charts and normalized charts, trendlines are displayed "as is", that is with no stacking.

Chart

For objects of the Chart type, the chart type, data area, display specifications are being edited in the properties of the chart itself. Also, there is an option for a Chart object to edit the display parameters for each chart series and the specifications for displaying points. If it is necessary to change a series being set up, the Current series property of the chart should be used. If the chart uses delimitations for the number of series, a summary series is configured by selecting the Summary value in the Current series property.

The Series order in legend property specifies how the series list will be displayed in the chart key:

- Direct -the order of the series in legend corresponds to the order of adding the series.
- Reverse -the order of the series in legend corresponds to the reverse order of adding the series.
- Auto (default value) -in this case the order of the series in legend depends on the chart type:
    - The reverse order is used for the following chart types: horizontal histogram, horizontal 3D histogram, stacked graph, stacked area graph, normalized area graph, stacked histogram, stacked 3D histogram, normalized histogram, normalized 3D histogram.
    - For the rest chart types, the direct order is used.

The Values Axis group of properties describes the parameters with which the base value of a chart (the Base value property) and the rules for calculating the maximum and minimum values of the chart can be specified.

For specifying the maximum and minimum chart values, a few different options are provided:

- AutoDetect (default value) - in this case, the maximum / minimum value is determined based on the data for the display set by the chart.
- Use Value - in this case, the specific figures of the maximum / minimum values are specified by the figures in the Maximum value / Minimum value.

- Use value with restriction - in this case the maximum / minimum values are specified in the Maximum value / Minimum value properties, however, if the actual maximum / minimum values of the displayed data exceed (by module) the specified values, -the actual maximum / minimum values will be applied instead.

If the maximum and minimum values of the chart are specified in such a way that the minimum value is greater than the maximum one (for example, as a result of an error in the explicit setting of values), then the calculation of the actual maximum and minimum values is performed as follows:

- If the value of the Calculate chart maximum value property is equal to Use value, and the value of the Calculate chart minimum value property is not equal to Use value, then when calculating the interval:
  - The minimum value is calculated by using the formula `MaxValue * 2 – MinValue`.
  - The minimum value is adjusted (down) in such a way as to get a "good-looking" scale.
- If the value of the Calculate chart minimum value property is equal to Use value, and the value of the Calculate chart maximum value property is not equal to Use value, then when calculating the interval:
  - The maximum value is calculated by using the formula `MinValue * 2 – MaxValue`.
  - The maximum value is adjusted (down) in such a way as to get a "good-looking" scale.
- If the value of the Calculate chart minimum value property is equal to Use value, and the value of the Calculate chart maximum value property is equal to Use value as well, then when calculating the interval:
  - The maximum value is calculated by using the formula `MinValue * 2 – MaxValue`.
  - The maximum value is adjusted (down) in such a way as to get a "good-looking" scale. It should be noted that the maximum value adjustment is performed despite the fact that the algorithm for calculating the maximum value is set to Use value. This correction is performed since the maximum and minimum values are specified incorrectly.

If numeric values are plotted on the values axis, and a `Date` type value is specified as the maximum / minimum values, the corresponding specified value is ignored, and the maximum / minimum calculation is performed automatically.

For charts located in a spreadsheet document, there is an option to specify the spreadsheet document range, which will be the data source for the chart. The Data area property is earmarked for this end. In this case, there is an option to specify how the data series are arranged, either in rows or in columns (Series in rows property).

Thus, if we take for example a test data set that is used later in this chapter, then:

- If the Series in rows property is set, then, the values from the following rows will be used for the series: 14.03, 15.03, 16.03, 17.03.
- If the Series in rows property is reset, then the values from the columns Wholesale buyers, Retail buyers and Dealers will be used as the series.

Trendlines

Trendline is a geometric visualization of the average values for the analyzed indicators, acquired by using a certain mathematical function. The 1C:Enterprise system provides an ability to plot trendlines by using the following functions:

- Linear. The equation used is `y=a*x+b`.
- Exponential. The equation used is: `y=b*e^(a*x)`.
- Logarithmic The equation used is: `y=a*ln(x)+b`.
- Power. The equation used is `y=a*x^b`.
- Polynomial. The equation used is: `a0+a1*x+…+aN*(x^N)`, where `N` is more than 2.

NOTE. A description of the functions used and their specific features is beyond the scope of this documentation. The specific behavior of the functions being utilized and their parameterization should be sought in the relevant training materials.

Trendlines' editing is carried out in a special dialogue box.

For one series, more than one trendline can be specified. To do this, new trend line should be added in the dialog box. When adding a trendline, the following is specified:

- Type - what kind of function to be used for plotting the trendline. The list of functions is given above.
- Approximation order - determines the value of N in the case if the trendline type is a polynomial function.
- Factor - what will the interpolation value (factor):
  - Point number - factor will be the ordinal number of a point on this chart (numbering starts from 1).
  - Point value - factor will be the Value point property cast as the `Number` type. If the values of the series or points are dates, and for the trendline the Factor property is set to Point value, then when calculating the trend, the dates are converted to numbers as the number of days that have passed from the oldest date for this chart. The number of days can be fractional.
  - Auto (default value) -the factor will be either a point number or its value, depending on the values of all points.

- Value in the stock chart - both in the stock chart and in the candlestick chart, each point is specified by four values: opening price, maximum price, minimum price, closing price. This property indicates which values will be used for interpolation.
- Color -specifies the trendline color. If the color is set to Auto (to do this, clear the set value with the clear value button), then the color of the trendline will be:
  - if several series are being displayed, -it is equal to the color of the series which the trend line was created for (accordingly, when the color of the series changes, the color of the trendline changes, too);
  - if a single series is being displayed, - then it is black.
- Marker - specifies the form of the marker. The marker is displayed only at the beginning and at the end of a trendline.
- Line - specifies the style of line the trendline is plotted with.
- Text - a trendline name to be displayed in the chart legend.
- Equation area - this property describes where and how the actual equation describing the plotted trendline (if it is specified) will be displayed.
- The Show group of properties specifies how the trendline specifications will be displayed:
  - In Legend - whether or not the trendline will be present in the chart legend.
  - Equation - whether or not the actual interpolation equation will be shown within the chart.
  - Determination Factor -manages the display of determination factor. The determination factor evaluates the interpolation function with regard to the specified data and allows to evaluate the applicability of the function in use to the utilized data.

Reference lines and bands

When plotting charts, you may need to visually highlight a certain area of the chart, for instance, to accentuate a certain month or product on a sales graph. It is allowed to highlight points and intervals in the following types of charts: stock chart (all types), histogram (all types), graph (all types), scatter chart, bubble chart, radar chart (all types), and gauge chart.

To manage the highlighting of the chart itself, the following properties are provided:

- Value reference bands (`ValuesReferenceBands`) and Point reference bands (`PointsReferenceBands`) - allow to highlight one or several value bands on the corresponding axis. Band is a rectangle of a certain color that highlights a specific range of values on the corresponding axis.

- Value reference lines (`ValuesReferenceLines`) and Point Reference Lines (`PointsReferenceLines`) -allow to plot one or several lines that follow some specific values of the corresponding axis.

Each of given properties has a special editor that is used for setting up the bands and lines.

For the reference bands, the following properties are used:

- Start and End - describe a value band or a point bans. The types of values and the values themselves depend on the data in use.
- Text - a band caption.
- Tooltip - additional information that pops up when the mouse is hovering over the band.
- Text area - allows to specify the format of the band caption display.
- Color and Transparency - allow to specify the band color and the color's transparency. Transparency is set in a range from 0 (fully opaque) to 100 (fully transparent).
- Line, Line Color and Line Color Transparency - allow to specify how to display the band's border. Transparency is set in the same way as for the band display color.
- Display area start and Display area end - allow to manage the size of the band display area. By default, a band is displayed starting from the corresponding axis and up the entire height of the chart. In the example **Ошибка! Источник ссылки не найден.**, the default band display -is from the Sausage point (the values axis strictly) to the Biscuits point (the border of the plotting area). Nonetheless, such style of visualization is not appropriate at all times. In such cases, by using these properties, the "indents" from the lower and upper band borders can be set (in percent). The figure shows that the indents are set at 20% from the lower band border and 20% from the upper band border.

A set of properties that is used to visualize a reference line is in some ways different.

On the whole, these properties are similar to the reference band properties of the same name. It is worth to specifically mention the following properties:

- Value - specifies the value which the reference line will be plotted for. The types of values and the values themselves depend on the data in use.
- Position - allows to specify where the reference line will be plotted, either directly “through the value” or between the previous and the current values.

Value tooltips

The diagram allows setting how the tooltip is shown when the cursor is on the diagram. There are several properties for the tooltip setting.
The property Value tooltip representation (`ValueTooltipRepresentation`) sets at which point in time the diagram will display the tip:

- Show on hover -the pop-up tooltip is shown when the cursor is on the value. While the pointer is moving, the tooltip is hidden.
- Show for the nearest -the pop-up tip is shown for the nearest value. The tooltip is displayed near the value. When the pointer is moving, the animated tooltip is moving between the values.
- Don't show -the pop-up tooltip is not shown.
- Auto -depending on the diagram type is interpreted as follows:
  - As Show for the nearest for the graph (usual, stack, and radar) and dot diagrams.
  - As Show on hover for other types of diagrams.

If several series are shown in the graph it may be necessary for the tooltip to show the information on all series in the selected point at once. The diagram property Value tooltip fill mode (`ValueTooltipFillMode`) manages this feature of the diagram:

- One value -in this case, the pop-up tooltip shows the value on which the pointer points.
- All point values -in this case, the pop-up tooltip shows the values for all series in the given point.
- Auto is -interpreted as One value.

If all point values display is selected for the diagram the tooltip is shown in the following way:

- The tootill is always shown in Show for the nearest mode.
- The delay from the moment of cursor movement and tooltip display is regulated by the property Value tooltip representation.

The tooltip text setting in the form of a formatted line is another option of the tooltip representation.

It also shall be noted that the platform automatically adds a tooltip to the series marker at the beginning of the lime if no tooltip is set for the value or All point values mode is enabled. If the tooltip shows one value the tootip frame has the color of the series.
The property `DiagramValue.Tooltip` or the relative parameters of the diagram values setting methods are intended for the tooltip setting.

When showing the tooltip it may be required to select all values of the point at the value of which the pointer points. The properties Show pop-up informational line of points (`ShowPop-upInformationalLineOfPoints`) and Show pop-up informational line of values (`ShowPop-upInformationalLineOfValues`) are intended for this feature management. If such representation is allowed the informational line will be shown when the tooltip is shown.

Selecting in the diagram

The diagram allows interactively slecting some of its elements and processing the highlighted data. What can be selected in the diagram is set with the help of the diagram property Selection mode (`SelectionMode`):

- Values selection -allows interactively selecting the diagram series and values.
- Points selection -allows interactively selecting the diagram points. The mode is available only for the charts that support points axis and do not support series axis (line, step, stacked line, area, stacked area, and normalized area charts; all bar, column, and radar charts; and stock and open-high-low-close charts). In other types of diagrams, the values is interpreted as No.
- No -the diagram disallows the interactive change of selection.
- Auto -is interpreted as Value selection.

The information in the diagram can also be selected by left-clicking on the required series of the legend.

If the diagram allows selection the selection result may be handled using the `OnActivate` event handler. This event is accessible for the form field extension for the diagram filed.

The selection in the diagram is performed in the following way:

- Selection is performed:
  - by single left-clicking on the value;
  - by single left-clicking on the diagram series in the legend;
  - by selecting a rectangular area in the chart plotting area.
- If the selection is performed holding Ctrl key the selected object is added to the selection or deleted from it (if this object was added to the selection earlier).
- If the selection is performed without holding Ctrl key the new selection becomes the only one.

In case of any selection, the diagram is shown as follows:

- The diagram is shown as usual if nothing is selected in it.
- If any data is selected in the diagram:
  - The series with selected values are shown in the color set for the series. The selected values are displayed with a border.

- The series with no selected values are shown in a paler color set for the series.

Interactive modification of chart values

To display data using charts, 1C:Enterprise affords an opportunity to edit data so displayed interactively, i.e. directly in a dedicated chart. This option may be helpful in various scenarios, for instance, when you execute "occurs, if" scenarios or perform planning.
Interaction modification of chart values is managed independently for three objects which are involved in the chart generation:

1. Value Edit Modechart property. This property defines, whether interactive modification of default chart values is available, or not (for the entire chart). If it is set to Auto, no interactive modification is available for the selected chart (this value is interpreted as Not used).
2. Value Edit Modechart series property. This property defines, whether modification of a specific series is available, or not. If it is set to Auto, availability of an editing option is defined in the chart settings.
3. `ChartValue.EditState` chart value property. It either allows or restricts editing of a specific chart value. If it is set to `ChartValueEditMode.Auto`, availability of an editing option is defined in the series and chart settings, accordingly.

Therefore, it is evident that value editing settings allow it to flexibly manage an interactive editing tool. For instance, you can have the following chart:

- Display actual sales figures for the previous year broken down by months. Obviously, no editing of this data is available.
- Display sales effected in the current year. However:
  - Actual sales figures for the months which have expired cannot be edited.
  - Of course, you can edit sales figures for the remaining months to the year end. As such, sales targets for the remaining period before a year expires are edited.

For editing purposes, use `WhenEditValue` form item event handler. It can be called both during value modification and upon termination or cancellation of value editing. To define an action which is currently executed, use `EditingState` handler parameter:

- `Terminated` - means that editing has been successfully completed. `ChartValue` handler parameter specifies a new value.
- `Aborted` - means that Esc has been pressed during value editing process. `ChartValue` handler parameter specifies a value available before editing is started.
- `NotTerminated` - means that a user currently modifies this value. `ChartValue` handler parameter specifies a current value.

See the mouse pointer to know, whether you can edit a value, or not. When you move a mouse pointer over an area where interactive editing of a value is allowed, its appearance changes. To edit, left-click on the marker (or column side) in a chart and move the marker (or column side). Further:
You can edit in a mode incompatible with 8.3.15 for column chart, diagram, step diagram and diagram with areas. Editing is available as well for summarized and normalized series. To edit, drag it with a mouse.

- Left-click to lock a form (if deemed necessary) and start editing.
- When you move your mouse, current point value is modified depending on the selected scale range and mouse movement distance. In normalized charts, the overall amount used for normalization purposes is taken into account.
  When you move a mouse, `WhenEditingValue` form item event is generated with a new value and `EditingState` parameter set to `NotTerminated`.
- During interactive editing of values, scale values are recalculated and the chart is redrawn.
- If you abort editing by pressing `Esc`, the chart will accept the value available before editing is started.
  `WhenEditingValue` event is generated with the previous value and `EditingState` set to `Aborted`. Editing is over.
- If editing is successfully completed (mouse left button is released), `WhenEditingValue` event is generated with a new value and `EditingState` set to `Terminated`.
- During the last stage, `WhenEditing` form item event is generated, while `Modified` property is assigned to the form.

Secondary value scale

Technically, a charting mechanism ignores units of measure displayed when it generates respective charts. However, a user benefiting from a chart so generated cannot absorb information properly, if a chart and a scale specify values in dissimilar units of measure. As an example, consider a chart which describes sales of a product in quantitative and monetary terms or when amounts are specified in different currencies. Another example of difficulties which may arise when information is incorrectly display is a chart, where series values are specified in similar units of measure, while the value order is materially dissimilar.
To address these problems, use a secondary value scale. This can be done as follows:

1. Add a secondary scale for one or more series.
2. Whenever necessary, modify secondary scale parameters.

It can be added both for an ordinary chart (all options) and column chart (all options).

Secondary scale parameters are set similar to properties of the main value scale. These parameters are grouped in a chart property bar in Auxiliary Value Axis.
In the properties of a requires series, set Active Value Scale to Auxiliary. The chart will display a secondary value chart. Consider an example.
You sell a product with respective figures broken down by months (January, February and March). The price is RUB 150.00. The sales volume totals 2,500, 6,000 and 6,300 pcs (respectively). You need to have both diagrams in a single chart.

Obviously, it is next to almost impossible to use this chart, as the sales diagram in quantitative terms cannot be read.
So far as a secondary value scale is concerned, use default settings and assign it to Sum series. The digram is now more informative.

You can read diagrams being part of a chart (both in quantitative and monetary terms) and see the axis scale and values.
If only two series are displayed on a chart with a secondary scale, you can use the same color to display chart axes which is assigned to the said series. For that purpose, use Value Scale and Secondary Value Scale chart generation area properties.
Moreover, it should be noted that a secondary scale can be displayed as a backup vis-a-vis the main value scale. As such, select Secondary Value Scale (for chart generation area) and in the opening dialog box set Display property to Visible. Therefore, if no series is assigned to a secondary scale, it will be used as a backup for the main chart value scale.

Chart integration

When you use dissimilar items per chart, in addition to a secondary value scale a need may arise to display certain items using a chart of another kind. Consider an example.
You have a spreadsheet with certain sales figures.

	Apples	**Pears**	**Melons**
June	50	35	0
July	60	40	0
August	70	37	56
September	65	38	70

To create the chart, do the following:

- Select Melons series: in Current Series chart property select Melons.
- Set Graphic Display Kind series property to Diagram.

You can select another series diagram kind, if the entire chart is either a diagram (all options) or column chart (all options). You can select the following chart

kinds as other kinds: column chart, diagram, step diagram, diagram with areas and volume column chart.
1C:Enterprise introduces no restrictions as to the number of series with diagram kind which is dissimilar to that of the chart in general. You can modify the diagram kind for one or more series.

Series accumulation

The accumulation mode in a chart can be set by the following two methods:

1. Select the "accumulation" kind of a chart (Chart Kind property in the chart settings). So far as a chart of this kind is concerned, "with accumulation" will be displayed in its name, for instance, Column Chart with Accumulation or Diagram with Accumulation. As such, accumulation is applicable to all series displayed on a chart. Specific information having regard to "accumulation" charts is available in the section dealing with various kinds of charts.
2. Select accumulation mode independently for a proper series displayed on a chart. Individual accumulation mode is managed by the following two properties: Accumulation Mode and Accumulation Group. This section deals with these properties.

For example, let's consider the following spreadsheet:

	Apples	**Pears**	**Melons**	**Rockmelons**
June	50	35	0	0
July	60	40	0	0
August	70	37	56	70
September	65	38	70	90

Similarly, follow the above steps when a chart is displayed with accumulation in general, while you need to exclude a specific series from the accumulation mode. For that purpose, set Accumulation Mode to Without Accumulation. This is the case, for example, when you need to display a specific series on a chart with accumulation with another chart kind without prejudice to a diagram displayed with accumulation. If Accumulation Mode is set to Auto, series accumulation mode is defined by the chart kind.
Let's assume that you need to display a chart in a way, when apples and pears are in a single column in a column chart, while the other includes - rockmelons and melons. For that purpose, display your spreadsheet as a chart with accumulation.

Now, per each series (Apples, Pears, Melons, Rockmelons) set Accumulation Group to any of the following:

- So far as Apples and Pears series are concerned, set to From trees.

- So far as Melons and Rockmelons series are concerned, set to From ground.

Therefore, Accumulation Group series property defines the way series involved in accumulation are to be summarized in. So far as series with similar Accumulation Group are concerned -, summarization is made jointly. The accumulation group name can be set at the user's discretion. It only identifies the group for the platform and is not displayed to the user.

7.3.4.2.3. Editing of Plotting Area Properties

When editing the properties of the plotting area, it is possible to describe how the scale of points, series and values will be visualized. These settings are made by using the plotting area properties of the same name: Point Scale, Series Scale, and Value Scale.
Each of the scales is configured by using the chart scale editor.

Let us look at the editor's options in more detail.

Scale settings

The Scale Position property -sets the chart scale position:

- Auto - For value scales, use Edge. For the point scales in the scatter and bubble charts-, use Edge. Otherwise -use the Base value.
- Base value- a scale is displayed on the base value of the perpendicular axis:
  - for a scale of points - it is positioned on the base value of the value axis;
  - for a scale of values- it is positioned on the base value of the point axis;
- Edge -The vertical scale is placed to the left of the chart, while the horizontal one- is placed below the chart.

The Scale Line and the Scale Line Color properties -specify the style and color of a scale line. The line style can be either selected from a drop-down list or configured in a dialog box. Color is configured by using a color dialog box.
Scale step property -allows to specify the scale step value. If the scale values are discrete, the step value is getting rounded to the nearest integer:

- If step = 1, all values have matching scale marks.
- If the scale step equals to 2, every even step is visualized, etc.
- If the value is assigned to 0, the scale step is calculated automatically.

The Scale Marks property allows to specify the position of the chart scale marks relative to the scale:

- Outside- marks are displayed on the outer side of the scale;
- Inside- marks are displayed on the inner side of the scale;

- Centered- marks are placed at the same distance from both sides of the scale;
- None -the scale marks are not displayed;
- Auto:
  - the scale marks are displayed on the outer side as long as the chart captions are displayed on the outer side, too.
  - the scale marks are displayed on the inner side as long as the chart captions are displayed on the inner side, too.

Grid settings

Show grid lines property allows to manage the grid line display:

- Show -shows grid lines;
- Don't show - hides grid lines;
- Auto:
  - grid lines are shown for value and point scales of the scatter and bubble charts;
  - for the rest of the charts, grid lines are hidden.

The Grid line and Grid line color- properties specify the grid line style and color respectively. The line style can be selected from a drop-down list or configure through a dialog box. Color is configured by using a color dialog box.

Scale label settings

Caption position- property sets the position of captions (labels) in relation to the scale:

- Don't show- hides scale captions.
- Inside -scale captions are displayed on the inner side of the scale;
- Outside -scale captions are displayed on the outer side of the scale;
- Auto-is interpreted as Outside.

Caption format property allows to specify a format string for conversion of a scale value into text.

Caption font and Caption color properties allow to specify the font and color of the scale caption (respectively).

Caption orientation property specifies the scale caption orientation:

- Horizontal- the caption text is displayed horizontally;
- Vertical- the caption text is displayed vertically;
- Custom angle -the caption text angle is specified by the Caption angle property:
  - The property's positive value rotates around the left caption edge, with 0 degrees- positioning the caption on the right side, and 180 degrees -on the left side.
  - The property's negative value rotates around the right caption edge, with 0 degrees- positioning the caption on the left side, and 180 degrees -on the right.

    - 0 degrees is the same is Horizontal value.
    - 90 degrees and -90 degrees are equivalent to the Vertical value, with an altering position of the text baseline.
- Auto - captions will be displayed horizontally if the chart location allows so, otherwise, -captions are displayed vertically.

The Maximum number of caption lines property allows to specify the maximum number of text lines to be used to display the labels. If the specified value is 0,- the number of lines will be calculated automatically.

Title settings

Caption display property sets the way a caption is displayed:

- Auto - shows a caption so long as it contains text to be displayed.
- Don't show -hides a caption.
- Show -shows a caption.

Title text source - property specifies the way a scale title is generated:

- Auto -if Title text property contains some text, the property's value is used. Otherwise, the text is generated on the basis of the chart data.
- Use text - the text specified in the Title text property is being used.
- Autotext - the text is generated on the basis of the chart data.

Title text property allows to explicitly specify the scale title.

Title position property specifies an area to output the title

- Beside the axit- the title is positioned by the chart axis.
- In plotting area- the title is positioned in the plotting area.
- In special area - the title is displayed in the special area:
    - for the horizontal axis, -it is placed below;
    - for the vertical axis, -it is placed on the left;

Title area property allows to configure a label area for a chart scale.

When configuring a label area, certain rules for the title area location should be considered. These rules are determined by the values of the Title position property (the chart scale settings dialog box) and those of the Position property group (the title area settings dialog box):

- The Title position property is set to In special area value. The Type of position value of the title settings area:
    - If an axis is positioned horizontally:
        - Auto, Center, Vacant location, Edge, Edge auto or Edge inside, - a label is displayed in the special area, centered in the middle.
        - Left top or Left bottom - a label is displayed in the special area, on the left.
        - Right top or Right bottom - a label is displayed in the special area, on the right.

    - Coordinates - a label is displayed in the special area, with Top and Left properties specifying the offset of its position inside the special area. In this case, 0- is the minimum value for the corresponding coordinate, and 1 -is the maximum value Thus, a value of 0,5 is the middle of the special area allocated for displaying the title.
- If an axis is positioned horizontally:
    - Auto, Center, Vacant location, Edge, Edge auto or Edge inside, - a label is displayed in the special area, centered in the middle.
    - Left top or Right top -a label is displayed in the special area, at the top.
    - Left bottom or Right bottom - a label is displayed in the special area, at the bottom.
    - Coordinates - a label is displayed in the special area, with Top and Left properties specifying the offset of its position inside the special area. In this case, 0- is the minimum value for the corresponding coordinate, and 1 -is the maximum value Thus, a value of 0,5 is the middle of the special area allocated for displaying the title.
- Title position property is set to In plotting area value. The Type of position value of the title settings area:
    - Edge and Edge auto - the vertical axis label is positioned to the left of the plotting area, whereas the horizontal axis is positioned at the top of the plotting area.
    - Edge inside-the vertical axis label is positioned to the left of the plotting area, whereas the horizontal axis is positioned at the bottom of the plotting area.
    - Center -a label is positioned in the middle of the plotting area
    - Auto - the same as Vacant location.
    - Top left -a label is positioned at the top left corner of the plotting area.
    - Bottom left -a label is positioned at the bottom left corner of the plotting area.
    - Top right -a label is positioned at the top right corner of the plotting area.
    - Bottom right -a label is positioned at the bottom right corner of the plotting area.
    - Coordinates - a label is displayed in the special area, with Top and Left properties specifying the offset of its position inside the special area. In this case, 0- is the minimum value for the corresponding coordinate, and 1 -is the maximum value Thus, a value of 0,5 is the middle of the special area allocated for displaying the title.
    - Vacant location-the system seeks for a suitable place within the plotting area.

- Title position property is set to Beside axis value. In this case, the behavior is similar to that when the title location is set to In plotting area value, but instead of the plotting area, the area beside the corresponding axis is selected. Depending on an axis position, the area has the following size and location:
  - An axis is positioned horizontally: the area is a rectangle with its width being equal to the length of the axis, and its height -being equal to the double height of the label.
  - An axis is positioned vertically: the area is a rectangle with its width being equal to the double height of the label, and its height -being equal to the length of the axis.

7.3.4.3. Chart

7.3.4.3.1. Overall information

The form attribute of the Diagram type is displayed on the managed form by using the form field of the Chart field format.

To configure a chart properties, use a special form that is invoked either from the property palette (by clicking on the Open link from the Settings property) or by using the GoTo command selected from the context menu of the form control belonging to the Chart field format. If the setting form is opened by using the GoTo command, then not the chart itself but its configuration form should be selected in the Select object dialog box.

The configuration form allows to fine-tune both the chart as a whole and its individual areas.

Chart type property specifies the method of data visualization. The availability of the chart's specific properties depends on the selected chart type. Chart field consists of three areas:

1. plotting area,
2. chart legend area,
3. and title area.

The plotting area may contain the following components:

1. point scale,
2. series scale,
3. value scale,

4. data table (spreadsheet).

To display a data table, `ShowDataTable` flag of the chart plotting area should be ticked. This property is not available for every chart type, though.

7.3.4.3.2. Operations with charts by using 1C:Enterprise language

Operations with chart data

Linking a `Chart` object to data can be implemented by using either of two following methods.
The first method is based on the possibility to generate chart series and points, as well as to set their values, directly from the 1C:Enterprise language. The example below generates a series of ten points for a chart with random data.

```
Generator = New RandomNumberGenerator;
Chart.Clear();
Chart.Refresh = False;
Series = Chart.Series.Add("Series");
For Counter = 1 To 10 Do
  Point = Chart.Points.Add("Point No " + String(Counter));
  Value = RandomNumber(1,10);
  Chart.SetValue(Point, Series, Value);
EndDo;
Chart.Refresh = True;
```

With this method of filling data, `Swap()` method is available, allowing you to swap the chart's series and points. The swap option may come in handy when changing the chart type. For example, a pie chart displays values represented as a single point and a set of series. To show the same data on the graph, one needs to swap the series and the points. `Autotransposition` property enables a mode in which the need for swapping is automatically detected, depending on the chart type and the data structure.
The second method of filling in the chart data is associated with the use of the `DataSource` property. As a source, a table of values or a cell range of a spreadsheet document can be used.

```
Query = New Query;
Query.Text =
  "SELECT
  | SalesTurnover.Period As Period,
  | SalesTurnover.TurnoverAmount AS Revenue
  |FROM
  | AccumulationRegister.Sales.Turnover(, , Month, ) AS SalesTurnover
  |
  |ORDER BY
  | Period";

QueryResult = Query.Execute();
SourceDataTable = QueryResult.Export();
Chart.SeriesInRows = False;
Chart.DataSource = SourceDataTable;
```

If when generating a series or a point its key value is set, then one can later search for this series and point. The key value is contained in the `Value` property:

```
Point.Value = 1;
Series.Value = "Series1";
```

The `SetPoint()` / `SetSeries()` methods allow to find a chart point or series by key value. Having defined a series and a point, one can read the current value at a point by using the `GetValue()` method or set a new value by using the `SetValue()` method.

```
// Search for a point
Point = Chart.SetPoint(4);
// Search for a series
Series = Chart.SetSeries("Series1");
// Detecting the chart's point value
Value = Chart.GetValue(Point, Series).Value;
// Changing the value of a chart's point
Chart.SetValue(Point, Series, Value + 10);
```

`SetPoint()` / `SetSeries()` methods generate a new chart point (chart series) once a point (series) with the specified key value is not found. For a new series, the `Value` property is to be filled. This feature is useful when filling a chart with data.

Suppose one needs to visualize the sales data for various items on a monthly basis. Items form chart series, and periods are represented by points. When iterating the data received from a query, a link to the same item can encounter multiple times if sales of this item took place in different months. In this case, it is convenient to use the `SetSeries()` / `SetPoint()` methods with key values to avoid multiple generation of the same chart components:

```
While Selection.Next() Loop
  Series = Chart.SetSeries(Selection.Item);
  Point = Chart.SetPoint(Selection.Period);
  Chart.SetValue(Point, Series, Selection.Revenue);
EndDo;
```

When displaying a diagram the situations of insufficiency of space for the convenient display of all data are possible. The graph of sales split by months for several years can be an example. Only 1 year (12 months) is displayed normally. The convenient scrolling of displayed data is necessary. In this case, it is possible to place the graph scrolling elements on the form and implement the graph movement in the required directions in the handlers as follows:

1. Delete series/point which goes out of scope. The method `Delete()` collections `DiagramSeries`/`DiagramPoints` is used for it.
2. Insert series/point which comes into scope. The method `Insert()`/`Add()` collections `DiagramSeries`/`DiagramPoints` is used for it.

It is also possible to change the position of the series/points in the relative collection. Such feature can be useful if it is required to interactively change the order of sequence as compared to the order set when the diagram was filled in. The position of series/points is changed with the help of the method `Move()` collection `DiagramSeries`/`DiagramPoints`.

Delimitation of the number of displayed series

The number of series used in the chart may be limited by the `MaxSeries` property. The data of those series that are not displayed on the chart due to the limitation, is summed up and shown in a separate summary series. The limit on the number of series is set by the `MaxSeriesCount`or `MaxSeriesPercent` properties, depending on the value of the `MaxSeries` property.
For instance, it is required to visualize the change in sales volumes to several customers on a graph. However, the revenue from two customers significantly exceeds the revenue from the rest. In this case, the number of the chart series can be limited to three:

```
Chart.MaxSeries = MaxSeries.Limited;
Chart.MaxSeriesCount = 3;
```

For each of ten series, the maximum value for all points will be determined, and only two series with the largest values will be displayed on the graph. All the rest data will be displayed in the form of a single summary series.

Chart areas configuration

On the whole, the `Chart` object is intended for operations with charts. To configure the properties of individual chart areas, use its `TitleArea`,

LegendArea, and PlotAreaproperties, as well as the ShowTitle and ShowLegend properties.
The properties of the ChartPlotArea object allow to customize the display of the chart scale, data table, and output values. Appearance of the title area and legend area is configured by using the properties of ChartTitleArea and ChartLegendArea objects.
For example, the value scale format can be specified as follows:

```
Chart.PlotArea.ValueScaleFormat = "NDF=2; NDS=.;NGS=' '; NG=3,0";
```

The value scale format, as well as other properties, can be specified by using the lotting area property palette.

Spline Mode

The method of plotting the connecting line of the graph depends on the values of the SplineMode and SplineStrain properties.
By default, the values in the chart plotting area are connected by straight lines. In the SmoothCurve spline mode, "smooth curves" are used for connecting values.

Trendlines

A trend line can be added to a series like this:

```
Series = Chart.Series.Add();
TrendLine = Series.TrendLines.Add();
TrendLine.ApproximationType =
ChartTrendLineApproximationType.Exponential;
TrendLine.Marker = ChartMarkerType.None;
```

To add a trendline, use the special ChartTrendlineobject. For a single series, multiple trend lines can be generated at once.
The following parameters can be specified for a trend line:

- Trend line color (Color property).
- Whether or not the trend line should be displayed in the chart legend (ShowInLegend property).
- Whether or not the actual equation used for the plotting of the trend line should be displayed (ShowEquation property).
- Configure the equation display mode, if it is displayed (EquationArea property), and so on.

Setting semitransparency

For certain types of charts, the SemiTransparencyMode and SemiTransparencyPercent properties are implemented, which allows you to adjust the mode and degree of the chart transparency.

If the `SemiTransparencyMode` chart property is set to `Use` mode, then the chart transparency is specified by the value of the `SemiTransparencyPercent` property. If the semitransparency percent is set to zero, then the default transparency percent for this type of chart is used.
In `Auto` mode, transparency is automatically set based on the value of the `SemiTransparencyPercent` property, chart type, and other factors.

Using a single color for a series

When you create several charts which use a single content of series (for instance, charts related to sales), it is advisable to use one and the same color for a single object in all charts. Thus, a chart is better perceived by its users.
Reports generated in a data composition system support this by default. If a chart is based on a form, you need to perform certain extra operations. Let's have a closer look at these operations.
The platform supports a dedicated `Value Appearance` feature which stores data as to whether a series value and color used to display the same are consistent, or not. The above object stores consistency data for all kinds of objects involved in generation of charts for the current user. In other words, if a configuration has charts which use warehouses, counterparties and goods, - `ValueAppearance` object will store consistency data for all objects listed hereinabove. This object stores max 100 most frequently used objects per each value kind (and related colors). Thus, if an application supports charts where series are related to goods, settings will store color data for max 100 goods that are most frequently used to generate charts. `ValueAppearance` can be stored in the value storage. Data composition system uses General/ChartColors for that purpose.
The scheme of operation is as follows:

- The chart is filled in with respective data.
- Information as to consistency of objects and colors is retrieved from the storage. For instance, you can use system value storage: `SeriesColors = SystemSettingsStorage.Load("General/ChartColors");`.
- Data so retrieved is loaded into a chart: `Chart.FillInSeriesColors(SeriesColors);`. Colors will be filled in for series with Auto value set.
  If a color from the current palette, which is already used in this chart, is obtained during the filling of the series of colors from the saved colors, then it will be replaced with a lighter or darker color (depending on the color itself and other colors already used). The resulting color will be used for series. If color conflicts occur for the next series, the color is again replaced with a lighter or darker one, taking into account the previous colors.
- Further, it is recommended to save color consistency data in the storage again:

`SystemSettingsStorage.Save("Common/ChartColors", , SeriesColors);`. It is recommended to save data upon generation of a chart, since it can have objects which have not been yet involved in the chart generation process. Thus, another color is assigned to a new object by the chart by default. Make sure that you save this data in the storage for further use.

If a form enables you to modify the chart series color, it is recommended to do it as follows:

- Get color and `ValueAppearance` object consistency data.
- Assign a color to a series using `ChartSeries.Color` property.
- Edit the color in the color and object consistency data set using `ValueAppearance.Set(Object,Color)`.
- Save color and object consistency data in the storage.

Event handling

For the chart field, event handling is implemented. The `Select` event occurs when a chart component is selected. The `DetailProcessing` event occurs when a chart component is selected, provided that the `Detail` property is set for this component.

With standard processing, the detail value is displayed in a special window. For example, in the `Detail` property of a chart series, a link to the `Customers` catalog item can be stored:

```
Series = Chart.Series.Add();
Series.Detail = DataTableString.Customer;
```

In this case, a form of the corresponding catalog item will be opened by double-clicking on the series name.

There is an option to implement a custom "detailing" for the selected item. For instance, to show a list of all sales documents for the selected customer. For this, the `DetailProcessing` event can be handled, for example, like this:

```
&OnClient
Procedure ChartDetailProcessing (Item, Detail, StandardProcessing)
  StandardProcessing = False;
  FormParameters = New Structure;
  // Transferring the Customers catalog item link
  // in the DocumentsByCustomer shared form.
  FormParameters.Insert("Customer", Detail);
  OpenForm("SharedForm.DocumentsByCustomer", FormParameters);
EndProcedure
```

Similarly, detailing can be specified and processed for a chart point or chart value.

The selection in the diagram is processed with the help of the event `OnActivate`. To get the selected areas of the diagram, use the method

`GetSelectedItems()`. This method is accessible for the form field extension for the diagram field (describes the diagram placed in the form) and for the object `SpreadsheetDocumentDrawing` (describes the diagram placed in a tabular document). This method will return the array containing the selected data. The subsequent behavior depends on the application solution.

```
&OnClient
Procedure DiagramOnActivate(Item)
  Data = Item.GetSelectedItems();
EndProcedure
```

7.3.4.3.3. Charting-specific features

Displaying points in stacked charts

In stacked charts, the following algorithm is utilized to determine how each data point will be displayed:

- Charts supporting the base value:
  - Two sums are calculated for each point: more and less than the base value.
  - Before accumulation, the base value is subtracted from the current value.
  - The acquired value is being accumulated by absolute value in the corresponding sum (depending on the sign).
- Charts not supporting the base value:
  - Two sums are calculated for each point: more and less than 0.
  - The current value is being accumulated by absolute value in the corresponding sum (depending on the sign).

Two chart values are displayed for each point: The cumulative sum of positive and negative values (of the base value or value 0).
This calculation is applied to the following charts:

- Various types of histograms with accumulation;
- A chart with areas;
- A chart with areas and accumulations;
- A radar chart with accumulations;
- A radar chart with areas and accumulations.

Skipping values in charts

Data displayed by the chart may not necessarily contain values in each point of the displaying series. In these cases, it is necessary to control how the chart will display such unspecified (non-numeric) values. The following chart properties Using non-numeric values and The type of connection for points with missing values are used to control the display of non-numeric values.

The first property specifies how the chart will interpret non-numeric values:

- As zero - In this case, the non-numeric values will be equivalent to 0.
- Ignore - In this case, the non-numeric values will be skipped when building a chart.
- Auto - A non-numeric value is considered equivalent to 0.

If non-numeric values will be interpreted by the chart as to be ignored, then by using the property Connect type of points with missing values it is possible to set the chart display mode for such values (this property controls the display of only those charts in which the adjacent points in the series are graphically connected to each other):

- Do not connect - The points between which there are non-numeric values, are not connected.
- Connect - The points between which there are non-numeric values are connected directly.
- Connect with the base - Is similar to the value Do not connect. A difference is that the point after (or before) of which there is a non-numeric value, is connected to the base value of the chart but not to another point of the series. Besides, a connection is performed at the same point.
- Auto - is interpreted as Do not connect.

A display of non-numeric values has a number of features depending on the type of chart:

- For radar graphs, a connection with the base value is interpreted as connection to the center of graph.
- For surface-charts, the drawing algorithm is similar to the one described above; but at connecting not missed, four adjacent points are taken into account.
- For the following chart types, a point for which a non-numeric value is indicated is not displayed:
  - A histogram of any kind (without accumulation);
  - A stock chart (any) - element is not drawn if at least one of the four values used is missing;
  - Isometric;
  - Gauge;
  - Bubble.
- For all normalized charts as well as the ones with accumulations, a non-numeric value is interpreted as zero.

7.3.4.3.4. Data for charts

To demonstrate different types of charts, the following data set is used:

	Wholesale buyers	Retail buyers	Dealers

14.03	6	8	6
15.03	5	9	4
16.03	9	8	6
17.03	5	4	7

Depending on the type of chart, the series can be arranged both in rows and in columns.
If a chart is built on a different data set, this data set is shown directly in the section where the chart is shown. As a rule, in charts (as the examples below show) the checkbox Series in rows is disabled.

7.3.4.3.5. Chart

A chart uses the display method in which the adjacent values within a series are connected by a line.

7.3.4.3.6. Step chart

A difference between step charts and usual charts is that in the first case the adjacent values are connected by steps instead of straight lines.
It is convenient to use a chart of this kind to emphasize the intermittent nature of change in reference to a value. For example, a change in stock balances at the end of the business day.

7.3.4.3.7. A chart with accumulation

A peculiarity of charts with accumulation is that each displayed value takes into account the values of previous series at the same point. Convenient when there is a need to evaluate the contribution of each series to the total amount. For example, a change in the contribution of each group of customers to the overall amount of revenue can be traced.

7.3.4.3.8. A chart with areas

In a chart with areas, an area between the line of chart and the axis of base value appears as shaded.

7.3.4.3.9. A chart with areas and accumulation

A chart with areas is similar to a chart with accumulation; the areas between the chart lines and axis are shaded.

7.3.4.3.10. A normalized chart with areas

A normalized chart displays contribution of each series to the overall amount of values at a point. For example, at the first point, the values of series 6, 8, and 6. The overall amount is 20, a contribution of the first series will be 30%, the second- - 40%, the third- - 30%.

7.3.4.3.11. A histogram, volumetric histogram, horizontal histogram, horizontal volume histogram

A histogram is used to visualize the same data as a chart but uses a different way to display these. Data on the histogram are shown in the form of rectangles, the sizes of which allow to compare values of different series at one point.

7.3.4.3.12. A histogram with accumulation, volumetric histogram, horizontal histogram, horizontal volume histogram

A histogram with accumulation as an accumulated chart (see above) allows to include the values of all previous series in the value of each series.

It is necessary to take into account that the histograms with accumulation are able to connect the adjacent values within series in order to facilitate reading of the displayed data:

```
Chart.ValuesBySeriesConnection =
ChartValuesBySeriesConnectionType.EdgesConnection;
Chart.ValuesBySeriesConnectionLines = New
Line(ChartLineType.DottedLine, 2);
Chart.ValuesBySeriesConnectionColor = WebColors.Red;
```

A series of histogram with accumulation can also be marked with the `Indicator` property. This series is shown as a chart and is not taken into account when creating the consolidated series.

7.3.4.3.13. A normalized histogram, a volumetric, horizontal, horizontal volume histogram

A normalized histogram, as a normalized chart, displays contribution of each series to the overall amount of values at a point.

For normalized charts, a combination of values within a series is available.

7.3.4.3.14. Pie, pie volumetric

This type of chart allows for showing the share of an individual value in the overall amount of values. A pie chart is based on the data of a singular point and a set of series.

If several points are present in a pie chart data, then the values of the point specified in the `ActivePoint` property of the chart will be displayed.

7.3.4.3.15. Doughnut, 3D Doughnut

As well as the pie chart, the doughnut chart shows the share of a separate value in the total sum of values but it may include more than one point in each series. Each point in the series shown in the doughnut chart adds a ring to it. The first point of the data is displayed in the chart center.

Donut chart inner radius can be managed with the help of the chart property `DonutChartInnerRadius`.

7.3.4.3.16. Stock, stock candle

Stock chart allows for showing at one point four values characterizing the change in the indicator over a period of time. A set of indicators includes the highest value of an indicator for a given period; the lowest value as well as the maximum and minimum values of an indicator for a given period.
A stock chart shows data of a single series. Herewith a value of the first point is used as initial value of an indicator, a value of the second point - - a minimum value, of the third point - - maximum value, of the fourth point - - at the end of period.
Thus, data of the first four points are displayed as a single point of the chart. The name of the first point of a group is used as the name of the merged point. Similarly, data of the following four points are combined.

A stock chart may contain data of several series. In this case, the values of the series specified in the `ActiveSeries` property of a chart are displayed.

7.3.4.3.17. Isometric plain, continuous, tape, pyramid, waterfall

An usual isometric chart is similar to a graphic chart; however, the values within the series are shown as rectangular parallelepipeds of various heights. Allows for displaying the dynamics of an indicator broken down by various factors.

An isometric pyramid shows values as kind of pyramids of various heights. Isometric tape displays the change in values of the series as kind of ribbon. A continuous isometric chart is an isometric ribbon with a filled area under it. A waterfall - is a kind of isometric chart, in which the series do not have thickness.
For charts of the Isometric continuous, Isometric Ribbon and Waterfall kinds, properties of the `SplineMode` and `SplineStrain` are permissible.
When the smoothing mode is disabled for a chart of the Isometric Ribbon kind, the property `Indicator` of the chart series can be used. A series having the flag `Indicator` is displayed semi-transparently forming intersections with other series.

7.3.4.3.18. Surface, frame, convex, concave , shaded, honeycomb surfaces

Surface charts make it possible not only to observe a change in value within the series but also to trace the dependence of a value on the series at one point.

7.3.4.3.19. A radar chart, with areas, accumulation, with areas and accumulation, with areas normalized

A radar chart uses a radar scale to display data. Points are located on a circle, values are indicated by points located at different distances from the center of the circle.

Assignment of the radar charts of other kinds is similar to the corresponding cases with a chart of the `Chart` kind.

7.3.4.3.20. Gauge

A gauge chart is formed in the same way as the pie charts are. A distinctive feature of a gauge diagram is - in displaying the scale as a gauging panel.

Using a gauge panel allows for visually attributing a value of the indicator to a specific group.

A selection of value ranges is made by adding bands of a gauging panel. A collection of bands is available through the `GaugeChartQualityBands` property:

```
Band = Chart.GaugeChartQualityBands.Add();
Band.Beginning = 5;
Band.End = 10;
Band.BackColor = WebColors.Green;
Band.Text = "Good"
```

A gauge chart can display values in two ways: using an arrow or in the form of sectors. A required method is specified in the property `GaugeChartValueRepresentation`.

7.3.4.3.21. A funnel, volumetric, normalized, normalized volumetric

A view of the Funnel chart is similar to a horizontal histogram. Values are displayed as segments forming a funnel shape with a corresponding set of data values.
For the Funnel and Funnel volumetric charts, their values are displayed by a width of the funnel segment. In the Funnel normalized and Funnel normalized volumetric charts, their values are displayed by a height of the funnel segment.

A funnel design is affected by values of the following chart properties: `FunnelSpace`, `FunnelNeckHeight` and `FunnelNeckWidth`.
For charts of the Funnel and Funnel volumetric kinds, it is possible to use the properties `SplineMode` and `SplineStrain`.

7.3.4.3.22. Scatter chart

A scatter chart represents values from different gauges, by a single dimension in the form of collection of points. In most charts, a gauging is located along one of the axes; however, in a scatter chart it is represented by points on chart while the gauges are on each of the two axes.

This chart is based on the following dataset:

	August	September	October	November	December

Skis	90	330	610	1 050	1 380
Ski boots	40	280	290	830	970
Ski gloves	20	210	610	730	810
Bicycles	700	540	100	50	20

What data to be stored along the chart axes is determined by the Source of the axis of points property:

- Series - In this case, the series which will be deposited on the axis of points is indicated in the Series on the axis points property.
- Points - In this case, the point values are deposited along the point axis.
- Auto - In this case:
  - A series one - The points are deposited on the axis points.
  - series more than one -Behavior is similar to explicitly specifying the value Series.

An example **Ошибка! Источник ссылки не найден.** is displayed with the following settings:

- A flag Series in rows is set;
- The property Source of the point axis value is set to Point.

If the chart settings are slightly changed, namely:

- The property Source of the point axis value is set to Series.
- The property Series on the point axis is set to the Ski value.
- For the Bicycle series, the trend band has been added given the linear equation.

In this case, it is clear that the sales of skis and bicycles depend on each other inversely: the more skis, the fewer bicycles are sold. To increase visibility of this conclusion, a trend band has been superimposed on the sales of bikes.

7.3.4.3.23. A bubble chart

A bubble chart -is a variation of the scatter chart where a point has a variable rather than a fixed size. Besides, a size of the point (bubble) is the third dimension for such a chart. Thus, a bubble chart allows for displaying data in three dimensions on a plane.

A chart is constructed on the following dataset:

Product	Sales	A market share
Boots	5 500	3
Pants	12 200	12
Suit	60 000	33
Blazer	24 400	10
Shirt	32 000	42

Similarly to the scatter chart, data is assigned to the chart axis using the Source of the point axis value property. In this case, the system logic completely coincides with that for the scatter chart.
For the bubble chart, note the following properties:

- The source of the bubble size value - determines where to get a value determining the bubble size on a chart:
  - No - In this case a behavior of the chart is completely equivalent to that of the scatter chart.
  - The overall series - Indicates the series whose values will be used as the bubble size. A series specification is performed using the Overall bubble size series property.
  - The next series - In this case a bubble size will be determined by the series following the series with data. For example **Ошибка! Источник ссылки не найден.**, a bubble size is determined by the Market share column.
- Displaying negative values of bubble chart - determines how the chart will “react” to negative values in data:
  - No (default value) - If the value is less than zero, then the bubble is not displayed.
  - On module - All the values are taken into consideration based on module
  - Transparent background - For calculating the size of a bubble all values are used based on module but bubbles with values less than zero will not be colored.
  - Inverted background color - In this case, a bubble will be colored with a color which is the inverse of the current series color.
- Size effect on a bubble - Describes how the value describing the size of a bubble will affect the size:
  - Increasing / decreasing diameter - In this case, the bubble's diameter will change directly / inversely proportional to the size value.
  - Increasing / decreasing the area - In this case, the area of a bubble will change directly / inversely proportional to the size value.
  - In any case, the bubble size will be limited by the properties Maximum Bubble Size and Minimum Bubble Size.

7.3.4.4. Gantt Chart

7.3.4.4.1. Overall information

The Gantt chart is a way of reflecting the duration and sequence of processes that are shown as bands deposited along the time axis.

In the chart area the construction, title and legend areas are displayed by default. For setting properties and managing visibility of the areas, the following properties are intended: `TitleArea`, `LegendArea`, `PlotArea`, `ShowTitle` and `ShowLegend`.
In the chart construction area, the time scale is located horizontally while the vertical scale of points (processes)- is located vertically.

7.3.4.4.2. Adding data

To control a composition of series and points, use the chart properties `Series` and `Points`.
To work with the chart values, use the `GanttChartValue` object which can be acquired by using the `GetValue()` method. A value provides an access to the spacing collection.
A value of the Gantt chart is displayed as one or more spacings. Consider an example:

```
// Adding series.
SeriesPlan = GanttChart.Series.Add();
SeriesPlan.Text = "Plan";
// Adding a point.
PointOfficeConstruction = GanttChart.Points.Add();
PointOfficeConstruction.Text = "Office construction ";
// Adding a value.
Value = GanttChart.GetValue(PointOfficeConstruction, SeriesPlan);
// Adding a spacing.
Spacing = Value.Add();
Spacing.Beginning = '20130125';
Spacing.End = EndOfDay('20130206');
```

A Gantt chart construed in this way displays a single spacing showing the duration of process Building office.

Points and series of the Gantt chart can form a hierarchical structure. Example:

```
// Creating points
PointPlanning = PointOffice Construction.Points.Add();
PointPlanning.Text = "Construction planning";

PointPurchasing = PointOfficeConstruction.Points. Add();
PointPurchase.Text = "Purchasing materials";

PointClearing = PointOfficeConstruction.Points.Add();
PointClearing.Text = "Clearing the area";

// Creating spacings
Value = GanttChart.GetValue(PointPlanning, Series/Plan);
SpacingPlanningPlan = Value.Add();
SpacingPlanningPlan.Beginning = '20130125';
SpacingPlanningPlan.End = EndOfDay('20130201');

Value = GanttChart.GetValue(PointPurchase, SeriesPlan);
SpacingPurchasePlan = Value.Add();
SpacingPurchasePlan.Beginning = '20130202';
SpacingPurchasePlan.End = EndOfDay('20130204');

Value = GanttChart.GetValue(PointClearing, SeriesPlan);
SpacingClearingPlan = Value.Add();
SpacingClearingPlan.Beginning = '20130128';
SpacingClearingPlan.End = EndOfDay('20130206');

// Display setting
GanttChart.ExpandPoint (Point Construction Office, True);
```

As a result, a chart will show the duration of the three stages of the construction process as well as the overall duration of construction.

To fill the Gantt chart with data, it is convenient to use the methods `SetPoint()` and `SetSeries()`. The method `SetPoint()` (`SetSeries()`) returns a point (series) of the chart with a specified key value in the `Value` property; for example, a reference to a catalog item. If the point (series) with such a key value is not found, then a new point (series) is created and the `Value` property is filled.

NOTE. A second parameter is able to pass a key value of the parent point (series) to the methods `SetPoint()` and `SetSeries()`. If this parameter is not specified, then the existing point (series) with the empty `Value` property can be automatically set as a parent to the point (series) being created.

7.3.4.4.3. The spacing connections

The Gantt chart spacings can be connected if it is necessary to reflect the fact of dependence of processes. The object `GanttChartInterval` allows for adding elements to the collection of the chart connections:

```
Connection = SpacingPlanningPlan.Add(SpacingPurchasePlan);
```

By default, a connection is displayed as a line directed from the end of the first spacing to the beginning of the second. If necessary, a type of connection can be changed in the property `LinkType`.

7.3.4.4.4. Specifying time scale settings

A full time chart spacing is automatically determined by the duration of all the displayed spacings if the property `AutoDetectWholeInterval` chart property contains the `True` value. When turning off automatic detection, the `SetWholeInterval()` method should be used:

```
GanttChart.AutoDetectWholeInterval = False;
GanttChart.SetWholeInterval('20150124', '20150209');
```

A time scale can be supplemented with new lines, for each of which the frequency, period display format and other properties can be specified. Adding a new time scale to display weekly marks looks, as follows:

```
WeeklyScale = GanttChart.PlotArea.TimeScale.Items.Insert(0);
WeeklyScale.Unit = TimeScaleUnitType.Week;
WeeklyScale.UnitConversionFactor = 1;
```

It should be taken into consideration that heading of the scale item in Gantt chart is displayed only at the beginning of the corresponding division. If the beginning of division is not displayed (it turned out to be “to the left” of the displayed range), then the title will not be displayed on the chart.
The labels placeable on the time scale can be accessed through the collection `Labels` of the time scale item. However, it must be remembered that the added label will be displayed inside the time scale, for which a label was added. Let's consider an example of adding a label to the previously added weekly scale:

```
Label = ScaleItem.Labels.Add('20150204');
Label.Text = "Control";
Label.LineColor = WebColors.Red;
```

Such placement of a label reduces visibility of its perception and of time by the selected scale. To improve perception, a "fictitious" scale can be added which won't have any periodic labels; a label can be added specifically for this spacing (to the contrary of previous example):

```
FictitiousScale = GanttChart.PlotArea.TimeScale.Items.Add();
FictitiousScale.ShowPeriodicalLabels = False;
Label = FictitiousScale.Labels.Add('20150204');
Label.Text = "Control";
Label.LineColor = WebColors.Red;
```

It is possible to change the background color of a chart fragment for a specified period. For example, weekend can be selected:

```
BackgroundSpacing = GanttChart.BackgroundIntervals.Add('20150124',
EndOfDay('20150125'));
BackgroundSpacing.Color = WebColors.BlanchedAlmond;
BackgroundInterval = GanttChart.BackgroundIntervals.Add('20150131',
EndOfDay('20150201'));
BackgroundSpacing.Color = WebColors.BlanchedAlmond;
```

7.3.4.4.5. Interactive spacing editing

These Gantt charts can be modified both within the 1C:Enterprise language and interactively. In order for a user to have an opportunity to change both location and size of spacings, the `Edit` property of the chart value needs to be used.

As an example, add the Fact series to the chart data and allow editing its values at the point Clearing territory:

```
SeriesFact = GanttChart.Series.Add();
SeriesFact.Text = "Fact";
Value = GanttChart.GetValue(PointOfficeConstruction, SeriesFact);
Spacing = Value.Add();
Spacing.Beginning = '20150125';
Spacing.End = EndOfDay('20150206');
Value = GanttChart.GetValue(PointClearing,SeriesFact);
// The interactive spacing editing is allowed.
Value.Editing = True;
Spacing = Value.Add();
Spacing.Beginning = '20150127';
Spacing.End = EndOfDay('20150206');
```

A spacing will be marked (as in the figure) after trying to change a position of spacing with mouse.

7.3.4.4.6. Event handling

Interactively changing spacings in the Gantt chart would lead to the event `OnIntervalEditEnd` to emerge. Properties of the modified interval are available in the event handler.

For example, in the event handler it is possible to check the output of the spacing Clearing area beyond full chart spacing and a refusal to edit if a duration of the factual data gets beyond the full spacing:

```
&OnClient
ProcedureGanttChartAtEndingSpacingEditing(Item, Spacing, CancelEdit)

  If Spacing.End > GanttChart.EndOfWholeInterval Then
    CancelEdit = True;
  EndIf;

EndProcedure
```

7.3.4.5. Dendrogram

A dendrogram is a graphical way of displaying a degree of similarity of objects. The dendrogram area contains both construction and header areas. By default, the comparing objects are listed in the construction area on the horizontal axis while a difference between the objects is gauged vertically.

A location of the connection line shows the difference in values of items.
As an example, let's present differences in prices of two products in the dendrogram: apples - 10 rubles, pears - 12 rubles.

This dendrogram of the 1C:Enterprise language can be construed, as follows:

```
Apples = Dendrogram.Items.Add();
Apples.Text = "Apples (2 USD)";
Pears (Груши) = Dendrogram.Items.Add();
Pears.Text (Груши.Текст) = "Pears (Груши)(12 Rb.)"
Connection (Связь) = Dendrogram.Connections.Add();
Connection.FirstItem = Apples;
Connection.SecondItem = Pears;
Connection.Distance = 2;
```

The Dendrogram collection `Elements` contains a list of objects, in our case - goods. Connections between objects are defined in the `Connection` collection.
Now let's add a third product to the comparison: Oranges, valued at 20 Rb. For this, one more element of dendrogram is to be created and one more connection is to be established:

```
Oranges = Dendrogram.Items.Add();
Oranges.Text = "Oranges (2 USD)"
Connection (Связь) = Dendrogram.Connections.Add();
Connection.FirstItem = Apples;
Connection.SecondItem = Oranges;
Connection.Distance = 8;
```

It should be taken into consideration that the arrangement of connections corresponds to the order of their addition. Adding the first connection led to a grouping of the items Apples and Pears. At creating the second connection, a presence of the previously created connection is automatically taken into account, so the second connection links the item Oranges with the item formed by connecting the items Apples and Pears.

Configuring areas of a dendrogram can be done using the properties `TitleArea` and `PlotArea`. For example, a format of the scale of values can be changed:

```
Dendrogram.PlotArea.ValueScaleFormat = "CDC=2;";
```

A mode of automatic adjustment of the scale of a dendrogram when resizing form is determined by the `ScaleKeeping` property:

- `AllItems` -All items are displayed in the dendrogram area regardless of the form size.
- `ItemCount` - In the visible part of dendrogram a number of items specified in the `ScaleItemCount` property is displayed. Remaining items are available when using the scroll bar.
- `MinimumWidth` - By reducing a size of form the sizes of the items can reduce to the width specified in the `ItemMinWidth` If there is not enough space in the visible part to display all the items, a scroll bar is used.

7.4. Form parameters

7.4.1. General information

Form parameters (bookmark Parameters) serve two purposes:

- Describe the data set that will affect opening of the form (parameterization). To do this, list all the necessary parameters and specify their kinds.
- Determine the parameters that will affect a key of the uniqueness of form. For this, it is necessary to set the Key parameter property for those parameters that should participate in formation of the form uniqueness key. When trying to open a form, the system searches for an existing form using the previously generated form uniqueness key. If there is a form in the system with the received uniqueness key, then this namely form is returned; if not -, a new form is created.

When calling the form, the values of the parameters created by a developer can be specified in the parameters structure along with the system parameters of forms (if any).

Form parameters can be transferred to a form at the time of its creation. An analysis of the transferred parameters can be performed in the event `OnCreateAtServer()` (collection `Parameters` is a property of the `ClientApplicationForm` object):

```
// In the place of calling.

// Forming a form parameter.
Parameters = New Structure();
Parameters.Insert("Importance",
PredefinedValue("Enumeration.Importance.Important"));

// Opening the form specifying parameters.
OpenForm("GeneralForm.ReviewForm", Parameters);

…

// In form module.
&OnServer
The OnCreateAtServer(Denial, StandardProcessing) procedure

 If Parameters.Importance = Enumerations.Importance.Important Then
 …
 EndIf;

EndProcedure
```

IMPORTANT! After calling the Event handler of the event `OnCreateAtServer` all the non-key parameters of a form are removed from the collection `Parameters`.

TIP. The non-key parameters of a form necessary for further work are to be stored in the form data.

Working with the form parameters the user shall understand that the behavior of the parameters can be considered equivalent to the system behavior if the parameter value is not directly specified (missed) upon the embedded language call. In his case, the parameter value will be set as `Undefined` (or default value). The form parameters behave similarly: if the parameter is set when the form is opened in the client application then this parameter will be present in the form parameter structure with a set value. If the parameter is not specified when the form is opened then the platform will automatically add the parameter in the structure of parameters but the value of this parameter will be `Undefined`. It is equal to the parameter missing upon the method call. The behavior described above is the same both for the parameters created by the developer and the standard parameters of the form.

7.4.2. The standard parameters of a form

In order to support the automatic interaction between forms, the system provides a number of standard parameters that are used to manage forms at their opening. Using these parameters, the system implements a selection in the form areas from the selection forms, opening of object forms, operation of the standard commands, etc. That is, they provide various scenarios for operation of interface embedded in the system. However, a developer can also use these parameters in the 1C:Enterprise language passing them when calling the `OpenForm (ОткрытьФорму) ()` method.

A list of the standard form parameters depending on the type of form extension can be viewed in the sections of the 1C:Enterprise language - Interface (managed) - Managed form - Expansion... inline reference.

7.4.3. Example of working with the form parameters

To demonstrate how the form parameters work, let's consider the implementation of selecting an item in the input field. The essence of this example will be implementation of the mechanism for selecting an item from a list in the 1C:Enterprise language

By the time of starting working with this example, a configuration that has the following properties is needed:

- There is a catalog`Goods` containing a hierarchy of groups and items;
- There is a catalog `Analogues` with attribute `SelectedGoods` of the kind `CatalogRef.Goods`;
- Both catalogs possess the form items.

Now let's implement in this configuration all the mechanisms that the platform uses to select an item from the list in the 1C:Enterprise language. In this case, we will see:

- how the standard form parameters are being used;
- in what way the system uses them;
- how a developer can use them;

Let's add an additional parameter that will control the selection form closure after an item is selected. Let's call that parameter `CloseAfterSelection` (type `Boolean`) Let's add it as a parameter to the form `ChoiceForm` in the catalog of `Goods`.

In order to open the item selection form, it is necessary to create in the item form catalog `Analogs` an event handler for the event `StartChoice`at the form item `SelectedGoods`.

```
&OnClient
Procedure Selected Product Commencement Selection (Item,
StandardProcessing)

  StandardProcessing = False;

  ParameterStructure = New Structure;
  ParameterStructure.Insert("ChoiceMode", True);
  ParameterStructure.Insert ("ChoiceFoldersAndItems",
FoldersAndItemsUse.Items);
  ParameterStructure.Insert(AllowRootChoice", False);
  ParameterStructure.Insert("CurrentRow", Object.SelectedGoods);
  ParameterStructure.Insert("CloseAfterSelection", False);

  OpenForm ("Catalog. Goods. ChoiceForm", Parameter Structure, Items.
SelectedItem);

EndProcedure
```

It is necessary to focus separately on the third parameter of the `OpenForm()` method. This parameter determines who will be the owner of the selection form and who will be notified about the choice made. In this case, we have specified as the owner of a selection form the form item itself but we can also specify by this parameter the form itself. In this case, it will be necessary to implement the event handler of the `ChoiceProcessing` form module to and decide in it, in which form attribute to place the selected data.

NOTE. If we do not implement the event handler for the `StartChoice` event, then the system itself will execute its actions. This is true for all further event handlers used in this example.

Now it is necessary to process the transferred parameters in the form of a choice. Let's do this in the handler of the `OnCreateAtServer()` module of the selection form.

```
&OnServer
The OnCreateAtServer(Denial, StandardProcessing) procedure

  StandardProcessing = False;

  Items.List.ChoiceFoldersAndItems = Parameters.ChoiceFoldersAndItems;
  Items.List.AllowRootChoice = Parameters.AllowRootChoice;
  Items.List.CurrentRow = Parameters.CurrentRow;

  CloseOnChoice = Parameters.CloseAfterSelection;

EndProcedure
```

In order to check the operability of the form parameters we had set, let's set, using the configurator, at the form selection table `List` the property `ChoiceFoldersAndItems` to the value `Groups` (without using the parameter a selection of the catalog's items will not be available).

NOTE. If in the `List` table displaying a list of goods the property `ChoiceMode` is not set to `True`, then a selection of goods will not be available.

Now we need to process a selection of the desired item in the form of selection. To do this, there is a need to define an event handler for the event `ValueChoice`of the form table.

```
&OnClient
Procedure ListSelectionValues(Item, StandardProcessing, Value)

  StandardProcessing = False;
  NotifyChoice(Value);

EndProcedure
```

It remains to implement the data processor for the item selection in the input field itself. To do this, there is a need to process the event `ChoiceProcessing` of our input field `SelectedGoods`.

```
&OnClient
Procedure SelectedGoodsSelectionProcessing(Item, ValueSelected,
StandardProcessing)

  StandardProcessing = False;
  Object.SelectedGoods = ValueSelected;

EndProcedure
```

We have independently implemented a system mechanism for selecting value in the input field on a form.

IMPORTANT! This example has not been completed. Its sole purpose is to demonstrate the mechanisms of working with form parameters.

If, when creating parameters (handler `Selected ProductBeginningSelection()`) to replace the string:

```
  ChoiceParameters.Insert("CloseAfterSelection", True);
```

on the string:

```
  ChoiceParameters.Add("CloseAfterSelection", False);
```

then a form of selection will stop closing after the selection is made. This can be used, for example, to implement a selection form (selection of several goods without closing a selection form).

7.5. Form commands

7.5.1. General information

The form actions are performed using form commands. The commands themselves serve only as a description of the actions performed. In order for a command to perform its function, it must be tied to a form item (such as Button)
There are several groups of commands that are present in the form:

- **Commands created by developer** in the form design process. For them, a handler needs to be created directly in the form module.
- **Standard commands** that are provided by extension of the main form attribute and extensions of attributes that are lists (for example, a table part of an object, a dynamic list, a set of notes related to the register events, etc.) if there is an associated form item with this attribute.
- **Global commands** - Commands provided by the global command interface. Such commands may be both non-parameterized and parameterized. Parameterized global commands will be provided to a form only if there are sources of parameters with the corresponding types on a form.

The availability of standard form commands and the form items is determined by the property Composition of commands of the corresponding form item.
The commands provided by a global command interface (bookmark Global commands) can be placed anywhere on the form, just like the form commands.
The Action property specifies a handler that implements the action performed by a command. If a handler is not specified, a command will not be available for use. Only procedures and functions without parameters that are client-side are available for selection in this field.
If a command changes the form data, then there is a need to indicate this by setting the property Changes the saved data. Then when trying to execute a command, the following actions will be performed:

- Attempting to lock the main form attribute. If that attempt fails -, a command will also fail.
- A sign of the change in a form is set (check box Modification).

If the property Modifies the saved data is set for a command and a form is in View Only mode, then all the form items associated with this command will also be in View Only state.
In addition, an execution of a parameterized global command, for which a parameter is provided by the attribute with the Stored Data property set, for a new and unsaved object will result in an attempt to write an object. In that case, a user will be asked a question about a need for recording. If the answer is negative - a command will fail.

NOTE. When you automatically fill in command panels and context menus for which the source of commands is specified, the standard commands are not

added if there are buttons added manually with the same commands in this item. This logic does not apply to commands added from a fragment of the global command interface.

To simplify development of various dialogs, the commands Yes, No, OK, Cancel, Redo, Stop, Ignore are added to standard form commands. If such a command is added to a form, then when a user clicks this button, the following actions will be performed:

- For a form that is opened in a modal mode, a closure of a form es executed and the corresponding value of the `DialogReturnCode` kind is returned.
- For a form opened in non-modal mode, only the closure of a form is executed.

When forming the name of a command execution handler, the Name property of a command will be used.

The standard Reference form command is used to open reference information on the configuration form / object. A location of this button on the command panel of a form is determined by the property `OnlyInAllActions` of this button. If this property is set to `Auto` (and for the auto-complete command panel), the actual location will be, as follows:

- The Taxi - interface If the form or parent configuration object has reference information, then the button will be located on the command panel. Otherwise, the button will be located only in the More (Еще) menu.
- Other interfaces - Button is always located on the command bar form.

Use.

Functional options - determines which functional options are associated with this form attribute.

7.5.2. Features of using various commands

For extensions of the managed form for an object, catalog, document, plan of characteristic types, business process, task and information register entry, there is a standard ShowIn List command which is intended to display the list form of the corresponding object with the positioning of the list being opened on the object from which the command is executed. In the interface, a command will be unavailable if the key attribute of a form is an empty reference (or empty record key), that is, a new object is being edited.

If a command is used as document, then for such a command the `Parameter` property is available, in which the form that will be used to display a list can be selected. A document itself and all the journals that include the document used are available for selection. The button associated with the `ShowInList` command is displayed on a form independently, regardless of whether the

Parameter property is full. If a value of the Parameter property is set, then as a header, by default, the button associated with the command ShowInList uses a representation of a list of the corresponding configuration object (document or journal).

When a document form the command bar is automatically filled in, the logs containing that document are analyzed while the current user should have an access View to these logs and they (logs) should be enabled with functional options. If there is at least one such log, then in the command bar instead of the ShowInList button there is a group of buttons with the ShowInList heading. Submenu contains:

- The ShowInList command for which the Parameter property is filled with a reference to a document in the form of which a command bar is created;
- The ShowInList commands (one for each relevant log) for which the Parameter property is filled with a reference to the corresponding document log.

If there are no available logs, then the command ShowInList appears in the command bar for the button of which the Parameter property is not set.

For display, the main form of the list (taking into account the possibility of overriding) of the corresponding configuration object is used.

If it is necessary to provide functionality of the ShowInListcommand within the framework of implementation of some other algorithm, then to do this use the ExecuteNavigation() method that exists for the ClientApplicationForm object.

Example of implementation:

```
Form = GetForm("Catalog.Counterparties.FormList");
Form.ExecuteNavigation(Object.Reference);
```

If it is necessary to somehow process the opening of a form with positioning on the desired object, then this can be done with help of the event handler of the event NavigationProcessing (the event is called for the managed form). If after execution of the event handler for the event NavigationProcessing an execution of the standard is required ((the value of the StandardProcessing parameter is equal to the True value)), then for the first table of the managed form associated with the main form attribute of the DynamicList kind the current string is set to the value passed as a parameter Object of the event handler. If the main attribute of the form being opened is a non-dynamic list -, then no actions will be performed.

7.6. Form module

A form module consists of a set of procedures and functions. The variables and module body are allowed.
Each procedure, function or declaration of a form module variable must be preceded by one of the following compilation directives:

- `OnClient` - Means that a method is executed on the client side and the variable exists for the entire lifetime of the client part of a form.
  From the client method, calls to any methods are allowed.
- `OnServer` - Means that the method is executed on the server side and the variable exists only during a call to execute a server call.
  For server methods, calls to server, server out-of-context and client-server out-of-context methods are allowed.
- `AtServerNoContext` - means that the method is executed on the server outside the context of the form. This compilation directive can't precede variables.
  In such methods the form context is not available. While calling these methods a transmission of form data to the server and back is not executed. A use of non-contextual methods allows to significantly reduce the amount of transmitted data when calling the server procedure from the client application environment.
  The out-of-context server form methods are allowed to call server methods of common modules.
- `AtClientAtServerNoContext` - means that the method is executed both on the client and on the server, outside the context of the form. This compilation directive can't precede variables.
  Also, this method does not have access to the form module variables.
  From the client-server out-of-context method, not only the methods of non-global server-side common modules can be called but the methods of non-global common modules with the check boxes Server and Client (managed application).

An absence of a compilation directive before the procedure means using a directive by default. A directive by default is `OnServer`
When transferring control from client to server using the **Context call**, the following particularity must be considered: before starting a call, the form data is transmitted to the server side; then the server call is executed and then the form data is transmitted back to client. It can take a while. At the same time, the **Out-of-context server call** does not perform such transformations, therefore it is performed faster.
It is not recommended to keep references to the form attributes in the module variables (and other available storages) for the duration of operations that may lead to a re-creation of the form attributes. These operations include: calling the `ChangeAttributes (ИзменитьРеквизиты)()` method when working

on server and executing any contextual server call when working on the client side.
In the program module of a form (i.e., the code fragment placed outside procedures and functions) it is permissible to use preprocessor instructions for explicitly selecting sections of the initialization code of the corresponding variables.

IMPORTANT! In the server form variable, it is not possible to save data between two calls to the server side of a form.

It is not allowed to use multiple compilation directives before a single method or variable. It is not allowed to have methods of the same name or variables that differ only by the compilation directives.
Form command handlers created by a developer can be located only in the client methods of the form module.
In the form module it is recommended to use the preprocessor directives only within procedures and functions.

NOTE. To understand the result of "crossing" the procedure boundaries by the preprocessor instructions, should be taken into account that the preprocessor instructions are processed before processing the compilation directives.

Here is an example of using compilation directives:

```
&OnServer
Variable ServerVariable;

&OnClient
Variable ClientVariable;

&OnServer
Procedure Server()
  Notify(ServerVariable);
EndProcedure

&OnClient
Procedure Command1Execute()
  Notify(ClientVariable);
  Server();
EndProcedure

#If Server Then
  ServerVariable = "Server";
#EndIf

#If AtClient Then
  ClientVariable = "Client";
#EndIf
```

7.7. Form items

7.7.1. General information

A visual representation of the managed form is described by a set of elements. The form items form a hierarchical collection. The hierarchy of form elements determines the appearance and composition of the control items displayed on a form. There are several kinds of the firm items:

- form -the root item of the item tree exists in a single copy;
- form field;
- form decoration;
- form table;
- form button;
- form group;
- addition to the form element.

The form elements of the form field and form table kinds are always associated with the form data. If the associated attribute is not specified or not available on client due to a restriction on rights, or it is excluded from composition (the properties View and Editing properties of the form attributes), the field will not be visible on the form and it will be automatically deleted when creating a form in runtime mode.

Each form item is described by some set of properties that determine its behavior on the form, including the particularities of display and transformation -of the

item's size when the form or surrounding items are resized (including either disappearance or appearance of items).

7.7.2. Overall properties of the form items

This section describes the overall properties of the form items. The properties specific to certain elements will be described below.

7.7.2.1. The property group "Main"

The PositionHeading property determines how the item heading is displayed. Heading is the synonym for a form attribute associated with the selected form item,unless the Heading property is specified for a form item. Heading always ends with a ":" symbol (automatically added by the system). Headings of items and groups can be dynamically transferred (with the automatic increase in the height of a form) when all the following conditions are met:

- Heading consists of several words separated by spaces.
- The item with heading is not:
  - Any element subordinate to the managed form table;
  - Menu, button (Normal button or Command bar button kinds);
  - Page heading;
  - A header of the group if it contains additional design items (appearance of groups in the interface of version 8.2).

If data is displayed in the group heading and the above conditions are met, then the text transition will be performed but the minimum text width cannot be smaller than the size allowed for displaying heading data.

When adding an item in the Data property, a reference to the form attribute with which this item is associated must be specified. If the form item is not associated with the form attribute, then it will not be displayed on a form. If a button is added to a form, then in the Command property a reference to the command that will be executed when the button is pressed must be specified. If the button is not associated with a command, then it will not be displayed on a form.

A form item can be in View Only mode (prohibiting any changes) if it has the View Only property set (in configurator or using the 1C:Enterprise language) or the group to which it belongs has the View Only property or the attribute associated with it has a flag Stored data and a form is set to View only mode.

To control visibility of the form items, there are two properties: Visibility and User visibility. The first property can be changed both in the form editor (in Designer) and programmatically. The User visibility property is configured only in Designer and sets the initial visibility of a form item in the context of roles. The resulting visibility of the form item is formed by adding “by AND” the properties Visibility and User visibility for a specific user. In addition, by changing

visibility of the form items in the dialog box Setting form a user actually performs a change to the User visibility property of a particular form item.
Visibility change (the item property Visibility) using the 1C:Enterprise language, in most cases, does not require accessing server. However, in the following situations, the server call will still be executed:

- when changing visibility with functional options;
- when visibility of the Group item changes;
- when visibility of the controlled form table which displays a dynamic list, as well as when visibility of the columns in such a table changes.

Using the Skip at input property, a skip of the form item can be set when traversing a form using the Enter button. If a property is set to Yes, then an item will be skipped while traversing the Enter button (but available when traversing using the Tab button or pointing to the item with the mouse). If a property is set to No, then the item will not be skipped when traversing with the Enter button. It is also possible to specify the automatic determination of a need to skip the form item. To do this, it is necessary to set the value of this property to Auto. In this case, the following operation algorithm will be used:

- The item of the Field kind will be skipped in case of a view field, an inscription field or a picture field and also depending on the value of the property Display warning when editing.
- An item of the Button kind will not be skipped if the button is the default one.
- The item of the Decoration type will be skipped.
- The item of the Table kind will not be skipped.

7.7.2.2. Connection of items with the form attributes

7.7.2.2.1. General information

Any item displaying any data must be associated with a form attribute. The "normal" connection can be specified - when the form item displays the data of any form attribute and the "special" communication modes.

7.7.2.2.2. Connection with a regular form attribute

In the simplest case, the form item displays the value stored in any form attribute. Depending on the kind of attribute, the system automatically determines which form item will be used to display data and of what kind this item will appear (if it is possible).
In this case, the Data Path property contains the name of the form attribute whose data will be displayed by this item.

7.7.2.2.3. Connection to the current table data

A form item can be associated with an attribute which is a column of a table placed on a form. In this case, this item displays the data fields of the current table row. Such a connection is allowed for both fields and tables; it is not required that the corresponding column be displayed in the table. The item associated with the current data can be in both the View Only and the edit modes.
For example, there is a need to place a field on a form that displays price for the current line of goods. To do this, we add a field to the desired place in the hierarchy of the form items (a kind of the input field can be both the Inscription field and the Input field) for which as data we select (in the dialog Attribute selection) `Object.Goods.Price`. Here `Object` - Is a form attribute of the `Document` kind of a specific type, `Goods` - Is a tabular part of a document, and `Price` -Is a table part attribute.

7.7.2.2.4. Connection to a specific row in a table

A form item can be associated with an attribute which is a column of the specified table row appearing as a form attribute. Such a connection is allowed for both fields and tables; it is not required that the corresponding column be displayed in the table. A specific row is specified by a row index (for example, `ValuesTable[1]`) or a sequence of indices for the hierarchical tables (for example, `ValuesTable[2].TotalsList[4]`). The item associated with the string data can be in both the View Only and the edit modes. Please note that such a connection is allowed only with the attributes of the form and is not allowed with the items of the form. Другими словами, выражение `<ИмяРеквизита>[Индекс]` невозможно заменить на `<ИмяЭлемента>[Индекс]` или `<ИмяЭлемента>.ТекущиеДанные.<ИмяРеквизита>[Индекс]`.

NOTE. This connection is not supported for dynamic lists.

For example, there is a need to place a field on a form that displays the value of the field `Debt` from the 4th row of the table of values `Counterparties`. To do this, we add a field to the desired place in the hierarchy of form items (the input field can be both the Inscription field and the Input field) for which we select as data (in the dialog Select attribute) `Counterparties[3].Debt`. Here `Counterparties` - Are a form attribute of the `(ValuesTable)` kind and `Debt` - is a column of a table of values.
If these fields refer to a missing line -, then such a field cannot be edited, even if an input field is selected, otherwise the change made in the input field will be made in the table referenced by the input field.
In the example above, if a user changes the input field, a value from this field will go to the `Debt` column of the 4th row of the `Counterparty` value table.

7.7.2.2.5. Connection to attributes through reference

If there are attributes of reference kinds in the form data (for example, `CatalogRef`), then the form item can be associated with the attribute acquired via this reference. Data for such items will be automatically acquired and updated when the reference changes. Communication with the attributes through a reference can be of any depth. Items associated with such attributes are always in View only mode.

NOTE. For the attributes of a composite kind (including such kinds as `CatalogRef`, `DocumentRef`, etc.), obtaining attributes through a connection is not available.

For example, there is a need to place on a field on a form that displays the article number of the current product line. To do this, we need to add to the desired place in the hierarchy of form items a field of the following kind: Inscription field for which we should select (in the Select an attributedialog) `Object.Goods.Product.Article`. Here `Object` - Is a form attribute of the `Document` kind of a specific nature, `Goods` - Is a table part of the document, `Product` - Is a table part attribute and `Article` - Is an attribute for the catalog `Goods`.

7.7.2.2.6. Connection with the collections totals

A form element can be associated with the attributes which are the totals of some collections: tabular parts, record sets, list of values (only the number of rows in the collection). Such attributes can be:

- Totals by number fields,
- a number of rows per collection,

Items associated with such attributes are always in View only mode.
As an example, put on a form the field displaying a sum total for the tabular part `Goods)`. To do this, as the data of the form field, we indicate the `Object.Goods.TotalsSum` where the `Object` - Is the main attribute of a form, `Goods` - Is the tabular part of a document and the `TotalSum` - Is the Special attribute. Connection can be established not only for the item itself. It also supports setting a connection for displaying data in both table basement and header of a bookmark (groups of the `Page` kind).
Depending on the kind of data that the form element displays (input box, button and group), using the View (Вид) property one can set a method of displaying data.

7.7.2.3. A group of the "Use" properties

If the values of the properties Connections between the parameters of selection and Parameters of selection are specified in both the properties of the

metadata object attribute and properties of the form items, then the property values will be merged. A merge will be performed “by OR” (by the parameter names).
It is possible to specify an extended prompt for the following form items:

- Field;
- Table;
- Group;
- Button;
- Decoration.

An extended prompt is set using the Extended Prompt property of the corresponding form item. An ability to specify this prompt with the help of the Show extended promt command from the context menu of the form item is enabled. At the same time, the same extended prompt is a form decoration of the Text kind. A text of the extended prompt is set using the Header property of decoration. It is possible to control some of the parameters for displaying the prompt using the properties from the Appearance section of decoration. If there is a need to use a special method of presenting the text of prompt, then for this one needs to select in the editing window of the extended prompt the value of switching Formatted string . In this case, the advanced design features (like a formatted document) will be available. Also, the above form items can be set to display a prompt: around the form item, a button next to the form item, a pop-up window or a method of displaying will be selected automatically. If the prompt display mode is set to Button, then the property Headingof the extended prompt with formatting will be displayed. If the display mode is set to Pop up, then the property Heading of the extended prompt will be displayed but in the form of a normal string (with the formatting removed). A button for displaying a prompt is not displayed immediately after the item text but from the outer side of the right border of this item.
If both the Prompt and Heading- of the extended prompt properties are set at the same time, the text from the extended prompt will be displayed.
The Combination of buttons property is intended to speed up an access to a particular form item. If a combination of buttons is set for some form items, then pressing this combination immediately transfers the input focus to the required element. However, it should be remembered that for some keyboard shortcuts (some combination of buttons) that are used by the 1C:Enterprise system, it is impossible to override the actions performed. In other words, if one assigns such a "reserved" combination to perform another action, then when a combination of buttons pressed, a system action will be executed.

7.7.2.4. A group of the "Events" properties

This group contains links to the handlers provided by a particular form item provides.

It should be understood that calling any event handler (not necessarily a form item event) from code in 1C:Enterprise language is not equal to calling the same handler by a platform itself as part of the sequence of events associated with a particular user action. So, for example, a program call to the event handler of the `OnWriteAtServer()` module of the catalog form does not mean that a transaction will be opened and a catalog item will be recorded. With such a call, only the handler code will be executed and it is possible that the code will be executed incorrectly or errors might occur during the code execution. Errors can occur because a developer who created this handler, proceeded from the fact that it will be called only by a platform itself and the same platform will provide some environment.

7.7.2.5. Item images

The form items allow for using images for their design. Inserting an image is possible in two ways:

- from Designer,
- by programming.

If an image is inserted programmatically, then either an empty picture or a picture from the Designer picture library can be inserted.
If an image is inserted from Designer, then another option is possible- to set a picture from the file on disk (external picture). A use of images is recommended only in cases when the external reports or processing are developed that can be used in various configurations, and the picture is a significant element of design. In other cases, a use of such images is undesirable. For the form commands and global commands it is not possible to select an external image.

7.7.3. Forms

Describes visual properties of a form. An item of this kind always exists in the singular and is at the root of the item hierarchy. Also, in the form's properties the handlers of the form events are described.
When opening a form, it is possible to interrupt this process in several places:

- The event handler for the `OnCreateAtServer` event - A failure to open in this handler can lead to various consequences.
- The event handler for the `OnOpen` - event. A failure to open in this handler can lead to various consequences.
  If a form is opened so that the resulting object of the `ClientApplicationForm` kind will not be held in any variable, then the form itself and all data associated with it will be released from memory as a result of failure to open and, in particular, the form wait handlers which could be installed by this time would stop being executed.
  If an open form is held in any variable, then a form will be created, all associated data will not be deleted (will remain available for use) and the

installed form wait handlers will continue to work. This will retain an ability to try to open the created form again.

A mode for opening a form is controlled by the special property - Mode for opening a window. This property describes how a window will be opened:

- Independent - a form opens:
  - Forms in separate windows mode. A form opens in a non-mode window which is an auxiliary window of the application.
  - The Forms in bookmarks mode. The form opens in a separate bookmark.
  - Taxi mode. A form opens in the working area of the main application window.
- Blocking the owner's window, - a form opens in the non-modal window when work with a form from which the initiation of opening the current form is blocked. This mode is intended for forms in which little information is entered and work with which does not require a long time; for example, to enter elements of catalogs containing a small number of attributes. This mode is externally similar to the modal opening of a form; however, when opening from the 1C:Enterprise language, the operation of the module does not stop for the duration of the opening form. In this case, all other interaction with a form is performed as with other non-modal forms. This form also opens in an auxiliary window.

  A form opened in this mode does not participate in the search for already open forms. If one tries to open the exact same form (even with the uniqueness parameter set to `False`), then the form opened in the Block the owner window mode will not be found and a new form will be opened.

  This mode is set by default for the following forms:
  - Item and the group of catalog,
  - exchange plan node,
  - item and a group of plan of the characteristics kinds,
  - account,
  - kind of payment,
  - task
  - recording of the independent register information.
- Block the entire interface - Is similar to the form that opens in the Block the owner window mode but not only the parent form is blocked; rather the entire interface of an application. This opening mode is not used as a default setting.

When a form is opened in blocking mode, the following algorithm for determining the blocking window is used:

- If a form is set in the `FormOwner` property and this form is not closed, a window of this form is blocked.
- If a form item is set in the `FormOwner` property and this form is not closed, the form window to which this item belongs is blocked.

- In other cases, if the value `Undefined` is set to `FormOwner` property or the owner's form is closed, the window that is the current window of the client application at the moment of opening a form is blocked.

The property Mode of opening window does not affect the form opening in the following cases:

- If it opens in the modal mode. In this case, a form will be opened modally and in a separate window.
- When opened in an existing main or auxiliary window. In this case, a form will open in the existing main or auxiliary window.
- In the event that a form opens from the form blocking the interface. In this case, the parameter `WindowOpeningMode` of the `OpenForm()` method will also be ignored. A form will open in a separate window, in the non-blocking mode.

In the Taxi interface for a form opened in the Block owner window or Block the entire interface modes, it is not possible to customize the form navigation panel.

Every form has a heading A heading can be manually, configured automatically and be combined. If the form property Heading is not set and the Auto Headingproperty is set, then the form heading will be automatically generated. Explicitly specifying the Heading property (if the Automatic heading property is set) will cause the form heading to be formed from the value of the Heading property; then the automatic form heading will be output via ":". If a form has only the Heading property and the Automatic heading property is reset, then only the Heading property value will be used as the form heading.

The property Display heading allows for controlling a display of the form header (including a flag of the form data modification). A property can be changed from the 1C:Enterprise language in the 1C:Enterprise mode. A property is used only when working in the Taxi interface. If the normal mode of the main window is used, then this property applies only to forms located on the initial page of the main window.

If one of the following modes is specified for the main window: - Workspace, Fullscreen workspace or Kiosk, then the property for displaying heading is applied to all controlled forms of the application solution.

The Display closing button property controls the presence of a closing button on a form (if the opening mode for a form assumes presence of the closing button) and the ESC key behavior. A property can be changed from the 1C:Enterprise language in the 1C:Enterprise mode. A property is used only when working in the Taxi interface. This property does not control the button for closing window which is located in the heading of the auxiliary window.

The property Display the closing button is applied only if one of the following modes for the main window is specified: -Workspace, Fullscreen workplace or Kiosk.
The form property Vertical scrolling determines the position of the form items during the initial display, as well as how the form items behave if the form shrinks vertically:

- The value `Use` - A vertical scroll bar appears after the vertical size of the form items reaches the size that was specified when editing the form in Designer (this also applies to the items with an automatic determination of the element height).
- The value `UseWithoutStretch` - When the form is stretched, the form items do not react to the resizing of a form.
- The value `UseIfNecessary` - Before the the scroll bar appears the form items will be compressed vertically to its minimum size.

Consider an example: there is a form with an attribute `List`of the `DynamicList` kind. This attribute is displayed on the form as a table `List` for which its height is indicated in 8 rows of the table. If the form property Vertical scrolling is set to value `Use`, then the form will practically not be compressed:

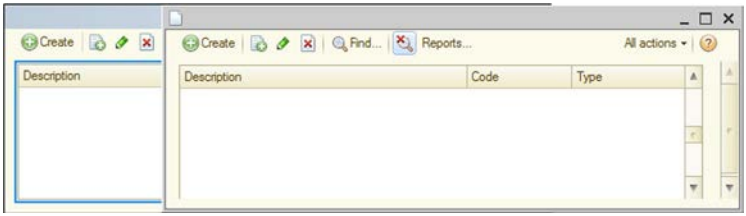


Fig. 142. Vertical scrolling form - use

In this case, the vertical scroll bar appeared as soon as the form was attempted to be squeezed vertically.

If the form property Vertical scrolling is set to the `UseIfNecessary`then a form will first reduce its vertical size as much as possible and only then will the vertical scroll bar appear:

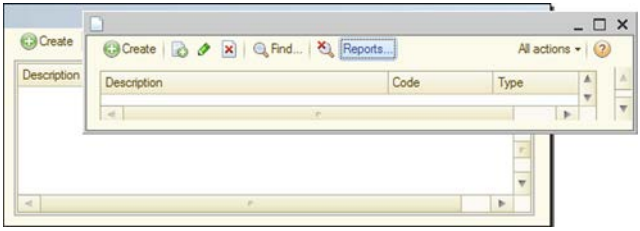


Fig. 143. Form vertical scrolling - to use if necessary

When setting a property to `Auto`, a form's behavior is determined by the system based on the kind of the main form attribute:

- If the main form attribute is of the `DynamicList` kind or the main form attribute is of the `ReportObject` kind and the `Report Result` property is filled in to expand the report form, then the `Auto` value is interpreted as the `UseIfNecessary` value;
- In other cases, the value `Auto` is interpreted as `Use`.

In the event that a form has vertical scrolling, then activating any item of a form that is not fully represented in the visible area of a form causes a form to try so that the fully or partially activated item either entirely (or by its greater part) is located in the visible part of a form. This can lead to the form's scrolling.

The Option version property determines how the managed form will be displayed in the Taxi interface:

- Auto - system automatically determines the form display mode. In this case, a property of the `ClientApplicationFormScaleVariant` of the `ClientSettings` object is analyzed. If a specific value of the scale option is not specified in this property, then in the Taxi interface the Auto value will be treated as Normal (Обычный).
- The Normal -form looks standard for the Taxi interface. The Scale property affects the size of a form.
- Compact - In this case, a form looks "almost" as in the interface of version 8.2. The Scale property does not affect the size of a form.

From a technical point of view, a form for which the value of the Scale Option property is set to Compact has the following features:

- As the form's font, the system font Font of the dialogs and menus is used;
- The form's items have dimensions that maximally correspond to those in the interface version 8.2;

- The distances between the form's items most closely correspond to the distances in the interface version 8.2;
- A tabular document, chart, Gantt chart, dendrogram, graphic scheme and geographical scheme are displayed without increasing the scale.

This property is intended to simplify a use of complex (saturated) forms in the Taxi interface without processing these forms. This feature is not essential since the resulting interface does not quite match the Taxi interface.

One can also control at once the scale of all the application forms. To do this, set accordingly a value of the `ClientApplicationFormScaleVariant` of the object `ClientSettings`. A value of this property will be used for all forms in which a value of the Scale Option property is set to Auto.

If a form is set to the usual display mode, then a form allows for controlling a scale of its display using the Scale property. This property indicates what the scale of this form should be relative to its base size (as a percentage). A property value can vary from 10 to 400. In the event that in the 1C:Enterprise mode the entire application scale is changed, then the scale set in the application will be calculated relative to the scale specified using the Scale property. So, setting the form scale to 200% (via the Scale property), the size of a form will be 2 times (for each coordinate) larger in comparison with the base size. If in the 1C:Enterprise mode for such a form a scale of 50% is specified (using the scale settings dialog), then the final form's scale will be 100%; that is, a form will take its usual view.

When a list of groups and the catalog's own contents are displayed simultaneously on a form, a task of synchronizing these lists arises so that when a group is selected from the list of groups, the list of items displays contents of the selected group. To facilitate this synchronization, the form extension property for a dynamic list List of groups is intended. This property becomes available if the main attribute of a form is the attribute of the `DynamicList` kind. In the List of groups property in this case one should specify a table of the controlled form that shows a dynamic list that displays only the group hierarchy of the displayed directory. In this case, the system provides automatic synchronization of lists when navigating through any of them.

For a form, there is a way to display a command bar (its location on the form can be controlled using the Position property of the command bar). A composition of the standard form commands can be adjusted through the form property Composition commands.

If the property Allow to change the form is removed, then a user cannot change composition and relative position of the form items in the 1C:Enterprise mode.

The Check filling automatically property is automatically responsible for checking the completion of form data.

If the form property Saving form data in settings is set to Use list value, then the Saving column becomes available in the list of attributes. The values of the

attributes for which the Saving property is set to True, will be stored in the form data store. To save, either the storage specified in the Settings storage form property or the storage specified in the Storage of settings of form data of the configuration will be used.

NOTE. The Save check box is not available for tabular parts of objects.

To set the View Only mode, there is a ReadOnly property on a form which is available only for programmatic changes. If this property is set to True, then the following standard commands will become unavailable (including when trying to use keyboard shortcuts to invoke these commands):

- For all forms:
  - Reinstall the parameters .
- For the tabular field:
  - Add,
  - Copy,
  - Delete.
- To expand a form of an object, writing the register of information, the constants:
  - Write,
  - Write and close,
  - Conduct,
  - Conduct and close,
  - Cancel conduct,
  - Start,
  - Completed.
- To expand the selection forms/settings:
  - Complete editing,
  - Select settings.
- To extend a table field for a dynamic list:
  - Nominate for deletion,
  - Conduct,
  - Cancel conduct,
  - Create a group,
  - Move to a group.
- To extend a table field for a list of values:
  - Change,
  - Move up,
  - Move down,
  - Sort descending,
  - Sort ascending,
  - Selection.

- All commands for changing the following collections of the data composition system:
  - collections of settings,
  - collections of the accessible fields.
- To expand the tabular field for `FormDataCollection`, `FormDataTree`, `Data FormsStructureAndCollection`:
  - Change,
  - Move up,
  - Move down,
  - Sort descending,
  - Sort ascending.

Also, form items associated with these commands will not be available.
If a form is set to the View only property and the main attribute of a form is a dynamic list, then all the tables associated with this list will also go into view mode. In addition, if the List of Groups property is filled, then the form item specified in this property will also go into view mode. In this case, the property of form items View only will not be changed.

7.7.4. Field

7.7.4.1. General information

The form item Field Is intended for displaying and editing any form attribute. A form field can be of several kinds:

- input field,
- inscription field,
- check box field,
- Image field,
- switch field,
- text document field,
- tabular document field,
- calendar field,
- indicator field,
- control band field,
- chart field,
- Gantt chart field,
- dendrogram field,
- graphic scheme field,
- geographic scheme field,
- HTML-document field,
- formatted document field;
- scheduler field.

If a field is a subordinate item for the form item Table, then it can take the following kinds:

- input field,
- Image field,
- inscription field,
- check box field.

In order to place an input field with views on a form: a tabular document field, chart field, Gantt chart field, dendrogram field, graphical scheme field, geographic scheme field, one must create a form attribute of the appropriate kind and specify this attribute as form element data.

In order to place an HTML document field on a form, there is a need to create a form attribute of the `String` kind, then create an input field HTML Document field field and specify the created attribute as data for this field. A value of the form attribute can be a navigation reference or a text of an HTML document.

NOTE. Note that use the HTML document field does not fully support the entire HTML functionality. In particular, no support is provided for the stateful scripts (variables storing values required by various event handlers), working with the SVG markup language, the property of an HTML document `parentWindow` as well as clicks on the links inside the displayed page. Restrictions may be imposed due to specific features of the functioning of HTML (including non-identical functioning) in the “1C:Enterprise” forms in the client applications as well as various web browsers.

To place a formatted document field on a form, there is a need to create a form attribute of the `FormattedDocument` kind, then create an input field of the Formatted document field kind and specify the created attribute as data for this field.

The input field property Input prompt allows for specifying a hint that will be formed directly “inside” the input field if the following conditions are met:

- A kind of the associated attribute is not a `Number` or `Date` kind.
- A value of the associated attribute does not contain the default value for this type.

If the input field displays the data of the `ValueList` kind, then the form of editing the list of values opened from such a field receives the following properties from the input field:

- `Mask`;
- `ChoiceForm`;
- `EditFormat`;
- `QuickChoice`;
- `ChoiceFoldersAndItems`;
- `ChoiceParameters` (supplemented with values, in accordance with the `ChoiceParameterLinks` property);

- `TypeLink` - A transfer of the constraint kind is performed only for the case of editing the data composition selection item; if this property is set for an input field that edits a right-hand value that contains a `ValueList`, then this restriction kind is set to a list of values.

The input field will ignore the `BackColor` property if the `Availability` property is set to `False` for this field or the `ReadOnly` property is set to `True`.

7.7.4.2. Updating of the data displayed

When working with an input field, it should be remembered about the peculiarities associated with changes in the form data or its (form) structure while working with data in the input field. The peculiarity lies in the undesirable updating of data in the input field which occurs when the following conditions are met:

- In the input field there are data that are not recorded in the form attribute associated with the input field. For example, at inputting text.
- At this time, a wait handler or an external event handler is triggered.
- In the event handler, the form data or a composition of the form items are modified.

In the described case, the text displayed by the input field will be replaced by the presentation of the data that was received in a form data after the context call was completed. This behavior can be changed in several ways.

`Property EditTextUpdate`

If in the above situation it is necessary to block the change of only one (or a small number) of the input field, then such a lock can be set using the input field property `EditTextUpdate`. If this property is set to the `OnValueChange` value, then the contents of the input field will be updated only if a value that is different from the current value of an attribute is placed in the associated form attribute. If the `EditTextUpdate` input field property is set to `Never`, then the text in the input field will not change even if the value of the associated attribute changes.

`Method RefreshDataRepresentation()`

If in the above scenario there is a reverse situation: there is a need to update only a limited number of fields and leave the remaining fields unchanged, then the `RefreshDataRepresentation()` method can be used to update specifying as a method parameter a list of the form fields that should be updated. If the parameter is set, the values will be set and the information displayed will be updated only for the specified field (or list of fields). If a blank array is passed as a parameter, then values will not be set for all the items of the managed form.

Method parameters can be only the following form items: `FormGroup`, `FormTable`, `FormField`.

Using the `RefreshDataRepresentation()` method makes sense only in the following cases:

- from an external event handler;

- a form wait handler;
- alert handler.

In other event handlers, the behavior does not change: values will be set (and, as a result, the data display will be updated) for all the items of the managed form (in web client - only those values that differ from the current form data).

As an example of using this method of updating data, the following use case can be used: In the input field, data identifying the counterparty is entered. In the wait handler, data from this counterpart is checked and some form items reflecting the result of a check are changed. In this case, it is recommended to use the `RefreshDataRepresentation()` method in the wait handler, specifying the items associated with the check results as a parameter list. The remaining form items will not be updated. It should also be remembered that a call to a context server call that changes the form data in the wait handler will unambiguously change all form data except for elements with the modified value of the `EditTextUpdate` property.

7.7.4.3. Data representation

Details of form attributes can be displayed in the form table and directly in form items. A table can specify only data displayed in the form details and being part of a collection. Generally, the following rules applies to a client application form: it represents data available in the form attribute designated as a data source for the form item (with accuracy of a form item kind).

If data is displayed in the form field and the said field has List Selection Mode check box duly checked, data representation depends on the selection list. As such, the form fields displays Representation column from the selection list, rather than data representation (if the said selection list has data currently available in the attribute). If data is displayed in tabular format, no selection list is used. **Always** table cell will use data representation available in the displayed field of the current table row.

This is due to the fact that each table row can have its own selection list. However, input field for a table row is generated when you start editing. Thus, when a table is displayed, the form is unaware of a proper selection list to be used for value representation purposes.

In view of the foregoing, the following behavior can be observed when data is represented and edited in tabular format:

- Data is displayed through its representation.
- When you enter the cell edit mode, input field starts to display representation which uses a selection list, provided that input field is assigned list selection mode.
- As soon as editing is over, data is displayed again through its representation (without regard to a selection list).

There are several scenarios to manage data representation in the form table:

- Use conditional design of a form.

- To edit a proper column, use a separate form input field and specify an updated selection list for the same, when a table row is changed.

7.7.4.4. Warning during editing

If an attribute is edited in the form input field which is filled automatically, but can nevertheless be filled in manually (in extremely rare cases), then this behavior can be implemented using the properties of the input field Display warning during editing and Warning during editing. If the value of the Display warning during editing is equal to Display, then when while trying to start editing the 1C:Enterprise field, a warning will be received which consists either of the line specified in the Warning during editing or is generated automatically.
For the standard `Code` and `Number` fields (for the objects with automatic numbering), the automatic warning string will look like the Number is automatically filled in during recording and for the rest of fields the automatic string will look like Editing the field "Field Name" is not recommended. If the value of the Display warning during editing property is Auto, then for the `Code` and `Number` standard attributes the value Display will be used and for the remaining fields - Do not display. The property Display warning during editing also affects the the input field property Skip at inputing. If the value of the Skip at inputing is equal to Auto, then the field will be skipped when inputing if the value of the Display warning property during editing is equal to Display (or Auto for the standard `Code` and `Number` fields).

7.7.4.5. A blank mark

If the `AutoMarkIncomplete` property of the input field is set to `True`, then the property of this field, `MarkIncomplete`, will be automatically updated when the value is transferred from the attribute to the input field.

7.7.4.6. Displaying switch and a check box

If a field of the Switch Field kind is inserted, then it is necessary to set for it the property of an item of the Selection List which determines a number and values of the switches. The horizontal location of switches is used by default.. To arrange them vertically, it is necessary to change the value of the Number of columns property. In case if a number of columns is set to 0, then the system can arrange the switch items in several lines selecting a number of the switch items per line based on availability of free space on a form. Selection is not performed dynamically, rather at the time of form creation.

For the fields of the Field of check box and Switch field kinds there is a possibility to specify a kind of display. For the check box field, it is done with the help of the property Check box type. The representation in the form of a toggle, switcher or check box can be chosen for the check box field.

Fig. 144. Check box types

On fig. 144there are various options for displaying a check box field. Each type of check box is described in the title of each line.
The form field of switch field type can be represented in the form of toggle or switch with the help of the property Switch type.

Fig. 145. Switch types

On fig. 145 there are various options for displaying a switch field. In the upper part of image the switch field is represented as a toggle and at the bottom - in the form of a switch.

7.7.4.7. Editing standard periods and dates

For the form attribute of the `StandardPeriod` or `StandardBeginningDate` kinds there are two ways of editing:

- Using one form field. In this case, it is possible to place on the form a field associated directly with the edited attribute. Then all editing will be performed in this field.
- Using multiple fields. In this case, on a form the can be placed the fields associated with the properties of attribute. In this case, a developer can himself implement the necessary logic of interaction between the standard period option and the date values.

7.7.4.8. Spreadsheet document field

To expand the field of a managed form for a tabular document, it is possible to customize its presentation:

- Selection display - allows for controlling the way the selection is displayed in a spreadsheet document:
  - Always - A selection will always be displayed, including if the field of the tabular document is not active. This value is set by default.
  - When active - A selection will be displayed only if the field of the tabular document is active.
  - Do not display - A selection will never be displayed.

- Selected multiple cells - A selection is displayed if it (the selection) consists of more than one cell in a spreadsheet document.
- Several active cells selected - A selection will be displayed if the selection consists of more than one cell and the table document field is active.

A behavior of the cursor keys in the extensions of the managed form field for a tabular document depends on a value of the Display Selection property. When this property is set to Do not show, the cursor keys perform a function of scrolling a tabular document in the direction of arrows. When a property is set to one of the values If several ... are selected (При выделении нескольких ...), then with the Shift key pressed the cursor keys change the selection size and without pressing the Shift key perform scrolling of a tabular document in the direction of arrows (without deselecting the area).

- A vertical / horizontal scroll bar - allows for controlling a presence of scroll bars in a spreadsheet document:
  - Do not use - The scroll bars will not be used.
  - Use always -The scroll bars are always present in a spreadsheet document.
  - Use automatic - The scroll bars are used in a tabular document only when the contents of that document according to the corresponding dimension does not fit in the visible area. This value is set by default.
- Type of cursors (Тип курсоров) - Allows for controlling the appearance of a mouse cursor which is located above a tabular document:
  - Special - A mouse cursor changes depending on which area of a spreadsheet document the cursor is located. This value is set by default.
  - Standard (Обычные) - A mouse cursor always has the appearance of a standard cursor (“arrow”).

7.7.5. Decoration

It is a design element of a form. A decoration is not associated with data (form attributes). A decoration can be an inscription or image.

A text displayed by a decoration of the Text kind can be dynamically transferred (when a form is compressed or stretched) while the following conditions are met:

- A text contains several words separated by spaces;
- A value of the decoration property Stretching horizontally is explicitly not set to the value No;
- A value of the Height property is set to 0;
- A value of the decoration property Stretch vertically is explicitly not set to No.

A decoration of the Image kind will automatically determine its size (by width and height) according to the set image if the corresponding property of a decoration (Width or Height, respectively) is set to 0.

TIP. It is recommended to use a decoration only for the unchangeable text explanations. It is recommended to display the text that can be changed from the 1C:Enterprise language using the field of the Inscription View kind.

7.7.6. Tables

7.7.6.1. General information

The form item Table is intended for visualization of tabular data. This can be a dynamic list, a table part, a list of values and so on.

A position of the table command bar is controlled by the property Position of command panel of the form item. A composition of standard commands located in the command bar of the table is regulated by the property Composition of commands. Commands that are disabled when editing this property become unavailable.

It is possible to group columns using the Group form item.

The `BorderColor` and `Border` properties set for the table column do not affect the column display.

For a form attribute of a `ValueList` kind, there is a choice of how the attribute will be displayed. If the option Table is selected, then the list of values will look like a table with available columns: Value, Presentation, Mark and Image. It should be remembered about the features of displaying values in the Value column. This column will display a value of the Presentation column. A value itself remains unchanged. If the Input field kind is selected, a special window will be opened for editing the list.

Also, a standard set of editing commands including setting and unmarking commands, is provided for the table associated with the `ValueList` kind attribute.

If in a vertical group of columns in a certain row the width of columns exceeds the width of columns from other rows, then in the narrower rows the last column of a group stretches to the width of the widest row. Stretching is performed even if for this column the Stretch horizontally property set to `False`. If a column stretching is to be avoided, then that column should be moved out of the group to be stretched and be placed separately.

7.7.6.2. Dynamic list display

When a form with a dynamic list is opened, data is read only for visible tables associated with dynamic lists. For the tables that are not visible when a form is opened, the data is read at the time they (tables) are displayed to a user. In particular, for the invisible tables, a value of the property `CurrentRow` will be

equal to Undefined. This particularity should be considered when developing a form logic.

NOTE. If a table displays a dynamic list and the Initial tree display property is set to Expand all levels, then when creating a form on server will be retrieved only a list of rows displayed at the first level and then, when a form is opened, the rest displayed rows will be retrieved. And for each displayed group, a separate request to server will be executed.

If a table is associated with the DynamicList kind attribute that has grouping set in the list settings (a property of the form attribute List Setting), then a list will always be displayed in the Tree mode (a property Display of the form item) regardless of which display mode has been set by a developer. If View Only property is set for the table associated with the dynamic list, when opening objects from such a list of forms they will automatically be set to View Only form parameter.

Also, for a table associated with a dynamic list, there is a possibility to manage the update of a list when data changes (in the same client application) using the Update when data changes property. If a property is set to Auto, then a list will be automatically updated for any changes in the displayed data. If a property is set to Do not update (Не обновлять), then a list will not perform an automatic update; to update it, one must explicitly execute the Update (Обновить)() command. If a dynamic list is set to automatic update at a specified interval and the Update when data changes property is set to Auto, then an interval is counted from the nearest data change; and if the value is Do not update - from the last automatic update call or execution of the Update () command.

For a table associated with the main form attribute of a dynamic list type, there is a possibility to control the multiple selection mode using the corresponding form parameter. A value of the form parameter Multiselect will be set to a table property of the same name. In this case, a property Selection mode will be set to the value Multiselect if the form parameter Multiple Select is set to True.

For a dynamic list in the selection mode, when adding a new object, a check will be performed for the added object to meet the specified selection criteria; and if the new object meets the selection criteria, a cursor will be pointing to it in the table displaying the dynamic list.

If a form of the list of accounts or a catalog with the hierarchy of items opens with the set parameters AllowRootChoice equal to True and ChoiceMode equal to True, then the following properties will be automatically changed for all the tables of the form associated with the main form attribute (such as DynamicList):

- the ShowRoot property will be set to True;

- the AllowRootChoice property will be set to True;
- the Display property will be set to Tree.

7.7.6.3. Image display

A standard image of the rows is formed based on the standard fields of the tables used to determine a state of the item (DeletionMark, Predefined, etc.)). If in the text of query list there are field the aliases of which coincide with the names of the standard fields but other data will be acquired in them -, then such fields will not be used to form a standard image.

If necessary, to substitute a standard image one should:

1. Add to the library of images the required image collection;
2. Select the added image in the RowsPicture property of the table associated with a dynamic list.
3. Add to the request text a field that has an image index value.

Example:

```
CASE
  WHEN Table.DeletionMark THEN 2
  WHEN Table.Predefined THEN 1
  OTHERWISE 0
END AS ImageIndex (PictureIndex)
```

4. Specify the added field in the RowPictureDataPath property of the table associated with a dynamic list.

An image of rows will be displayed at the first (counting from the left) visible field of a dynamic list of the input field or inscription field kinds. If in the process of editing the table column composition there is a change in the order of columns, then an image will also be displayed at the column that has become the leftmost (from the view specified above). However, if a deactivation of columns visibility is performed using conditional design, then the system behavior becomes somewhat different: if a column to which an image is attached (or the hierarchy symbol) is hidden, then an image (or hierarchy symbol) will not be displayed on this row.

7.7.6.4. Modifying table height

To control the height of the client application form table, use ItemHeightControlVariant property:

- In form rows - in this case, the table height is set using the Height property of the managed form table. If a value of this property is equal to 0, then the height of a table will be 16 units for the table displaying a dynamic list and 8 units - in all other cases. To limit the maximum height of a form table, AutoMaxHeight and MaxHeight properties are used.
- In the table rows - in this case, the height of the table is set using the property HeightInTableRows of the table of the managed form. If the

value of this property is equal to 0, then the height of a table will be 15 rows for a table displaying a dynamic list and 7 rows - in all other cases. To limit the maximum height of a table, the properties AutoMaxHeightInTableRows and MaxHeightInTableRows are used.

- By content - if the table is linked to a dynamic list, then the table height is set by the property value HeightInTableRows. If a value of this property is equal to 0, then the height of a table is determined by the value of the Height property.
  If the table is not associated with a dynamic list, then all rows of the table are fully displayed but starting with 3.
  If the AutoMaxHeightInTableRows property is set to True, then the table displays no more than 50 rows. If the AutoMaxHeightInTableRows property is set to False, then a number of the rows displayed in the table is limited by the value of the MaxHeightInTableRows property. If the value of the MaxHeightInTableRows is equal to 0, then the table has no restrictions on the number of rows displayed.

When setting this option of controlling the height of a table, the system ignores a value of the VerticalStretch property and assumes that this property is set to False.

7.7.6.5. Column scrolling

When designing forms, there are situations when it is necessary to prevent certain table columns from scrolling. For example, in the tabular part there is a need to fix the columns Number, Products and Sum. In this case, the columns Number and Products must be pressed to the left edge of a table and the Sum - to the right.

To perform such an operation, the FixingInTable property is intended. This property is available for the field located in a table and for a group of theColumn group kind. In this example, it is necessary for the Number and Products columns to set the FixingInTable property to the value Left and for the column Sum - to the value Right. The columns will be pressed to the corresponding edge of a table regardless of where these columns were located in the form editor but the fixed columns are arranged in the order in which they are located in the form editor.

If the fixation flag is specified for a group, then the entire group will be pressed (along with all the subordinate items of the form), while a value of the FixingInTable property for the subordinate items is ignored. If the fixing flag is specified for a column belonging to the group for which the FixingInTable property is set to No, then a column will be treated as a separate column not included in a group.

7.7.6.6. Search in tables

No search made in a table has impact on the content of `FormDataCollection.RowFilter` property. `FormDataCollection.RowFilter` property content cannot be displayed through View State form item add-on.

The property Search when inputting characters is responsible for how the system will process the character set from keyboard if the active item is a table and the direct data entry into a column is not possible:

- Auto - in this scenario for a table associated with a dynamic list is interpreted as Use, while for a table displaying any other type of data, it is interpreted as Do not use.
- Use - means that search upon symbol input is made for attributes of any type associated with a table, if for a column where symbols are input Edit Mode is set to Log in. Either search line (if allowed by Search Line Location) or search form (if search line is not supported by this table) will be used to search data.
- Do not use - means that no search is made when you input symbols for attributes of all types associated with a table.

The following algorithm is used to search data:

- The search string is split into words. The search string should not exceed 1,000 characters, and the number of words obtained during splitting should not be more than 20. The splitting is done in compliance with the following rules:
  - A string is split by using the space and tab whitespace characters for separators.
  - Next, each resulting fragment is processed:
    - If a fragment is in the date format (with or without time indication) with regard to the current regional settings of the session, then this fragment is considered to be a single word.
    - Otherwise, the fragment is being split further by using the characters ", .- / \" as separators. In this case, each resulting substring is considered to represent a separate word.
- For each word, its own set of conditions for each table column is generated that are combined "by OR". Conditions are generated depending in the column type:
  - `String`. A word is searched using its initial letters. If a word contains letters and digits:
    - A sequence which contains letters only is deemed a separate word.
    - A sequence which contains digits only is deemed a separate word.
  - `Number`. A word is converted into a number. If the said conversion is not feasible-, a search by field will be aborted. If the said conversion is successful, a search will be made using the following condition: `FieldValue >= Integer(Word) AND`

FieldValue<(Integer(Word)+1). For values with leading zeros, you can perform a search on either the string with leading zeros, or on the string specified without them.

- Boolean. The word equality with localized value representation of Boolean type is verified (for current session localization language).
- Date. A word is converted into a date with a month and a year added by default. The field value will be verified for availability thereof in a day defined by the date so obtained FieldValue >= DayStart(Word) AND FieldValue<=DayEnd(Word)).
- So far as reference fields are concerned, a search will be made using initial letters of a word available in the reference field representation.

A set of conditions for each word is combined “by AND”.

An element in a table is deemed as found (inter alia, if it has subordinate elements), if at least one of the following conditions is met:

- It contains all searched words.
- If has at least one searched word, while the other words are available in its parents, in the first visible column of a string or reference type.

If a column value is modified upon completion of a search attempt with respect to any string, it remains visible, until the searched value is edited and the search is repeatedly made.

If a table has a predefined search filter, commands intended to move the current table element up and down are disabled for the table.

7.7.7. Button

On a form, the command is displayed by the form item Button. A button can be displayed in the command bar and just on a form. In the case of its display on a form, a button may also have the form of a hyperlink (for this there is a need to change the View (Вид) property).

A command that will be accessed by pressing a button is set using the Command property. A button will be shown as disabled if the command associated with this button has the property Modifies the saved data set to Trueand the ReadOnly property is set to True for this button or the form as a whole.

If the button associated with the command is located in the command panel its position is managed by the property Position in command panel (PositionInCommandPanel):

- In additional submenu -the button is positioned only in an additional submenu More.
- In command panel -the button is positioned only in the command panel.
- In command panel and additional submenu -the button is positioned simultaneously in the command panel and More menu.
- Auto -the position is chosen by the system automatically depending on the command.

At the same time, the user can change the position of the button with the help of the form settings editor which is available in 1C:Enterprise mode with the help of the command form More -Change form.

The property Unique commands(UniqueCommands) is intertwined with the property Position in the command panel. In short, this property allows positioning two buttons associated with one standard command in one panel with automatic filling in but these buttons shall be positioned in different places of the command panel. The key issues: a) it shall be a command panel with automatic filling in and b) it works only for the standard commands. Let's see an example.

Let's suppose there is the attribute named TabDoc of TabularDocument type in the form. This attribute is placed in the form (with the same name) and the command panel (TableCommands), for which the attribute TabDoc is selected as the command source, is created in the form. We want to position the command Fix table in the form command panel and to leave it in its default position (More -View -Fix table). At first, we drag the command Fix table from the list of standard commands (in the form editor Commands -Standard commands -TabDoc- Fix table) to the command panel TableCommands. In the result, the command will be in the menu More (it is the standard position for this command) but will disappear from its default position. We are not satisfied with it. Therefore, we point at the button TabDocFixTable in the editor and set the following properties:

- The value In command panel for the property Position in command panel. Now the required command has turned out to be visible at once and has disappeared from More menu.
- Clear the property Unique command. In the result, the command will reappear in the position on the menu in which it was by default.

For the button associated with the Create by Parametercommand one can specify the kind of a document that will be created when a button is clicked. This can be done using the Parameter button property.

If the Prompt property is not set for the form command, then for the button associated with this command a pop-up prompt will not be displayed. If the button's heading cannot be fully displayed on a button, then the full text of the heading is displayed in the pop-up prompt by the first (or only if the button does not have a prompt) row.

A button can have several display modes (the Display property). If the Image and text display is selected, then the image located on a button can be located on the left or right relative to the button text. To control this location, the button's property Position button (available for the form and command bar buttons). If this property is set to Auto, then an image will be located on the left side of a text.

In the event that an image display is selected for a button, an image will always be located in the center of a button.

A button of the controlled form can be displayed in various ways. For this purpose, the property of the button property Button image is intended.

This property can have the following values:

- Auto - In this case for standard commands the value specified for a command is used. For other commands the value is treated as Normal.
- Normal, Oval - button image corresponds to the value (**Ошибка! Источник ссылки не найден.**).

The button's image for buttons located on a form and in the command bar can be specified. A property does not affect commands located in submenu.
For the button displayed on a form, it is possible to control a display of the image of this button. This is done using the Image Display property.

To display the button's image, several different ways ca be specified:

- Auto - In this case for standard buttons the display option specified for the command is used. In other cases it is treated as Always.
- Always -In this case, the button image is always displayed.
- When active - In this case, in the normal state, a button is displayed without a background and border. In case if the mouse cursor is pointing to a button, the input focus or the Mark - property is set on it. A standard button with the specified border and background colors is displayed.
- No - In this case, in the normal state, the button is displayed without a background and border. When hovering cursor, an underline appears in the text and the mouse cursor changes. The Mark property is not displayed whatsoever in this display option.

The property defines a display for buttons located on a form and in the command bars. The Image Display property has no effect on commands located in submenu.
Another way to control the appearance of the button exterior view is to set the View property. Depending on where the button is located, this property can take on the following values:

- Regular button, A button on the command bar - In this case, a button is displayed as usual;
- Hyperlink, The hyperlink of the command bar -In this case, a button is displayed as a hyperlink.

When dragging a button between a form and a command bar, a value of the View property is preserved adapting to the location of a command.

7.7.8. Group

7.7.8.1. General information

When developing a form, there is a possibility of different combinations of items. Fields, form pages, commands, columns can be combined.

A form may contain groups of items of control. These can be field groups, pages, command groups. It is also possible to create column groups for the Table type elements.
In editor, is possible to create groups of the following types:

- Regular group - Is intended for combining form items.
- Command bar - A form item designed to accommodate buttons and groups.
- Pages - This group is intended for organizing a panel with bookmarks. Bookmarks can be located both vertically and horizontally. In order to add a page to such a bar, you need to add as many nested groups as the page should have on the bar. Inside the group of the Pages kind there can be only groups of the Page - kind of the group of special kind intended to form bar pages. This group may contain other nested items.
- The Column group - allows for combining columns in a table. With the help of this kind of group the rule of column grouping (vertical or horizontal grouping) can be changed.

Groups can be nested. You can move the form items between groups. This automatically determines the admissibility of such a transfer. If the transfer requires changing any item properties (for example, View), the change is performed automatically. If, as a result, the requirements to subordinate items change, they are either automatically changed or deleted.

7.7.8.2. A standard group

A standard group may have different design options:

- No - A group is not specifically marked up;
- Weak markup - A group heading is displayed with a large green font;
- Standard markup - A group heading is displayed with a large green font; Around the group items (on each side) the indents are formed;
- Strong markup - A group heading is displayed with a large green font; On the left side (for the entire height of a group) a green band is formed. An indent is formed from below.

If background color of a standard group differs from Auto, then the indentation appears on all sides of this group. The indent of a standard group is defined for each edge (left, right, up, down) but it does not apply if a group touches the edge of a form by its edge.
A standard group can be configured in such a way that a user (in the “1C:Enterprise” mode) can collapse or expand a group. For example, there is a possibility to place information in a collapsible group that is not important when working with a form on a regular basis but may sometimes be required for viewing and analysis.
A behavior of the group is determined by the property Behavior:

- A standard group. For such a group, the Behavior property is set to Standard. This group does not change its display in any way during user actions

(except for programmatic modification of the composition and the external view of the group and its items).

- A collapsible group. For such a group, the Behavior property is set to Collapsible. Such a group can be collapsed or expanded by a user in the “1C:Enterprise” mode. An application developer cannot determine the current state of the group from 1C:Enterprise language. Also, he cannot forcibly collapse or expand a group. The initial state of a group is determined by the Collapsed checkbox. A group cannot be collapsed or expanded if it does not display a heading or it is not filled.

  A user can manage the state of a group (collapsed or expanded) using a special image or hyperlink. To set the state control item, use the Display control property. Also, there is an opportunity to set the group heading (Collapsed heading property) which will be displayed if the group is in the collapsed state. If the Collapsed heading property is not filled, then the normal group heading will be displayed in the collapsed state.
- A pop-up group. For such a group, the Behavior property is set to Popup. In the initial state, a pop-up group is displayed in the same way as a collapsed group.
  When the heading of such a group is clicked, the group "pops up" over the parent form in a special window. At each time point on a form there can be only one pop-up group. This implies that pop-up groups cannot be nested. The nested popup group will appear as a collapsible group. Showing and hiding a pop-up group does not require a server call.

  A pop-up group must have a heading. Without a specified heading, a pop-up group behaves like a standard group. A heading acts as a text of the hyperlink clicking on which leads to displaying the window with the contents of a group.

When interactively navigating to an item (using the Tab button), which is located in a collapsible or pop-up group, the nested items of the group are automatically displayed in the required mode. When interactively exiting a collapsible or pop-up group (using the Tab button), a focus is transferred to the next form item (in the moving order). At the same time, the collapsed group remains expanded and the pop-up group will be hidden.

A program interface is available from 1C:Enterprise language which allows for performing the following set of actions:

- To display a group. The method for expanding a group of the managed form `Show()` is used. At calling this method:
  - A standard group does not respond in any way. An error is not generated.

    - A collapsed group gets expanded. The active form item remains unchanged.
    - A pop-up group is being shown. The first item in the pop-up group becomes active.
- To hide a group. A method of expanding the group of the controlled form `Hide (Скрыть)()` is used. At calling this method:
    - A standard group does not respond in any way. An error is not generated.
    - A collapsing group gets collapsed. The active form item remains unchanged.
    - A popup group is hiding. An item becomes active in front of the pop-up group heading.
- Get the group's state. The method used to expand a group of the managed form is `Hidden()`. At calling this method:
    - For a standard group, a `False` is always returned.
    - For a collapsible and pop-up group, `True` is returned if a group is collapsed or not shown (respectively). `False` returns otherwise.

7.7.8.3. Command bar

A command bar property Command source defines a form item (Form as well as form items of the Table kind and the fields of the Field table document or Graphic chart field) which will provide "its" commands for display in a command bar. A composition of commands for display in a command bar is governed by the property Composition of the form item commands that is a source of commands.

In case when all buttons in the command bar have a value of the `OnlyInAllActions` st to `False` and not all buttons can be placed on a command bar (in width), the More button (with a submenu) is automatically formed in a command bar. This submenu contains all the command bar commands.

A group added to the command bar can be one of the following types:

- Submenu (Подменю) - A group that allows for making drop-down menus.
- Group of buttons - allows for creating a group of buttons that will have the following properties:
    - a group of buttons is separated by separators on the left and right;
    - each group of buttons can have its own source of commands;
    - It is possible to set a density of buttons in a group: compact or standard (property Display). A value of the Auto property is treated as Standard.

The Display property affects only those buttons of the command bar for which the Display of image property is treated as Always.

7.7.8.4. The features of using form groups

The View only group property affects all the items subordinate to a group.

The group property Allow change of composition controls an ability to change composition of a group in the process of setting up a form by a user. If the property is disabled, a user cannot change composition and the order of the items within a group. However, the property does not affect an ability of a user to change visibility of the group items.

If a group has the Heading data property, this data will be automatically displayed in the group heading. If a group has the Heading property set, then the data of Heading is displayed in brackets after heading: Heading (Heading's data).

Any group of buttons (command bar, submenu, group of buttons, context menu) is populated according to the same rules. Buttons will be displayed in the following order:

1. The buttons provided by the command source this group is associated with.
2. The command interface buttons if the command source implies its inclusion.
3. The user commands added to this group.
4. Menu commands More (Еще) ((All actions (Все действия)) (only for a command bar).
5. The last one will be Reference (if it was provided by the command source).

If the above automatic order of disposition of buttons in a group is not satisfactory, then the following features can be used:

- Clear the AutoFill check box for the form command bar, context menu or item command bar and add the necessary commands (in the desired order) manually.
- Disable a command source of a command bar, then add the necessary commands (in the desired order) manually, then add a group of the Button group kind and set the command source for an added group.
- There is a possibility to manually place the button associated with the standard command of a source command in the desired location on a command bar. In this case, the corresponding command will not be used when the source commands are automatically placed in the current command bar but will remain at the place where it was placed manually.

7.7.9. Adding a form item

Addition of a form item is intended for displaying additional properties of the form item (addition of a form item) and managing these items. Form item add-ons are supported for Table. The following add-ons are available:

- Display Search String. Allows it to customize display of the search string.
- View State. Allows it to customize display of a list of applied search queries . Cannot be located in a command bar.

- Search Control. Allows it to customize appearance of the search control button.

The said add-ons are described in more detail below. A table available in the client application form enables you to perform interactive search in displayed data. Search can be carried out by using the following tools: search line, search dialog box, current value search, search history and display period. The display period can be set, only if a table displays a dynamic list where a document table or log is used as its primary table.

As soon as search is completed, limited record set is displayed in the spreadsheet (from all those available to the current user) complying with search criteria. You can set up search parameters for all tables associated with the following form attributes: dynamic list, value table, value tree and tabular section of any object.

To manage search capabilities, a form table supports the following Table form item properties:

- Search bar position - defines the search bar position. Can take values: Auto, Command Bar, None, Top, Bottom.

  If this property is set to Command bar, displayed parameters depend on the table data source:
  - Table will display dynamic list data. As such, a search line is displayed in the form command bar (if a dynamic list is a primary form attribute) or in a command bar associated with a dynamic list.
  - Otherwise, a search line is displayed in the command bar associated with the table being set up.

  The search bar contained by the command bar is always docked to the right edge of the command bar (along with the buttons located to the right of the search bar).

  If the property is set to None, the search bar will be absent on the form, and a dialog will open when user starts typing in the search bar.

  If the property is set to Top, a search line is placed between the table command bar and the form table. If the property is set to Bottom, a search line is placed immediately below the table.

  Auto is interpreted as No (if a table displays a value tree or list) or Command bar in any other case.

  Calling the search bar is performed as follows:
  - By pressing the key shortcut Ctrl + F;
  - By mouse;
  - When you start to type in a table.
- View Status position- describes where the View Status is displayed: what fields were searched for and what values were searched for in each field. Can take values: Auto, None, Top, Bottom.

If the property is set to None, the view status will be absent on the form. Consequently, to determine whether a search has been completed or not will be possible will only be possible by the availability of the Cancel Search button.
If the property is set to Top, the view status will be placed between the command bar of the list and the table. If the property is set to Bottom, the view status will be placed immediately below the table.
Auto is interpreted as Top.

- Search control position - determines where the search control button will be placed. The button opens a menu that contains the following information: Find by Current Value, Advanced Search, Cancel Search, Set Time Period and search history (last 5 queries). Set Time Period is available, only when a table displays a dynamic list, where a document table or log is defined as its primary table. The property can take the following values: Auto, None, Command Bar.

  If the property is set to None, the search control button will be absent on the form (but the commands will be available through the More menu). Command Bar property value places a button on a command bar associated with the table. Moreover, the said button is placed to the right of the search line, if it is available in the command bar. If no search line is displayed in the command bar, search management button is available in the command bar in the ordinary way.
  Auto is interpreted as Command bar (if a table displays a dynamic list) or None in any other case.

If there are several command bars on the form, with a single table of client application form being the source of their commands, a search line and search management button will be placed in a single command bar only:

- Either in the command bar of the table itself (if auto-complete is enabled for it);
- Or in any of the rest command bars.

7.7.10. Special command bars

In addition to creating the customized command bars (Form item Group of the Command barkind), a form editor provides an ability to work with the specialized command bars:

- The Form command bar - The system provides a command bar of a form. Its visibility, position and content can be controlled. The visibility and position of a bar are controlled by the Alignment command bar property and the content is controlled by the AutoFill property. In addition to the standard commands that can be added to the command bar of a form, the commands

from the section Form command panel of the global command interface are automatically added there.
If the main attribute of a form is a dynamic list and the AutoFill property is set on the command bar of a form, then the commands provided by this dynamic list will be automatically added to the form. If a form also contains a form item displaying a tree, then the form command bar will be supplemented with commands that are provided by the attribute associated with that item.

- Command bar of the table - The command bar is automatically provided by the system for the form item of the Table kind. Its visibility, position and content can be controlled. The visibility and position of a bar are controlled by the Alignment command bar property and the content is controlled by the AutoFill property.
- The context menu of an item - There is a possibility to modify the standard context menu of a form item. For this, the context menu must be displayed in the tree of form items using the context menu command Display context menu of the form item.
  If a Group view item is added to the context menu of an item and specify another form item as the command source for this group, then the added group will be filled with commands (in the 1C:Enterprise mode) using the same rules as the context menu of an item selected as the source command. When using the context menu, it should be remembered that if the auto-fill is enabled for the context menu, then such a context menu cannot be disabled from the 1C:Enterprise language.

In all of the above command bars, one can add his own commands that are created using the command editor.
If there is a need to place the commands located in the command bar of the form in any command bar, then the following needs to be done:

- Place in the command bar the item Group - Command Bar.
- Choose from this group as a source of commands Global commands of the form command bar.

If there is a group for which the source of commands Form in the command bar or context menu containing a group with the source of commands Global commands of the form command bar has been set, then when filling this group the commands related to the form command interface are not added.
When installing (from the 1C:Enterprise language) as the current item of a button in any command bar, it should be remembered that a focus will be on the first button of the command bar and not on the button used in the operation of setting the current item: `ThisObject.CurrentItem = Items.CommandBarButton;`. Also, it should be remembered that the property button `DefaultItem`is ignored when a button is placed in the command bar. If a user moves a button from the command bar to a form, then this property begins to be used.

If the following changes are made to a form in the process of calling the context server: changed data structure; added or removed form commands; added, removed or moved form items; parameters of the tables of the controlled form are changed, then the standard command bars and groups will be refilled. Further:

- all the user commands inserted into such bars and groups will be deleted;
- modified parameters of standard commands will be set to default values.

7.7.11. A conditional design of a form

A form allows for setting a conditional design of the form items. Configuring the conditional design is performed in the same way as setting the conditional design of the data composition system. A call for setting the conditional design is available from the properties panel of the root item of a form. It is not recommended to use the conditional design of a form for designing dynamic lists if such design can be obtained by setting the conditional design of the list itself.
The following kinds of conditional design are available:

- **Background color** - allows for changing background color and is used for:
  - Form items: Input field, text document field, table, standard button
  - Table fields: Inscription field, input field, switch field, image field.
- **Text color** - Allows for changing text color and is used for:
  - Form items: inscription field, input field, text document field, image field, switch field, table, standard button, hyperlink, decoration-text, decoration-image.
  - Table fields: inscription field, input field, switch field
- **Font (Шрифт)** - Allows for changing font and is used for:
  - Form items: inscription field, input field, text document field, image field, switch field, calendar field, table, standard button, hyperlink, decoration-text, decoration-image.
  - Table fields: inscription field, input field.

NOTE. Changing the font of a form item performed using conditional design is not taken into account when determining the size of items.

- **Highlight negative** - Allows for setting a selection of negative values. Is used for the following form items: inscription field, input field and the table fields: inscription field, input field.
- **Horizontal position** - Specifies the horizontal position of a value and is used for:
  - Form items: inscription field, input field, decoration-text.
  - Table fields: inscription field, input field.
- **Mark unfilled** - Allows for setting a blank mark and is used for:
  - Form items: input field, table.

- Table fields: input field.

TIP. If management of property Mark unfilled is performed using a conditional design, then it is recommended for the field to be designed to set a property AutoMarkUnfilled property to a value No.

- **Text** - Is used to set a text and is applied to the table fields: inscription field, input field.
- **Format** - Allows for changing a format of the output value and is used for:
  - Form items: A group of the Page kind, a group of the Standard group kind.
  - Table fields: inscription field, input field.
- **Visibility** - Is used only to disable a display of the table fields: Inscription field, input field, switch field, image field. A location of the cells with the display disabled is being filled by changing size of the neighboring cells. An algorithm for stretching cells may vary in different client applications. It is not recommended to use conditional design to hide entire rows in a table. Using conditional formatting to hide rows can have a negative impact on performance and to lead to the incorrect display of the table contents. When managing visibility on a mobile platform, the following feature should be taken into consideration: in case when a visibility of a table field depends on data of the current row, then in absence of the current row a display of the column heading may not correspond to any visible row in the table. The headings will correspond to the current row only when the current row appears in the table for which a conditional design is set. When using a conditional formatting that depends on data of the current row, it is not recommended to display the headings of such a table on the mobile platform.
- **Availability (Доступность)** - Is used only to disable the availability of table fields: Inscription field, input field, switch field, image field.
- **View Only (Только просмотр)** - Is used only to enable the View Only mode for the table fields: Inscription field, input field, switch field, image field.
- **Display** - Is used only to disable a visibility of the table field values: Inscription field, input field, switch field, image field.

NOTE. For a conditional design, the current computer date adjusted for the time zone of the 1C:Enterprise session is used.

If using a conditional design for designing a column of the form table and the condition contains a field of the attribute displayed by the table to be designed, then the current data of the table to which the column to be designed belongs is used in calculating a condition. If a field to be designed is not related to a table, then the current data of the first form table (in the order of placement in the form editor) is used to calculate the condition.

Conditional design allows for setting some kinds of design of the form items depending on the values of the form attributes and the design of a form will be dynamic, that is, it will take into account changes in the form data. To change a design of the form does not require any special action. If for a conditional design of a form the dynamic list fields are used, then for these fields the Use property should always be set
Let's review an example of a conditional design.
Suppose we need in the document Consumption of goods to select those rows of the tabular part, in which the number of items is less than 10 pieces.
To do this, it is necessary to specify the following conditions in a conditional design of aform:

- A conditional design of the text color kind-. For the color value, select Custom text (Особый текст) from the style.
- As a condition, let's select the following expression: Object.Goods.Quantity Less "10". In this expression, the Object - The main form attribute, Goods - tabular part, Quantity - An attribute of a tabular part.
- As editable fields, select the entire tabular portion of Goods.

Thus, a font color of the first row of the table part became red since the quantity of goods in this row is less than 10.
However, it is not always necessary to highlight the entire row with a special color; rather it is necessary to highlight a cell of the table part bearing quantity.
To do this, let's change the specified conditional design: as editable fields, let's select not the entire tabular part Goods but a specific field: Quantity
In this case only single field will be highlighted.

It should also be noted that it is possible to use the values of columns that are not related to information database data in the condition expressions.

7.7.12. Features of the form items conduct

If the field headings or decorations of the Inscription kind use the alphabet characters not supported by the font set for these form items, then an incorrect calculation of the length of the rows to be displayed is possible. As a consequence, - the rows in these form items may not be displayed completely.
The managed form items associated with the standard attributes `Reference` and `Conducted` are not available for editing.
If there is a need to use in the solutions the fonts other than the one defined by the Auto value of the corresponding property, it is recommended to limit yourself to the following fonts:

- The true fonts of the 1C:Enterprise system:

    - Small text font,
    - Standard text font,
    - Large text font,
    - Very large text font.
- Vector fonts out of the "MS Core Fonts" package:
    - Andale Mono,
    - Arial,
    - Arial Black,
    - Comic Sans MS,
    - Courier New,
    - Georgia,
    - Impact,
    - Times New Roman,
    - Trebuchet MS,
    - Verdana,
    - Webdings.

Using other fonts does not guarantee correct display of form items in various client applications or operating systems.

A behavior of the form fields of the inscription, image, decoration-inscription, decoration-item kinds does not depend on the setting of the `ReadOnly` mode (both for specific form items and for a form as a whole).

If the properties `Length` and Accuracy for the form attribute of the Number kind are set to the same values, then the input field allows for entering any number but when the input field is intended to be left, a message will be received about incorrect input if the entered value is not in the range from -1 to 1 (excluding - 1 and 1).

If the actions that lead to server call and form rebuilding are performed in the handler `OnChange` of the field located in a form table, the event handlers of this table `OnEditEnd` and `BeforeEditEnd` will not be called.

A tabular document located on a form, vertically scrolls line by line.

If the input field or the inscription field has the Height property set to a value greater than 1 and the Stretch vertically property is set to Auto, then this value will be interpreted as `True`; that is, the specified form items will be stretched vertically.

A drop-down list of the input field adjusts its width to the width of the items displayed in it. The minimum width of a drop-down list is defined as the maximum value of the following values:

- fixed value specified in the platform;
- a value of the `DropListWidth` property of the extension field of a managed form for the input field.

If one of the items displayed in a drop-down list is wider than the calculated minimum width of the list, then a width of a drop-down list is equal to a width of the widest displayed element, but not greater:

- screen size for thin client;
- size of a form for web client.

In the event that when opening a form, a dynamic change in the visibility of a form item is performed, then in the opened form, the input focus will receive an item following the item for which a visibility change was performed. If there is a need to set the input focus to another item -, it should be done independently.

7.7.13. Rules for placing form elements

7.7.13.1. General information

Form items are placed in a rectangular logical grid. This grid is hierarchical, with the uppermost level of a logical grid being the grid specified for a form, and in the logical grid cell its logical grid can form if the form item that is in this cell is a group of the Normal group kind or a group of the Page kind. The actual form, a group of the Normal group kind and a group of the Page kind will be called the **form grouping item**. A width of a logical grid cell is determined by a width of the form item located in this cell, and its height - by a logical string. A concept of "logical string" is not described by certain numeric parameters. The easiest way to show this concept is by example.

The following items are visible in the editor:

- A form itself with the vertical grouping installed and the form command panel disabled.
- Group without display Group1 with a horizontal grouping. The items Element1 and Element3 are placed in this group.
- Group without display Group2 with a horizontal grouping. The items Element2 and Element4 are placed in this group.

For some reduction in a size of the image, for the table Element3 a height of the table is reduced. This change does not affect an operation of the item allocation algorithm (in this example).

The two rows of the logical grid are displayed. The first row contains the items Element 1 and Element 3 and the second - Element 2 and Element 4. It is easy to notice that each of the described rows will have a different height. But, nevertheless, exactly two rows will be used in the form logical grid. Thus, a height of the logical grid row is calculated by a height of the tallest item of all that are located on the same horizontal row. For the first row, a height will be determined by a height of the Elements 1 and Element 3 items and for the second row - by a height of the Elements 2 and Element 4 items.
It should also be noted that each row of a form has its own cell width. This is because the items Element 1 and Element 3 are located in the same grouping

element and the Element 2 and Element 4 -in another grouping element. Within each grouping element the items are placed separately.
Thus, it is clear that in this example there are three logical grids:

1. A logical form grid. For this example, a 2 x 1 grid is set: two items (groups) with vertical grouping specified for a form.
2. A logical group grid with items Element 1 and Element 3. This group defines a 1 x 2 grid: two items (Element 1 and Element 3) in a group with horizontal grouping for a group. This grid is nested in a logical form-level grid.
3. A logical grid of a group with the items Element 2 and Element 4. This group defines a 1 x 2 grid: two items (Element 2 and Element 4) in a group with horizontal grouping for a group. This grid is also nested in a logical form-level grid.

In future, when mentioning the placement of items, a logical grid described above will be assumed. When developing a form, there is a need to adhere to the following simple recommendations that will ensure the operability of the developed form on all supported screens and resolutions:

- If possible, use default values for the item properties.
- Minimize the explicit indication of the size of form items.

In overall, the rules for placing form elements can be defined, as follows:

- The items of a form are placed in two directions:
  - Horizontal direction - Each new item of a form is located to the right of the previous one. Such a group can be represented as a 1 x N matrix, where N - is a number of items in a group.
  - Vertical direction - Each new item of a form is located below the previous one. Such a group can be represented as a matrix N x 1, where N - is a number of items in a group.

  A placement direction depends on the Grouping property of the form grouping element: form or group. When placing no restrictions on a number of the displayed items.
  Thus, by combining groups with different grouping settings and putting groups into one another, quite complex (in structure) forms can be created. At the same time, the system provides an ability to control the alignment of items both within one grouping element and between them.
- An order of displaying items is determined by the order in which they appear on the bookmark Elements of the form editor.
- It is possible to influence a placement of items using the following parameters:
  - Horizontal and vertical spacing of the items;
  - Size of the items;
  - Alignment of the items;
  - An ability to stretch items horizontally and vertically;
  - etc.

- The area of placing items is limited by size of the window in which a form is formed. A window size is limited by size of the working area of the main window or screen settings. If as a result of placement the cumulative size of elements exceeds the size of the window allocated to a form according to the corresponding coordinate (vertical or horizontal), then scroll bars will appear in a form.
- The rules for placing items within a group are equivalent to the rules for placing inside a form.

Placement of items is simultaneously influenced by both general placement rules (described above) and the features of processing various parameters that affect the placement. A total number of combinations of all the influencing parameters is extremely large so an operation of sets of similar properties will be discussed further. Combinations of all influencing properties is not going to be considered. A description of the properties that affect a location of items as well as placement algorithms will be given later in this section.

A formation of the form, as it is displayed to a user, consists of two stages:

1. On server:
   - The system retrieves a tree of form items.
   - Items that are disabled by functional options, access rights and user (using the form settings dialog box) are excluded from the item tree.
   - The resulting tree of items is used to form a description of a form - the internal representation of a form which is not yet suitable for direct display. In this description, actual values for the properties are set to `Auto`. No other calculations are currently performed (sizes of the form items, visibility of the elements, etc.)
   - The resulting form description is passed to client.
2. On client:
   - The client application determines characteristics of the screen on which a form will be displayed: orientation, sizes, fonts used, etc.
   - Based on the information received, sizes of the form items are determined, visibility of the items is processed, etc. Groups and tables for which no visible subordinate items are left are also excluded from the form's description. The result is a description of the form which almost completely corresponds to how a user will see it: items are placed in the cells of a grid used, sizes of items are calculated. In this case it should be understood that a size of items depends on the current font, features of an item itself and the client application used.
   - The resulting “almost form” is converted into a real set of the control items of the target platform (thin client, thick client, web client, mobile device) and displayed to a user.

Features of placing form items in compatibility mode with version 8.3.6 and below are discussed in the 1C:Enterprise documentation for version 8.3.6 (http://its.1c.ru/db/v836doc/bookmark/dev/TI000000404).

7.7.13.2. Groups and the merging form items

The form items can be combined into groups for which a special form item Group serves. Grouping allows for simplifying management of items, for visually selecting items on a form and for controlling the grouping of items (using the Grouping property). Several types of the Group form item are available, each of which has its own particularities in grouping control:

- Command bar - Control of grouping is not available.
- Page - Allows for managing a grouping of items.
- Standard group - Allows for managing a grouping of items.
- Group of columns - allows for managing grouping of items (columns of a form table).

For a group of the Standard Group or Page a grouping (the Grouping property) of items can be set. Let's describe an example of using such a grouping.

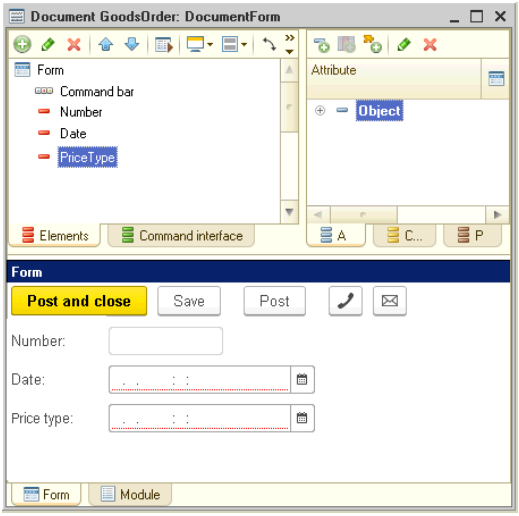


Fig. 146. A vertical grouping

Let's consider creating a document form. In this case, a grouping of form items is selected as Vertical but we want such document attributes as Date and Number to be located not in two rows but in one.

To do this, we will create a group named DateAndNumber, set the Grouping property to the Horizontal value and place the form items Number and Date there.

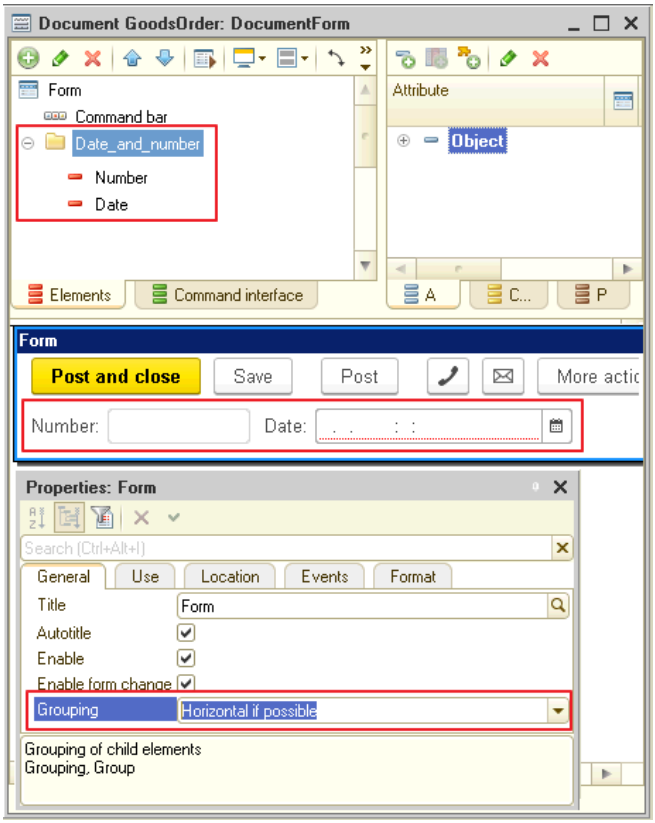


Fig. 147. A horizontal grouping

A combination of the above methods is the Horizontal if possible method. In this case, the items will be arranged horizontally if the dimensions of the parent grouping item allow for placing all the subordinate items horizontally. Otherwise, items will be placed vertically. This placement is performed by the system automatically.

It is possible to change the arrangement of columns in the form table in a similar way. As an example, consider the form of a list in which there are columns Code and Article.

However, in our list it is necessary to make so that the code and article are located in two rows, one under the other. To do this, there is a need to create a group of

the columns subordinate to the List item and give it the name CodeAndArticle (for example). Then for the columns group a vertical grouping should be set.

But there is a need for the code and article to have a common heading (for example, Codes). To do this, follow these steps:

- For a group of columns CodeAndArticle, to set the Heading property;
- Then for the same group to set the property Display in heading;
- A final step is to set the Display Heading property.

Now it is necessary to make so that headings of the columns Code and Article are not displayed on a form (so that a height of the table heading does not increase). To do this, follow these steps:

- There is a need to highlight the items Code and Article;
- Clear the check box Display in heading. This operation can be done for each item separately (without selecting two items at once).

Thus, within a framework of the table it is possible to flexibly change the arrangement of columns and their design in order to obtain the desired look of a table.

NOTE. It should be noted that groups can have unlimited nesting. In this case, the Grouping property can be different for each nested group.

7.7.13.3. The sizes of form elements

When determining actual size of each item of a form, the system is guided by the limitations that the application developer has established during development of a form. It is possible to control the following item properties:

- To control a width of an item:
  - Width - A width of an item.
  - AutoMaxWidth - Is a flag warning about the system automatically calculates maximum width of this item. Does exist for some items.
  - MaxWidth - Is the maximum item's width specified by the application developer. Does exist for some items.
- To control a height of an item:
  - Height - A height of an item.
  - AutoMaxHeight - Is a flag warning about the system automatically calculates the maximum height of a given item. Does exist for some items.
  - MaxHeight - Is the maximum item's height specified by the application developer. Does exist for some items.

- It should also be mentioned that the linear dimensions of an item are influenced by values of the `HorizontalStretch` and `VerticalStretch` properties.

The width and height of a form item is specified in abstract values depending on the font set for this element, taking into account the particularities of a display of this item and a client application. It should also be remembered that when explicitly specifying a width (or height) of an item, the actual dimensions will depend on the operating system used, its settings, and a client application. Besides the fact that actual dimensions of items depend on the kind of client application (thin client, thick client or web client), these dimensions also depend on a web browser used and its version. It should be understood that width and height of a form item do not describe a dimension of the data area of an item. These parameters describe the overall dimensions of an item excluding heading, command bars, etc.

The actual dimension of a form item in the 1C:Enterprise mode may not coincide with the specified dimensions. When determining dimensions of items the system takes into account font (and its sizes) specified in the properties of form items.

NOTE. Changing the font of a form item performed using conditional design is not taken into account when determining the size of items.

Calculation of the dimensions of an item along any axis depends on the state of stretching along this axis:

- Stretching is disabled:
  - The explicitly specified property `Width` (`Height`) determines a size of an item. In this case, properties defining the maximum dimensions of an item are ignored.
  - The property `Width` (`Height`) is set to 0. This means automatic sizing of an item. In this case, dimensions of an item according to the corresponding gauging is determined by the content of an item. At the same time, a size of an item cannot exceed the maximum size. A maximum size of an item along the corresponding axis can be specified either automatically calculated (`AutoMaximum...` properties) or explicitly set manually (`Maximum...` properties).
- Stretching is enabled:
  - The explicitly specified `Width` (`Height`) property determines the “weight” stretching / compression ratio.
  - The property `Width` (`Height`) is set to 0. This means automatic sizing of an item. The system will select the appropriate dimension based on the data type, content and other parameters. At the same time, a size of an item cannot exceed the maximum size. A maximum size of an item

along the corresponding axis can be specified either automatically calculated (`AutoMaximum...` properties) or explicitly set manually (`Maximum...` properties).

- If the maximum width or height of an item is set to 0, then the item can be resized unlimitedly according to the corresponding gauging.
- When determining a width of the input field, the following features apply:
  - If the `MaxWidth` property is set explicitly (the `AutoMaxWidth` property is disabled), then it defines the maximum width in characters. In this case, for example, the input field buttons fit this width.
  - If the `AutoMaxWidth` property is enabled, then the maximum width is determined depending on the type of input field, and the buttons are also arranged there for better alignment with the right border. The `Width` property in this case works differently: if it equals 0, then the width is determined depending on a kind of input field, and a number of the enabled buttons is added to it.

When determining dimensions of items, the limiting factor "from above" is the dimensions of the parent grouping items: Forms, groups, pages. The dimension of a grouping item limits the resizing of nested items. In other words, if a group has a given width and stretching is forbidden, then for all the subordinate items the specified group width will be a constraint when calculating a width of these items.

7.7.13.4. Stretching groups

Changing dimensions of a normal group (stretching) depends on the items in a group and on values of the Stretch Vertically and Stretch Horizontally properties:

- Auto - Does not impact on the stretchability of the children items. If a group itself has displayed design items, for example, a vertical or horizontal line or a background color is set for it, then the corresponding items will have dimensions corresponding to the largest child and stretch within the same limits.
- Yes - Does not impact on the extensibility of the children items. The items that correspond to a group itself (design items, an area with a background color) or empty space in the absence of such items will stretch without restrictions. If a group has the Alignment of items and headings property set to Elements right,..., then such items will move along with the right border of a group if the items themselves cannot stretch.
- No - The group dimensions are determined by the standard dimensions of the children items. Within these dimensions, the children items can stretch. Further stretching / compression of the child items is not possible. The same dimensions will be used for the design items of a form itself and its background color. If a group is given a width (height), then its dimensions

along the corresponding axis are defined as a maximum between the standard dimension of the child items and the specified value.

7.7.13.5. Width and height of groups

If a group has an explicit width or height, then when calculating the dimentions of the child items, an attempt is made to fit them into a given constraint unless the superior group or the dimensions of the target device have imposed a more restrictive constraint.
If, as a result of a calculation, the width of the child items is less than the specified width of a group, then a group expands to the specified value.
If as a result of a calculation the width of the child items is greater than the specified width of a group, then the width of a group is ignored. If calculation is performed in the Designer and the Show location errors mode is enabled, then the corresponding error is displayed in the message window.
If a group has a displayable heading, then it is calculated according to the same rules as these for the child items- If a group has a given width, then an attempt is made to fit the heading into it by wrapping rows. If the width of header is still turned out to be larger than the width of a group, then the width of a group will not be less than the width of a heading (larger if the width of the child items was greater).

7.7.13.6. Controlling the distance between items

For some specific forms, it may be necessary to place items more densely or, conversely, more sparse. Such a requirement may arise for the entire form as well as for some of its fragments. Moreover, to change a distance between the items may be required both vertically and horizontally.
To solve this problem, the grouping elements have the properties Horizontal spacing and Vertical spacing. By changing these properties, the required density of items can be achieved.
The `Auto` value set for any of these properties means that the spacing will be determined by the parent grouping item. For a form, the `Auto` spacing value is determined by a platform.

7.7.13.7. Alignment of the form items

7.7.13.7.1. General information

A form element consists of the **data area** - Form item area in which data is displayed and **Heading area (heading)** - Form item area that displays a heading (see fig. 148). **Standard width of the data area** determines a width of the data area and has the following properties:

- it is determined by the data type of the attribute associated with the form item or the Width property of the form item;

- does not depend on the properties of the form item Expand horizontally.

Fig. 148. Structure of the form item

When performing alignment, the orientation of headings and data areas is formed relative to an imaginary line that runs along the border between headings and data of all items located in the grouping item. A size of the heading area is calculated by the maximum width of the heading of all items.

7.7.13.7.2. Alignment in the group borders

Within the grouping item, the alignment of borders of the components of a form item is determined by the Alignment of Elements and Headings property. This property allows for setting the following alignment options:

- No. In this case, the data area is located immediately after the end of heading. No alignment is carried out

- Items left, headings left. This is the alignment mode that was used in 1C:Enterprise version 8.3.6 and earlier. In this case, the field headings are aligned on the left border of the grouping item. The data areas are also aligned on the left border.

- Items right, headings left. In this case, the field headings are aligned on their left border, and the data - on the right border.

- Items left, headings right In this case, the headings are aligned on the right border, and the data - on the left.

- Items right, headings right. In this case, all parts of the form item are aligned on the right border of "their" area.

As mentioned above, the alignment of borders is carried out within the current grouping item. In this regard, the resulting alignment of items may not be quite satisfactory.

However, in a particular case, it may be necessary to align the check boxes in the ABC classification group with respect to the same imaginary line as the first three items (Code, Name, Data). To control this behavior, there is the End-to-End Alignment property (available for a group of a managed form). If the ABC classification group has this property set to Use, then the alignment will somewhat change.

It can be seen that the ABC classification group began to use the same line as the first three items for alignment. This behavior applies only to the first left items which are located in a group. The alignment itself is implemented by setting the same width for the items headings which (headings) are located on the left. If this property is set to Auto, then the end-to-end alignment is determined by a platform itself. Explicitly specifying the value Not to use for this property will result in such a grouping item that the alignment will not take into account the items that are not included in a group.
The end-to-end alignment is not used for groups whose Behavior property is set to a value other than Standard.

7.7.13.7.3. Alignment between groups

When developing complex forms problems may arise related to the mutual alignment of the form items if these items are located in different groups.
Suppose there is a need to create a form that contains check boxes for selecting actions and explanations for these actions. The check boxes are implemented by check boxes themselves (form items) and the explanations - by decorations of text kind. The check boxes are located in one group, explanations - in another. To simplify an example, a display of headings and a selection of the groups themselves are disabled for groups.
An example clearly shows the problem: a vertical alignment of the items in groups does not correspond to the desired - Decoration clearly have a smaller height than the check boxes.
To solve this problem, the property of the group form United group is intended. Consider how this property works.
When a group has this property set, then in this case the whole group is considered to be a united item (this is the way the mechanism of placement of the form items is configured). This property is set to this namely value when creating a new group.
If a group has this property turned off, then the item allocation mechanism will analyze all the items of such a group as separate items. So, if this property in a group of GroupActions is disabled, then the form will take the following form:

The main question is this: Why did the first row of the item GroupActions for height took as much hight space as the whole item of GroupDescriptions?
As mentioned at the very beginning, a unit of measurement horizontally when placing form items is a row. A hight of a row is defined as the maximum height

of one of the form items that are located along the same horizontal line and which have upper borders (assuming that the Vertical position in the group property of all the items is set to the same value). Until both groups had the check boxes Combined group, each group was perceived by the item placement system as a single item. Therefore, the groups on a form were located at the same level and within each group the alignment worked in its own way.
After the check box GroupAction was cleared, the United group was no longer perceived by the placement mechanism as a whole. Each item of a group is counted separately. But for the GroupDescription item, the Combined Group check box is still set. This means that the allocation mechanism assumes that the Action1 Item and the GroupDescription item are located on the same row and the height of this row corresponds to the height of the GroupDescription item.

The result is the desired behavior: Dimension of decorations has ceased to influence the overall alignment of items.
Thus, using the United group property, it is possible to create groups consisting of several columns and rows simultaneously, with support for alignment of elements in all columns and rows:

- The property is set - A group is considered a single item;
- A property is reset -Every item that is part of a group with a property dropped is considered a separate item.

In addition, the group for which the United group property has been reset (the group becomes “separated”) has the following features:

- Of all the properties of a group, only the following properties are used: Grouping, Background color, Display headingand Display (to define the parameters for defining a group heading). The Grouping property is used to determine the order in which the items of a current group are placed in the parent grouping items.
- If the Display heading property is set to `True`, then the group heading is placed in the top-level group as the first item while grouping is observed, that is, for horizontal grouping a heading will be placed to the left of the other items, for the vertical one above the other items.
- The separators are not formed.

7.7.13.7.4. Controlling the relative position of items

When developing various forms, it becomes necessary to control the relative positioning of form items. For example, it is necessary that one or several buttons be located in the center of a group or form.
To organize this location, there are several properties:

- A Horisontal position of subordinates. Available for the grouping items (form, regular group and page type group). Defines what the horizontal relative position of the subordinate items within the grouping item will be.

- A Vertical position of subordinates. Available for the grouping items (form, regular group and page type group). Defines what the vertical relative position of the subordinate items within the grouping item will be.

Below is an example, when for a group with horizontal items the buttons are located in the center of a group and for a group with vertical items the buttons are located at the bottom of a group.

In the event that any of the described properties is set to Auto, this means that the actual value of the property is determined through the properties of all parents of the same name, and for a form the value of a property is automatically selected by a platform. The bypassing of the parental items will end when the desired property is set to a value other than Auto.
It should also be remembered that if a value of the Horizontal Position property is different from Left for the grouping item, then the alignment of headings stops working for the subordinate elements.

7.7.13.8. Dividers of items

In some cases, a platform can form dividers on a form. These are special form items with which the adjacent form items can be resized.

The divider is automatically formed when placing the form items under all the following conditions:

1. Both separable items are stretched in the appropriate direction.
2. There are no other items between the items to be separated (they are located side by side).
3. The height of each of their shared items is at least two rows.
4. A property Value `Width` (or `Height` depending on orientation of the divider) of each of the separable form items is greater than the minimum width (height) and less than the maximum width (height) of the form item. Maximum dimensions are set using the `Maximum...` or `AutoMaximal...` properties. The minimum width and height are automatically determined by a platform.
5. None of the items to be separated is a caption with an auto-wrap text.
6. The dimension of the corresponding spacing between the items of the parent group (or form) is sufficient to accommodate the divider.
7. A parent group (or form) does not have any child items of the Group kind of the Standard group type with the United property turned off.

7.7.13.9. Adaptation of a form to the mobile client

The mobile client interface is similar to the mobile platform application interface. This means that one form is displayed on the screen at a time, the screen has a relatively small width. As a rule, horizontal scrolling of a form is not applied on mobile devices. At the same time, vertical scrolling of a form is quite acceptable.

Based on these prerequisites, the mobile client tries to automatically rebuild a form so that it does not fit a screen of a mobile device. In the process of adaptation, a width of the form items is reduced for those items whose width in the configuration is set so that it exceeds the actual screen width on a mobile device. A width for such items is formed in such a way that an item fits (in width) in the current orientation of a mobile device without horizontal scrolling. In addition to changing width, the system can attempt to reduce a number of the displayed form items in order to improve the visual appearance of a form. To control a need for such a restructuring, the form property is Option of minimizing form items by importance (`CollapseItemsByImportance`). The `Auto` value is treated as `Use`.

In order to make it easier for the mobile client to adapt a form, form items (and columns of tables) have the Importance at displaying property (`DisplayImportance`). By changing a value of this property, a developer can achieve the desired kind of form on the device screen. A value of this property is handled by the mobile client as follows:

- More important items are given more space on a form.
- If the more important item is located below the less important ones and these less important items occupy more than three rows of a form, then the less important items are combined and placed in a collapsible group. It is important to understand that here and further a height is important but not a number of items. In other words, the three rows of the form grid can occupy both three items with a height of one row each as well as one form item occupying three rows of height.
- If the first or last of the collapsible items is the command bar, then it does not collapse as it can refer to an important item.
- If the more important item is located within the hierarchy of groups or pages and at the same time in different places of the hierarchy, the less important items are located above it occupying more than three rows, a collapsible group is formed in each level of hierarchy.
- If there are several items of higher importance, then less important items between them are placed in a collapsible group if less important items occupy more than three rows.
- Less important items that are below the last more important item are not collapsed.
- If the items of different importance are also located inside a collapsed group, the algorithm is applied recursively to this group.
- The heading of a coagulated group is formed according to the following algorithm:
  - If an already existing group is being collapsed, then the heading of this group is used.
  - If a single item with a specified heading falls into a collapsible group, then the heading of this item is used.

- If none of the above items is appropriate - Heading is formed by combining through "," all the headings of the items included in the collapsible group.

If for the form item the `DisplayImportance` property is set to `Auto`, then the actual value of the property will be determined by the following rules:

- Value `VeryHigh` for the form item displaying data from the main form attribute (and the subordinate atributes).
- A value `High` if the item is stretched vertically and is one of the following items:
  - A form table displaying a dynamic list;
  - A field of a tabular document;
  - A field of chart of any kind and type;
  - formatted document field;
  - An HTML document field.
- If an item is a command bar, then:
  - A value is `High` if the source of commands for the command bar is the controlled form itself.
  - The value of importance of an item that is the source of commands for the command bar.
  - The importance value of the most important item in the same group as the command bar provided that no command source is set for the command bar.
- The value `Standard` for all the remaining items.

7.8. Form command interface

A command interface of a form consists of:

- from the navigation bar of a form,
- command bar of a form itself.

There is a possibility to edit the command interface of a form: inclusion of new, use of existing commands, etc.

Note that the command interface of a form is edited on a separate bookmark in the form editor and describes a composition of the navigation panel of the auxiliary window in which the form is displayed, and partly a composition of the command bar of the form. Basically, this mechanism is needed to configure the global interface commands that need to be displayed on the form in these two panels. A composition of the form command bar is defined both directly in the structure of items and in the editor of the command interface.

The command automatically enters the command interface of the form in case the parameter kind of the parameterized command matches the kind of the main attribute of the form.

In addition, there is a possibility to forcefully add a command to the desired command interface bar. To do this, simply drag it into the desired group of the desired command bar.
The installed AutoPosition check box indicates that the command sequence generated by the system by default will be used. If it is disabled, the order of commands can be edited.
In order to configure visibility of the commands that are located in the command interface bars, there is a possibility to uncheck the box in the Autovisibility column and change value in the Visibility column.

7.9. Working with a form from the 1C:Enterprise language

7.9.1. Opening a form

In order to open a form, there are two ways:

- Use the method `OpenForm()`.
- Use a combination of the `GetForm()` method and the `Open()` method of the `ClientApplicationForm` object.

In any of these cases, there is a possibility to pass the form parameters to the form to be opened.
It is recommended to use the `OpenForm ()` method in all cases, except for a need to open a form in a modal mode and then get the result of the functioning form working through the attributes of the form being opened.
This feature is due to the fact that the return value of the modal form will be the data that the form itself returns and a developer will not have an access to the `ClientApplicationForm` object to get the form attributes. If one first gets a form using the `GetForm ()` method, there will be a possibility to get an access to the attributes after the `Open ()` method completes. If it is necessary to open the form in blocking mode, then for this you should use special approaches.

Example 1:

```
// Opening a list form of the catalog of goods
// in the "View only" mode
Parameters = New Structure("ReadOnly", True);
OpenForm("Catalog.Goods.ListForm", Parameters);
```

Example 2:

```
// Open modal form including after closing
// access form attributes
Form = GetForm("GeneralForm.SelectionPeriod");
Result = Form.Open()
```

NOTE. It is not recommended to perform software opening and closing of the same form in one call to the 1C:Enterprise language.

It is not recommended to place in the event handler of the `OnOpen` form the program code for opening any other form as this may lead to a violation of the window display order. In this case, it is recommended to use a wait handler for a short time interval or open other forms interactively, for example, by pressing a button. It should also be remembered that the exceptions that occur during opening of a form are not intercepted in the 1C:Enterprise language and are displayed to a user "as is".

7.9.2. Redefinition of the form being opened

In application systems, the problem of opening various forms with different parameters of an object (or current environment) may arise. For example, for a catalo item of the `Products` there is a neeed to open different forms for actual product and service. Or for some automated workplace, there is a need to open different forms depending on current settings of a user.
To perform such a task, in the object manager module an event handler `FormGetProcessing` should be created In this processor, there is a need to perform all the necessary checks and decide which form should be opened. This processor is called only if an attempt is made to open a standard form of an object using the default form names.
So, at opening a form of the catalog item `Products` in the following case the processor will be called.

```
OpenForm("Catalog.Products.ObjectForm");
```

However, a processor will not be called if someone tries to open a form, as follows.

```
OpenForm("Catalog.Goods.Form.ItemForm");
```

Consider an example with different forms of the directory item form in more detail.

When a form is opened, the analysis of properties of the attribute View of the Products catalog will be performed and if the value of this property is equal to the Enumerations.KindOfGoods.Servicesvalue, open the form FormServices of the catalog Products.

```
A procedure FormGetProcessing(FormType, Parameters, SelectedForm,
AdditionalInformation, StandardProcessing)

  If FormType  = "FormObject" And Parameters.Key.View =
Enumerations.KindsOfGoods.Service Then

    SelectedForm = Metadata.Catalogs.Goods.Forms.FormService;
    StandardProcessing = False;

  EndIf;

EndProcedure
```

This processor should be placed in the Products catalog manager module.

NOTE. Processor FormGetProcessing is not called when trying to open common forms, constant forms, settings storage forms, external reports and external processings.

7.9.3. Modifying properties of the form items

In the process of working with a form, situations arise when it is necessary to change any property of form items, for example, the availability of items.

To perform these actions, use the Items collection. This collection provides access to a list of all form items (without regard to hierarchy).

To access the hierarchy of items, use the Parent and ChildItems properties (for the group items, table, and a form itself).

So, disabling the availability of the form items ViewPrice (ВидЦены) is, as follows:

```
Items.KindPrice.Availability = False;
```

Attention should be paid to a feature of setting the properties ReadOnly, Accessibility and Visibility for the items containing subordinate items. Setting a property for the parent item also affects all subordinate form items. In this case, a value of the variable property of an item itself does not change. In other words, the actual value of the properties ReadOnly, Accessibility and Visibility of a specific item of a form is defined as

the addition “by AND” of the values of these properties of all the parents of the item.
For example, we have a `CurrencyDocument` group, which consists of the `Currency` and `CurrencyExchange` input fields. If it is necessary to make the entire group inaccessible, it suffices to do the following:

```
Items.DocumentCurrency.Availability = False;
```

TIP. When programmatically modifying the properties of form items, the unnecessary modification of properties for which the Change property on client requires accessing server in Syntax Assistant should be avoided. This slows down the work of a form and requires unnecessary calls to server.
NOTE. If modification of the form items leads to a change in the arrangement of items on a form (for example, when visibility of the form items changes), then various event processors associated with activation of items may be called (for example, a table will activate the `OnActivateRow` event).

Programmatically changing the `View` property of a form item that has an extension (field, table, group, decoration) leads to a creation of the new extension of the form item. The properties of the form item related to extension are set to default values and are not transferred from the item extension that was taking place before change to the `View` property.

NOTE. It should be remembered that the properties of the form items related to the displayed data (for example, `DataPath`, `TitleDataPath`, etc.) can be changed only for new form items or after changing view of the form's item.

7.9.4. Form's modification

There is a possibility of the programming modification of a form. For modification (creation, modification and deletion) are available:

- Form's attributes,
- Form's local commands,
- Form's items.

NOTE. Only those objects that are created from the 1C:Enterprise language can be deleted.

A general scheme of the programming work with a form can be represented, as follows:

- changing composition of the form's attributes,
- changing composition of the form's commands,
- changing composition of the form's items,

In particular, it should be noted that a work on adding, changing and deleting parts of a form is possible only on server. You should also pay attention to the

fact that the program model of a form is not affected by the user settings and functional options of a form.

When making custom settings, a user will customize exactly the form that he sees (taking into account program modifications of a form). If a user adds any form items, then access to such items from the 1C:Enterprise language is impossible. This feature should be remembered when developing the user interaction mechanisms. So, for example, if a user adds a page to the form item Groupof the Pages view, then while transitioning to this page, the property of the CurrentPage group will become equal to Undefined. The property of the CurrentItem (ТекущийЭлемент) will also contain the value Undefined in case when the active item of a form is an item added by a user.

Consider in more detail all the stages of a form's modification.

7.9.4.1. Changing a composition of attributes

Changing (adding and deleting) of attributes is performed by the method ChangeAttributes() of the ClientApplicationForm object. At the same time, delete and add actions are performed in one call. This allows for performing an operation to change the form's attributes. It should be understood that the operation of changing the composition of attributes is a resource-intensive operation (in fact, full creation of a form actually takes place), which is why the operations of changing composition of the form's attributes are performed in a batch manner.

Let's consider details of creation attributes.

First, there is a need to create the required number of objects of the FormAttribute kind. When creating an attribute, we indicate its name, kind and the place in the hierarchy of the form's attributes which will be taken by the created attribute.

So, if we create a table of values from two columns, the programming code for creating them will look like this:

```
MyAttributes = Array;
MyAttributes. Add(FormAttribute("TableData", New
TypeDescription("ValuesTable"), , "TableValues", False));
MyAttributes.Add (New FormAttribute("Supplier", New
TypeDescription("CatalogRef.Contractors"), "TableData", "Who
delivers", False));
MyAttributes.Add (New FormAttribute("Goods", New
TypeDescription("CatalogRef.Goods"), "TableData", "Name of goods",
False));
```

It should be noted that for the last two attributes the penultimate parameter indicates for which attribute columns will be created. In other words, it is possible to add columns for details of those kinds that allow it.

On fig. 149 the attributes created in the form editor are shown, the analogue of which creates the code communicated above.

Fig. 149. Equivalent to programming code

After all the attributes planned to be created in a form are really created, a list of attributes should be changed:

```
ChangeAttributes(AttributesToBeAdded, AttributesToBeDeleted);
```

Execution of this code will cause the system to first remove the attributes listed in the `AttributesToBeDeleted` array, and then to add the attributes from the `AttributesToBeAdded` array. After that, a form will be reformatted.
When deleting attributes there is a loss of data they contain; however, if the attributes to be added and deleted have compatible kinds or the attribute to be added differs from an attribute to be deleted only by its properties (but not by kind), the data stored by an attribute will be saved.
After adding an attribute an access to it in the program code is possible only through the construction `This Object.AttributeName`. Expression "`ThisObject.`" Is mandatory for programmatically added attributes.
In order to change a composition or properties of an attribute, first a variable attribute should be acquired. This can be done using the `GetAttributes()` method.
Attention should be paid to the two features of the acquired list:

- This list is not dynamic and does not track changes in details that occurred after calling the method;
- despite the fact that the resulting list can be changed, these changes will not affect the real properties of the form attributes.

After the list of form attributes we are interested in is received, any actions with the received attributes (for example, change the headings of all the attributes) can be performed and then the method of changing the details can be executed. Changing attributes should be done with prior deletion.
For example, if we want to change properties of the `ParameterOrder` attribute, then this should be done in the following way:

```
ArrayAttribute = GetAttributes("ParameterOrder");

...
// Execute change of an attribute
...

AttributesToBeDeleted = NewMassive;
AttributesToBeDeleted.Add("ParameterOrder");

ChangeAttributes(ArrayAttribute, AttributesToBeDeleted);
```

7.9.4.2. Change of the commands' composition

In order to manage composition of the form commands, the `ClientApplicationForm` object has a special collection- `Commands`. With the help of this collection we can add, delete and modify form commands. So, to add a command `InstallCommandStatus` with the heading Set status, which will call the processor with the name `ProcessorProgramCommand` , it is necessary to execute the following program code:

```
&OnServer
The OnCreateAtServer(Denial, StandardProcessing) procedure

  Command = Commands.Add("InstallCommandStatus");
  Command.Action = "ProcessorProgramCommand";
  Command.Heading = "Install status";

EndProcedure

&OnClient
Procedure ProcessorProgramCommand(Command)
  ...
EndProcedure
```

A processor of command must exist in the form module and be preceded by the compiler directive `&AtClient`.

NOTE. A single processor can serve several programmatically added commands.

7.9.4.3. Working with the form's items

After all the necessary attributes and commands are created, the control items can be added.

To manage the form's items, the `ClientApplicationForm` object has the `Items` collection with which one can add, delete, change properties of the form's items and also move the form's items between parents.

The `Items`collection provides access to a list of the form's items which does not take into account the possible hierarchy of items. To work with a hierarchy, the object of the collection `Items` has the `Parent` and `ChildItems` properties.

The first property points to the form's parent item, for example, for a field's form located in a group, the `Parent` property will point to the form's item of the `FormGroup` kind.
The `ChildItems` property exists for those form's items that can have subordinate items. For example, for the form's item of the `FormGroup` kind, the `ChildItems` collection will contain those items that are located in this group.
To move an item from one collection to another (for example, from one group to another), there is the `Move (Переместить)()` method. The parameters of this method describe the item to be moved, the new item's parent and the form's item in front of which the item to be moved is to be placed.
If the last parameter of the method is not specified, then the item to be moved will be placed at the end of the items' collection of the new parent.

NOTE. When programmatically changing a composition of the form's items, it is possible to invoke various event processors related to activation of items (for example, a table will call the event `OnActivateRow`).

Let's take a closer look at how to add a control item.
An example shows how to place two items on a form:

- The form's field associated with the form's attribute;
- The button associated with the form's command.

```
&OnServer
The OnCreateAtServer(Denial, StandardProcessing) procedure

  MyAttributes = Array;

  KindRow = New TypeDescription("Row", , New StringQualifiers());

  MyAttributes.Add(New FormAttribute("ObjectDescription", KindRow, "",
"ObjectDescription", False));
  ChangeAttributes(MyAttributes);

  Command = Commands.Add("ChangeRow");
  Command.Action = "ProcessorProgramCommand";
  Command.Heading = "Change row";

  Item = Items.Add("ObjectDescription", Kind("FormField"));
  Item.View = FormFieldType.InputField;
  Item.DataPath = "ObjectDescription";

  Item = Items.Add("ChangeRow", Kind("FormButton"));
  Item.CommandName = "ChangeRow"

EndProcedure

...

&OnClient
Procedure ProcessorProgramCommand(Command)

  ThisObject.ObjectDescription = "Description of object formed by a
command";

EndProcedure
```

The form in which this code will be placed will look like on fig. 150. A form is displayed after a user has clicked the Edit row button.

Fig. 150. The result of software modification of a form

7.9.5. Working with a dynamic list

This section provides examples of performing some operations on working with a dynamic list placed in a form.

NOTE. The examples below are not complete. They are intended to demonstrate various techniques of handling a dynamic list.

7.9.5.1. Dynamic list query parameter

This section provides an example of setting a dynamic list query parameter. Suppose the following data query is specified for a dynamic list:

```
SELECT
  Goods.Code AS Code,
  Goods.Name AS Name,
  Goods.Article AS Article,
  CommodityInventoriesBalance.QuantityBalance AS Quantity,
  PricesGoodsSliceLast.Price AS Price,
  Goods.IsFolder
FROM
  Catalog.Goods AS Goods
    LEFT JOIN AccumulationRegister.GoodsInventory.Balance(, Warehouse
= &Warehouse) AS GoodsInventoryBalances
    PO (GoodsInventoryBalances.Goods = Goods.Reference)
    LEFT JOIN InformationRegister.GoodsPrices.SliceLast(, PriceKind =
&PriceKind) AS GoodsPricesSliceLast
    PO Goods.Reference = PricesGoodsSliceLast.Goods
```

Then, to set the parameters of this request (in the Processor `OnCreateAtServer()`), there is a need to specify the `Warehouse` and `KindPrice (ВидЦен)` parameters, as follows.

```
// ListGoods - form attribute of the DynamicList kind
// Parameters.Warehouse and Parameters.KindPrice - form parameters of
the corresponding kinds
&OnServer
The OnCreateAtServer(Denial, StandardProcessing) procedure

  GoodsList.Parameters.SetParameterValue("Warehause",
Parameters.Warehouse);
  GoodsList.Parameters.SetParameterValue("KindPrice",
Parameters.KindPrice);

EndProcedure
```

7.9.5.2. Filters

This section provides examples of installing and removing a selection in the dynamic list.

Set

Function adds a selection to the dynamic list and returns the created selection item.

```
Function AddSeelection(ListSelection, FieldName, Value, ComparisonType
= Undefined, Use = True)

  NewItem =
ListSelection.Items.Add(Kind("DataCompositionFilterItem"));
  NewItem.LeftValue = New DataCompositionField(FieldName);
  NewItem.ComparisonType = ?(ComparisonType = Undefined,
DataCompositionComparisonType.Equal, ComparisonType);
  NewItem.RightValue = Value;
  NewItem.Use = Use;

  Return NewItem;

EndFunction
```

Example:

```
// DynamicList - a form attribute of the "DynamicList" kind
// There should be the column "Counterparty"
//
// Setting of the selection by "Counterparty" field, value
// selection is in the variable "CounterpartyReference", selection is
on,
// condition of selection - equal
AddSelection(DynamicList.SettingsComposer.FixedSettings.Selection,
"Counterparty", CounterpartyReference);
```

Delete

A procedure removes a selection of the dynamic list and returns the result of deletion. If the field name with the selection to be deleted is not specified, the selection list is cleared.

```
Function DeleteSelection(SelectionList,FieldName = "")

  If IsBlankString(FieldName) Then

    SelectionList.Items.Clear();
    Return True;

  EndIf;

  Field = New DataCompositionField(FieldName);

  SelectionDeleted = False;
  For Each FilterElement From ListSelection.Items Do

    If FilterElement.Use AND FilterElement.LeftValue = Field Then

      ListSelection.Items.Delete(FilterElement);
      SelectionDeleted = True;

    EndIf;

  EndDo;

  Return SelectionDeleted;

EndFunction
```

Example:

```
// DynamicList - a form attribute of the "DynamicList" kind
//     A list should contain the column "Counterparty"

// Deletion of selection by field "Counterparty" is executed.
Result =
SelectionDelete(DynamicList.SettingsComposer.FixedSettings.Selection,
"Counterparty");
```

7.9.5.3. Grouping

This section contains examples of setting and deleting groups in a dynamic list.

Setting

A function adds grouping to the dynamic list and returns the created grouping item.

```
Function AddGrouping(GroupsList, FieldName, Use = True, GroupType =
Undefined)

  Field = New DataCompositionField(FieldName);

  NewItem =
GroupingsList.Items.Add(Kind("DataCompositionGroupField"));
  NewItem.Use = Use;
  NewItem.Field = Field;
  NewItem.GroupType = ?(GroupType = Undefined,
DataCompositionGroupType.Items, GroupType);

  Return NewItem;

EndFunction
```

Example:

```
// DynamicList - a form attribute of the "DynamicList" kind
// There should be the column "Counterparty"
//
// Creation of a group by the "Counterparty" field is executed,
created
// grouping will be used, grouping kind - items only
Grouping =
List.SettingsComposer.Settings.Structure.Add(Kind("DataCompositionGrou
p"));
AddGrouping(Grouping.GroupFields, "Counterparty");
```

Delete

The procedure removes the grouping of a dynamic list and returns the result of deletion. If the name of the field with the group to be deleted is not specified, the group list is cleared.

```
Function DeleteGrouping(GroupingsList, FieldName = "");

  If IsBlankString(FieldName) Then

    GroupingsList.Items.Clear();
    Return True;

  EndIf;

  Field = New DataCompositionField(FieldName);

  A gropings is deleted = False;
  ArrayItemsForDeletion = NewArray;

  For Each GroupingItem from GroupingsList.Items Do

    If Grouping.Item.Field = Field Then

      ArrayItemsForDeletion.Add(GroupingItem);

    EndIf;

  EndDo;

  For Each DeletingItem from ArrayItemsForDeletion Do

    GroupingsList.Items.Delete(DeletingItem);
    GroupingDeleted = True;

  EndDo;

  Return GroupingDeleted;

EndFunction
```

Example:

```
// DynamicList - a form attribute of the "DynamicList" kind
//
// All groupings in the dynamic list are deleted.
Result =
DeleteGrouping(DynamicList.SettingsComposer.FixedSettings.Grouping);
```

7.9.5.4. Order (sorting)

This section provides examples of setting and removing the ordering of a dynamic list.

Setting

Function adds the sorting field to the dynamic list and returns the created order item.

```
Function AddOrder(OrderList, FieldName, Use = True, OrderType =
Undefined)

  Field = New DataCompositionField(Field);

  NewItem = OrderList.Items.Add(Kind("DataCompositionOrderItem"));
  NewItem.Use = Use;
  NewItem.Field = Field;
  NewItem.OrderType = ?(OrderType = Undefined,
DataCompositionSortDirection.Vozr, OrderType);

  Return NewItem;

EndFunction
```

Example:

```
// DynamicList - a form attribute of the "DynamicList" kind
// A list should contain the column "Name"
//
Sorting is performed on the "Name" field,
// sorting direction - descending, sorting is used
AddOrder(DynamicList.SettingsComposer.Settings.Order, "Name", True,
DataCompositionSortDirection.Desc);
```

Delete

A procedure removes sorting of a dynamic list and returns the result of deletion. If a name of the ordering field is not specified, the ordering list is cleared.

```
&OnClient
Function DeleteSorting(SortingList, FieldName = "")

  If IsBlankString(FieldName) Then
    SortingList.Items.Clear();
    Return True;
  EndIf;
  Field = New DataCompositionField(FieldName);
  OrderDeleted = False;
  ArrayItemsForDeletion = NewArray;
  For Each OrderItem from the OrderList.Elements Do
    If OrderItem.Field = Field Then
      ArrayItemsForDeletion.Add(OrderItem);
    EndIf;
  EndDo;
  For Each DeletingItem from ArrayItemsForDeletion Do
    OrderList.Items.Delete(DeletingItem);
    OrderDeleted = True;
  EndDo;
  Return OrderDeleted;

EndFunction
```

Example:

```
// DynamicList - a form attribute of the "DynamicList" kind
//
// Deletion (not disabling) of the order on the "Name" field is
executed
Result = DeleteOrder (DynamicList.SettingsComposer.FixedSettings.
Order, "Name");
```

7.9.5.5. Using conditional appearance in reports

This section contains examples of specifying a conditional design of the dynamic list. The conditional design uses the dynamic list `List` which contains the fields `Organization` and `Sum`(such as a `Number`). In the example, there are three conditional design items are set:

- if a value of the `Sum` field is less than 500, then the text of the `Organization` and `Sum` fields is displayed in red;
- if a value of the `Sum` field is more than 500 but less than 10 000, then for all fields of a row the text is displayed in blue;
- if a value of the `Sum` field is greater than 10,000, then the text of the `Organization` and `Sum` fields is displayed in green.

The example also uses the selection setup function.

```
Function SetUp.ConditionalDesign(DesignList, NamesDesignedFields,
StructureDesign, Use = True) Export

  NewItem =DesignList.Items.Add();
  NewItem.Use = Use;
  // Let's define the designable fields specified by an array with the
field names.
  For Each FieldName From NamesDesignableFields Do
    DesignableField = NewItem.Fields.Items.Add();
    DesignableField.Use = True;
    DesignableField.Field = New DataCompositionField(FieldName);
  EndDo;
  // Let's setup the design set by a structure,
  // in which - Key: Name of the design parameter,
  // and a value - value of the design parameter
  For Each DesignItem From StructureDesign Do
    NewItem.Design.SetParameterValue(DesignItem.Key,DesignItem.Value);
  EndDo;
  Return NewItem;

EndFunction
```

Example of setting a conditional design:

```
ConditionalDesignItems =
List.SettingsComposer.FixedSettings.ConditionalAppearance;

DesignableFields = NewArray;
DesignableFields.Add("Sum");
DesignableFields.Add("Organization");
// ***** Setting a red color for the sum less than 500
Design = NewStructure;
Design.Insert("TextColor", New Color(128,0,0));
NewItem = SetConditionalDesign(ConditionalDesignItems,
DesignableFields, Design);
// Let's see a condition of design: Sum < 500
AddSelection(NewItem.Selection, "Sum", 500,
DataCompositionComparisonType.Less);
// ***** Setting a green color for the sum more than 10 000
Design = NewStructure;
Design.Insert("TextColor", New Color(0,128,0));
NewItem = SetConditionalDesign(ConditionalDesignItems,
DesignableFields, Design);
Let's set the design condition: Sum > 10 000
AddSelection(NewItem.Selection,"Sum", 10000,
DataCompositionComparisonType.Greater);
// set a blue color if the sum is in the range from 500 to 10 000
Design = NewStructure;
Design.Insert("TextColor", New Color(0,0,128));
// Passing an empty array as a list of editable fields
// indicates appearance of all fields in a row
NewItem =
SetConditionalDesign(ConditionalDesignItems,NewArray,Design);
// Let's create the group of conditions "AND"
SelectionsGroup =
NewItem.Selection.Items.Add(Kind("DataCompositionFilterItemGroup"));
```

```
SelectionGroup.GroupType =
DataCompositionFilterItemsGroupType.AndGroup;
// Let's see a condition of design: Sum > 500 AND Sum < 10 000
AddSelection(SelectionGroup, "Sum", 500,
DataCompositionComparisonType.Greater);
AddSelection(SelectionGroup, "Sum", 10000,
DataCompositionComparisonType.Less);
```

7.9.6. Transferring mechanism

The 1C:Enterprise supports the Drag & Drop function. With its help, there is a possibility to transfer data between different form items. For example, there is a possibility to transfer the catalog list items from one group to another, transfer data from a tabular field to a tabular document field or transfer the list of selected files from client operating system tool for the work with files to any form item.
Transfer operations support the following form items:

- table of a form,
- tabular document field,
- calendar field,
- Image field,
- decoration-image,
- scheduler field.

When dragging, the following terms are used:

- **data source** - The form's item from which data can be transferred;
- **data receiver** -The form's item in which data can be transferred to.

It is possible to allow for or prohibit the control items to provide or receive data, i.e. to be a source or a receiver of data. For this, the form's items have the following properties:

- Allow for starting a transfer - Allows the form's item to provide data.
- Allow for transferring - Allows the form's item to accept data. These properties can be set from the properties palette or from the 1C:Enterprise language.

When a mouse clicked on a selected area of the control item, the `DragStart` event processor is called at the control item - of the data source. If a source of transferring is the scheduler field, then the `DragStart` event occurs when the mouse's cursor goes beyond this field. As the parameters of this event, the objects of the `DragParameters` and `Execute`kinds are transferred. The parameter `DragParameters` have the following properties:

- The `value` - Contains a transferring value, for example, for a managed form table it can be an object reference, for a tabular document, a -tabular document area, date - for a calendar. This property can be assigned its value (for example, some structure), then this value will be a transferable object. For a table of the manageable form, associated with a form attribute such as `FormDataCollection`, `FormDataTree` or

DataFormsStructuraWithCollection, is passed as the transferring value:

- row identifier (or the array of identifiers) if transferring is done within one table;
- the collection item (or an array of items) if transferring is performed between different tables.

When performing a transferring operation from the scheduler field, the Value property contains copies of the items being transferred.

- The Action (Действие) - Indicates the transfer action and is a value of the DragAction kind.
- AllowedActions - Indicates the allowed transfer actions and is a value of the DragAllowedActions kind. Using this property, it is possible to specify which operations are possible with the data of the source data (for example, only copying).

The Execute option allows for enabling or disabling the standard processing of a transfer operation from this control item. For the DragStart event, the standard processing is the start of data transferring.

Next, at the form's item - data receiver the event processor DragCheck is called. This processor is called whenever the cursor hits a new object in the form's item - data receiver (for example, in a new table cell or a tabular document field when it hits a new date in the calendar field). A set of parameters for this event depends on the form's item - data receiver but the first two parameters are the same for all. These are the objects of the DragParameters and StandardProcessing kinds. The remaining parameters describe an object under cursor. When processing this event, there is a possibility to control the kind of cursor, that is, for example, to indicate that transferring into this control item is prohibited or only copying is possible. To do this, there is a need to set the necessary action in the property Action of the DragParameters parameter. It must be kept in mind that the transfer action to be set must be allowed, that is, it must not conflict with the value of the AllowedActionsproperty. For example, the action Copy does not conflict with the value of valid actions CopyAndMove and the value Transfer is in conflict with a value of the valid actions Copy. The StandardProcessing parameter is used to indicate the possibility of standard processing by the form's item of this event. The standard transfer processing depends on the control item:

- For a table, a possibility of inserting a value is checked, that is, a kind of value is checked and if it matches the type of displayed data, the standard actions are performed. The standard actions for the hierarchical dynamic lists - transfer to a group, for the tables displaying record sets or table parts, -changing the order of rows and copying.

If the table's property `ChangeRowOrder` is set to `False`, then at transferring inside the table a standard processing cancel the transfer and the event `Transfer` will not be called.
If the property `ChangeRowOrder` is set to `True` or transfer is performed from outside the table, a standard processing will not cancel transfer.
A standard processing does not cancel transfer if the table's property `SelectionMode` is set to `Multiple`, the `ChangeRowOrder` property is set to `True` or transfer is done from outside the table and more than one item is transferred.

- For a table displaying a tree, the standard transfer processing is, as follows:
  - When transferring tree elements inside one node, the order of allocation of the tree items inside the node is changed.
  - When dragging tree items between different nodes, the transfer of tree items into the selected node is performed.
- For a table document field - verification of a possibility to insert the transferring value.
- There is no standard processing for an image field and calendar.
- When transferring files, the standard action will be an attempt to open the files that are involved in the operation of transfer.

If the mouse button is released, then in the form's item - data receiver the event processor `Transfer` is called. A parameter set of this event is the same as the `DragCheck` event. The `StandardProcessing` parameter allows for enabling or disabling the standard event processing by the form's item. The standard transfer actions are described above.
Then, in the control item - data source the `DragEnd` event processor is called. When processing this event, the form's item - data receiver may, for example, delete the transferred data or clear any variables.
For some form's items (form table, image field, decoration-image), it is possible to specify how the system will behave when trying to transfer files onto these form's items. To specify behavior, use the File transfer methodproperty:

- As file. In this case, an object of the `File` kind (or an array of such objects) will be the transfer value. This kind of transfer is not supported when using a web client.
- As a reference to a file. In this case, as the Transfer value the object of the `FileRef` (or an array of such objects) kind will act. This way of transferring is supported for both thin and web clients.

If for the form's item the file transferring method is set to Like file reference, then:

- In the web client, the event processor `DragCheck` occurs once, when the mouse button is released and just before executing the event processor `Transferring`.

- In the thin client, the event processor `DragCheck` occurs multiple times during transfer.

If the file transferring method for the form's item is set to As file, then the web client ignors transferring and the event processors `DragCheck` and `Transfer` are not executed.

Let's see the object `FileRef` in more details. It is the special object enabling the consistent file operation in different environments and it contains some key data about the selected file. At this, you should understand that, for instance, the web client operation is prone to more limitations if compared with the thin client. Such limitations explain the following features of objects of `FileRef` type:

- These objects may not be serialized.
- As a result of the previous paragraph, these objects may not be stored in the form attributes, in the infobase data and be passed between the client and the server.
- The object `FileRef` can't be created from the embedded language. It can be created only by the platform during the drag mechanism operation.

At the same time, data structures located in the core memory (arrays, structures, fits etc) may be freely store objects of the `FileRef` type.

Moreover, consider the fact that the link to the file secures presence of the file on the hard drive only immediately after it is generated in the platform. At this, the object `FileRef` does not enable you to check whether the file that the object refers to exists and is located at the same place. Thus, objects of `FileRef`type are recommended to be used immediately after they are received. However, if you still need storing such objects before their actual use, you should do it using the client variables of the form where you operate the files.

7.9.7. Different approaches to modality

7.9.7.1. General information

In application solutions developed using the 1C:Enterprise system, it often becomes necessary to open a form that will provide several opportunities at the same time:

- Block work with other fragments of the application solution interface;
- Allow for entering any information (including an answer to the question);
- Stop execution of the application solution's code until work with the open form is completed.

At the same time, the forms in consideration can be created by an application developer in the 1C:Enterprise language and be part of the platform itself (for example, forms that open the `Warning()`, `Question()`, etc. methods).

Within the 1C:Enterprise system, there are two schemes for using such forms:

- Modal windows;

- Blocking windows.

Within a single application, either one approach or a mixed model can be used. To control this possibility, there is a special configuration property - Modal Usage Mode.

The first scheme of using modality (modal windows) involves a use of special kinds of the operating system's windows - modal windows. In case of using this scheme, the execution of code in the 1C:Enterprise language stops until the modal form is closed. A return value will be received directly at the place of opening the modal form and immediately after this (in the same method) any actions with this value can be performed: analyze, use in the algorithm, etc. Modal windows can be opened using the `OpenFormModal()` or `Warning()` methods. However, modal windows are not supported when running on iPads or in a web client.

In these cases, it is recommended to use the modal usage pattern with help of the blocking windows. The main difference between blocking and modal windows is that opening a blocking window does not stop execution of code in the 1C:Enterprise language. A result of the blocking window operation will be obtained by using a special callback procedure. A consequence of this feature is that the algorithm that uses modal calls to control its work will need to be completely reworked for use with the blocking windows.

From the point of view of an application developer, a difference between the two schemes of using modal windows will be shown in the following example. Suppose that there is some client processor that asks a question to a user and, depending on the answer, executes one or another branch of the application code.

Example with the modal windows:

```
&OnClient
Procedure ExecuteAlgorithm()
  Result = Question("Execute quickly or not slowly?",
QuestionDialogMode.YesNo);
  If Result = DialogReturnCode.Yes Then
    // First version of an algorithm
  Else
    // Second version of an algorithm
  EndIf;
EndProcedure
```

In this example (modal windows), a work of the 1C:Enterprise language will stop at the `Result = Question()` row and will continue only after a user answers the question.

Example with the blocking windows:

```
&OnClient
Procedure ExecuteAlgorithn(Command)
  Callback = New
NotifyDescription("ExecuteAlgorithmCallback",ThisObject);
  ShowQueryBox(Callback, "Execute quickly or not slow?",
QuestionDialogMode.YesNo);
EndProcedure

&OnClient
Procedure ExecuteCallbackAlgorithm(Result, AdditionalParameters)
Export
  If Result = DialogReturnCode.Yes Then
    // First version of an algorithm
  Else
    // Second version of an algorithm
  EndIf;
EndProcedure
```

In this example (using blocking windows), the algorithm is divided into two parts:

- Place where the question to user is shown;
- The place where the user's response is processed.

But these are different fragments of the applicable code. After calling the `ShowQueryBox()` method, execution of code in the 1C:Enterprise language will not end. However, a user will be able to answer the question asked only in the case when execution of code in the 1C:Enterprise language is completed and control will be transferred to the “1C:Enterprise” system. Consequently, in the place where the method that displays a form with question is called, it is impossible to get the result of a user working with the open form.

After a user answers the question, the export procedure will be called, a description of which an application developer must transfer to the method opening the blocking window. A description of the callback method is created using the `NotifyDescription` object. The callback method can be located in a form's module, a command module, and general module. If there is a need to specify that the callback method is in the same module as the blocking window call, a value of `ThisObject` should be used. It is also possible to transfer various auxiliary data to the callback processor, which in the processor itself will be available through the formal parameter `AdditionalParameters`. This may be the input data for the algorithm to operate, various parameters, etc.

In general, after calling any method of working with blocking windows, the method containing such a call should be completed immediately, and all further code should be located in the alert processor.

In case there is a need to organize a cascade callback of the callback method, the `ExecuteNotifyProcessing()` method should be used. Also, this method can be used in cases where the implementation of any action can occur either immediately or after asking a question about a need for action. For example,

when trying to re-read a file with data, the current state of the editor is checked; and if data in the editor is changed-, a question is asked about the possibility of re-reading the data.

Example:

```
&OnClient
Procedure RereadData(Command)

  Callback = New NotifyDescription("Re-readDataCompletion",
ThisObject, NameDataFile);
  If Modification, Then
    ShowQueryBox(Callback, "Data changed. Re-
read?"QuestionDialogMode.YesNo);
  Else
    ExecuteNotifyProcessing(Callback, DialogReturnCode.Yes);
  EndIf;

EndProcedure

&OnClient
Procedure Re-readDataCompletion(Result,FileName) Export

  If Result = DialogReturnCode.Yes Then
    // execute data reading
    Modification = False;
  EndIf;

EndProcedure
```

7.9.7.2. Closing an auxiliary or main application window and blocking windows

Closing a form can be performed in two work scenarios -when a single form is closed and when the entire application is closed (the main application window is being closed). In order to separate these scenarios, the system implements a separate process of processing the closure of the entire application. To indicate that the form is being closed during closure of the entire application, in the event processors `BeforeClose`, `OnClose`the special parameter `Exit` is implemented. In addition, in the event processor of the form `BeforeClose` and the event processor `BeforeFinishingWork` of the managed application module, the `MessageText` parameter is implemented. Let's consider these parameters in greater detal:

- `Exit` - This parameter is set by the system to `True` if a user tries to close the main application window (if this parameter exists for the selected processor). When closing one form, this parameter will be set to `False`.
- `MessageText` - This parameter must be formed in the event processor (if this parameter exists for the selected processor).

Consider in more detail the joint operation of these parameters during closure of an application. Code in the 1C:Enterprise language that processes closure of the form with specified features will be, as follows:

```
&OnClient
Var AnswerBeforeClosing;

&OnClient
Procedure BeforeClose(Rejection, Exit, MessageText,
StandardProcessing)
  If Not Exit Then
    If AnswerBeforeClosing <> True Then
      Rejection = True;
      Callback = New NotifyDescription("BeforeClosingCompletion",
ThisObject);
      ShowQueryBox(Callback,  "Close form?", QuestionDialogMode.YesNo);
    EndIf;
  Else
    Rejection = True;
    MessageText = "At closing form all data will be lost. You intend";
  EndIf;
EndProcedure

&OnClient
Procedure BeforeClosingCompletion(Result, AdditionalParameters) Export
  If Result = DialogReturnCode.Yes Then
    AnswerBeforeClosing = True;
    Close();
  Else
    AnswerBeforeClosing = Undefined;
  EndIf;
EndProcedure
```

In the example given, a special reaction is implemented on closing the entire application. Processing the process of closing an application works, as follows:

- If a form closes during application finishing its work, an application developer can detect this by the status of the `Exit` parameter and determines whether any message should be produced to a user that requires his response: complete work or still continue since there is substantial information contained in the form.
- If a developer believes that a message needs to be issued to a user, then he can do it using the `MessageText` parameter of `BeforeClose` or `BeforeExit` event processors. The `Rejection` parameter should also be set to `True`.
- If at least one form or procedure of the event processor `BeforeExit` reports that it “does not want” to close an application, the system will produse dialogues requiring a user to respond. A number of dialogues is equal to a number of rejections to close (including the processor of managed application module). A web client will display one dialog that will contain all the

messages that were installed in the managed forms and the managed application module. At the same time, in the web browser Mozilla Firefox the dialogue will contain only the text set by the web browser itself while the text set in the 1C:Enterprise event processors will not be displayed.

- In the thin client, the dialogue will display warning text set by a developer asked and the Complete work and Resumework buttons. If in all dialogues the Complete Work - id selected, the application's work will be terminated. If at least in one dialogue was selected Continue Work -, the application will not be completed and work will continue. In case of rejection from completing work, the form in which the Continue Work button was pressed will remain on the screen.

If the `Exit`parameter is set to `False`, this means that a single form is closed, and in this case the logic of operation isб as follows:

- First, a question is asked about the necessity (possibility) of closing a form and the actual closing of a form is canceled.
- After a user has answered the question, it is noted in the special client variable that the “real” closure of a form will now be performed and it will be closed again.

When processing closing of an application (parameter `Exit` is equal to the `True` value) in the processors `BeforeClose`, `OnClose`as well as `BeforeExit` and `OnExit`, a use of any server calls and opening any forms is prohibited. This leads to a recommendation to store in the form's data only information that may be needed to determine whether a form can be closed at closing the application's work or not. For existing forms, it may be necessary to reprocess the mechanisms responsible for closure.

Thus, displaying the dialogue with the Complete work and Continue work buttons is performed when at leaving the processors `BeforeClose`the processor's parameters are set, as follows: parameter `Exit` is set to the value `True` and the parameter `Rejection` is also set to `True` In this case, a dialogue will display the text that is set in the `MessageText` text. If the `MessageText` parameter is not set, then the string Operation in this window is not completedwill be shown in the dialogue.

Similar behavior will be observed if the `Rejection` parameter in the processor `BeforeExit` is set to `True`.

In all other cases, it is considered that there are no reasons preventing closure of forms and application.

7.9.7.3. Features of using the blocking windows

For the objects `StandardPeriodEditDialog`, `FormatStringWizard`, `FontChooseDialog`, `ColorChooseDialog`, `ScheduledJobDialog` in the callback processor that is being transmitted when calling the `Show`

`()` method, as the first parameter a value is transmitted of the main closable dialogue property if the OK button is pressed or `Undefined` otherwise. For example, when editing a standard period, the `StandardPeriod` (or `Undefined`) object will act as the first parameter of the callback processor.

Chapter 8. Working with queries

8.1. General information

To form and execute queries to database tables, the special `Query`object in the system is used. A query is convenient to use when there is a need to get a complex selection of data grouped and sorted, as needed. One of classic examples of its use is a summary of the register of record's state at a certain point in time. In addition, the query mechanism allows for easily receiving information in different time sections.

8.2. The sources of query data (table)

Sources of data for query language data are database tables. Tables are divided into two main classes: real and virtual.
Real tables are stored in a database, that is, they are interpreted from a real database table. In case of a real table, there may be present the calculable fields whose values are calculated as function of several real fields.
Virtual tables are not stored in database. When accessing information from virtual tables, the system automatically collects information from real database tables to execute a query. A virtual table can be parameterized, that is, the actual content of a virtual table can be determined by the values of parameters the actual values of which are specified in the query text. For each virtual table, a name is defined that is used in queries to identify a table.
The table's name can be set in English and Russian. For example, `Catalog.Goods`. Names of tables and fields cannot match keywords of the query language.
A separate subclass of tables do form the so-called object tables. An actual database table is always used as an object table. The semantic difference of object tables from others stems from the name - The object tables are intended for storing the state of objects of the 1C:Enterprise system, such as catalogs, documents, etc. Each object table corresponds to the kind of objects of the 1C:Enterprise system. For example, to objects of the `Catalog.Goods` kind does correspond a single table, objects of `Catalog.Counterparty` - another. Each individual record of the object table stores the state of an individual object of the corresponding kind. In accordance with this, each object table has a field of the

Reference kind for the current record. In addition, for object tables a method is defined for obtaining a user representation of an object from the contents of record fields.
The object tables can also be hierarchical tables. For hierarchical tables, a specially designated field Parent is defined containing a reference to the record, to which according to a hierarchy, submits the current record.
As a table field there can be either a table field or a nested table. The main difference between a regular field and a nested table is that within one record, a single field corresponds to a single value while a nested table corresponds to a value of the QueryResult kind with the pre-set array of columns. An example of a nested table may be the tabular part of a document or catalog.
As a kind of value fields of tables a value of the kind NULLcan act. Such values are contained in the table fields of records for which this field is not defined or does not make sense. For example, the values of this kind are contained in records belonging to the catalog's groups, in fields for which it is established that they can only have meaning at the items of this catalog.

8.3. Virtual and standard fields

A standard field -Is a field that is formed from data that is stored in a DBMS; at the same time, data can be further processed by means of a DBMS. A virtual field - Is a field that is formed from data that is stored in a DBMS, but the data is additionally processed by means of the 1C:Enterprise server (including in the file version). The following fields in query tables are virtual:

- Fields of representations obtained using as the Representation field of the corresponding table, as the query language function.
- DataVersion;
- PredefinedDataName;
- PointInTime;
- The ValueType of a table of the view plan of characteristics;
- Typeof the account plan table;
- Kindof the document log table;
- RecordType of the accounting register tables;
- RecordType of the accumulation register tables;
- any fields with values of the following kinds: ValueStorage, UUIDand a reference of the external data source.

Virtual fields possess the following features:

- Not all virtual fields can be converted “back” to the value stored in the DBMS. Such fields are the fields of representation and a field of the ValueTypefor a table of the view plan of characteristics.
- Not all virtual fields with the same name can be compared with each other if they are obtained from different tables. Such fields are DataVersion,

`PredefinedDataName`, `PointInTime`, `Kind`of the document log table.

- If a value of the system enumeration is obtained in the query field (references to the table of external data sources or a unique identifier) and the resulting value is not combined with other kinds, then when using the `Group By` operation in query, the values of this field are grouped regardless of which tables the value is obtained.
- For virtual fields of the string kind, a use of string operations and comparison with data of the `String` kind are not supported.

8.4. Query language

8.4.1. General information

As described above, to execute a query, one must compile the text of query. Query text -Is the instruction according to which a query should be executed. The query text describes which information database tables are used as sources of a query data, table fields that need to be processed in a query, rules of grouping, sorting results, etc.
An instruction is compiled in a special language (query language) and consists of separate parts - sections, sentences, words, functions and comments. The rest of this chapter discusses the purpose and methods of using the syntactic structure of a query language

8.4.2. Syntax diagram of a query language

This chapter describes a syntactic structure of the query language using a set of rules. Each rule describes one syntactic structure.
Each syntactic structure can contain keywords; delimiters (dots, commas, parentheses), in turn, other syntactic structures.

<Syntactic structure>

```
THIS_KEY_WORD
<This_Syntactic_ structure>, <This_Syntactic structure>
THIS_FUNCTION (<This_syntactic_ structure>)
```

In the rules describing a query language, the language constructs are indicated in angle brackets. Keywords and function names are described in capital letters.
Syntactic structures may contain optional items - keywords, etc. In the rules describing the query language, the optional items are enclosed in square brackets.

```
[ЭТО_НЕОБЯЗАТЕЛЬНОЕ_СЛОВО]
[<This_optional_structure>]
```

In some cases, one of several alternative items can be used in a syntactic structure. Such items in the rules are listed through vertical bars.

```
EITHER_THIS_WORD | EITHER_THIS_WORD
<Either_this_structure> | <Either_this_structure>
```

The descriptions of all structures are accompanied by examples explaining the procedure for their use in the query language.

8.4.3. Comments in query language

Text of query may include comments. A comment is a part of line that begins with a sequence of characters // and extends to the end of the line.

```
// This is a comment
```

During execution of query comments are dropped.

NOTE. A query designer removes comments from text.

8.4.4. Using predefined configuration data

A query text may contain predefined configuration data, such as:

- values of enumerations,
- predefined data:
  - of catalogs,
  - charts of characteristic types,
  - charts of accounts,
  - plans for calculations' kinds,
- empty references,
- values of the business process route points.

Also, a query text may contain values of system enumerations that can be attributed to fields in database tables: `AccumulationRecordType`,`AccountType` and `AccountingRecordType`.

Quering predefined configuration data and system enumeration values are accessed using a literal of a functional kind.

```
Value(<RepresentationValue>)
```

For system enumerations, the representation of a value is of the following kind:

```
<NameSystemEnumeration>.<Value>
```

Valid names of system enumerations are given above; a list of valid values for each of them can be reviewed in its description.

For predefined configuration data, the value representation looks like:

```
<KindPredefinedValue >.<NameObjectMetadata>.<Value>
```

A kind of a predefined value can be:

- `Catalog` (`Catalog`);
- `ChartOfCharacteristicTypes;`
- `ChartOfAccounts` (`ChartOfAccounts`);
- `ChartOfCalculationTypes` (`ChartOfCalculationTypes`);
- `Enumeration` (`Enum`).

A metadata object name is specified as the metadata object name, as defined in Designer.

For the enumerations defined in Designer, a value is specified as the name of the corresponding metadata object of the `EnumerationValue kind`. For all other kinds of the predefined values - As a name of the predefined data item, as specified in Designer, or `EmptyReference (EmptyRef)` to specify a null link.

For business process route points, the value representation is of the following kind:

```
BusinessProcess.<NameMetadataObject>.RoutePoint. <RoutePointName>
```

Below are a few fragments of queries that explain a use of predefined data in queries.

```
WHERE City = Value(Catalog.City.Moscow)

WHERE City = Value(Catalog.City.EmptyRef)

WHERE KindGoods = Value(Enumeration.KindGoods.Service)

WHERE RecordType = Value(AccumulationRecordType.Incoming)

WHERE RoutePoint = Value (BusinessProcess.Agreement.
RoutePoint.Consent)
```

8.4.5. Bilingual presentation of keywords

One of the essential features of query language of the 1C:Enterprise system is that, as in the 1C:Enterprise language, all keywords can be expressed in two ways: in Russian and English. The rest of this chapter is about the Russian way

of writing for keywords. The table below is about matches between Russian and English ways of writing for keywords in the query language.

Russian	English
AUTOORDER	AUTOORDER
BOOLEAN	BOOLEAN
IN	IN
OUTER	OUTER
INNER	INNER
ASC	ASC
ALL	ALL
CASE	CASE
SELECT	SELECT
CAST	CAST
WHERE	WHERE
YEAR	
GROUPING	GROUPING
DATE	DATE
DATETIME	DATETIME
TENDAYS	TENDAYS
DAY	
DAYOFYEAR	DAYOFYEAR
WEEKDAY	WEEKDAY
FOR	FOR
DATEADD	DATEADD
IS	IS
ISNULL	ISNULL
VALUE	VALUE
AND	AND
HIERARCHY	HIERARCHY
FROM	FROM
UPDATE	UPDATE
OR	OR
HAVING	HAVING
ELSE	ELSE
INDEX	INDEX
TRUE	TRUE
TOTALS	TOTALS
AS	AS
QUARTER	
WHEN	WHEN
COUNT	COUNT
END	END

Russian	English
ENDOFPERIOD	ENDOFPERIOD
LEFT	LEFT
ЛОЖЬ	FALSE
MAX	MAX
BETWEEN	BETWEEN
MONTH	Month
MIN	MIN
MINUTE	Minute
SETS	SET
BEGINOFPERIOD	BEGINOFPERIOD
NOT	NOT
WEEK	WEEK
UNDEFINED	UNDEFINED
OVERALL	OVERALL
UNION	UNION
TOP	TOP
PERIODS	PERIODS
TO	BY or ON
LIKE	LIKE
SUBSTRING	SUBSTRING
FULL	FULL
HALFYEAR	HALFYEAR
INTO	INTO
RIGHT	RIGHT
PRESENTATION	PRESENTATION
EMPTYTABLE	EMPTYTABLE
DISTINCT	DISTINCT
ALLOWED	ALLOWED
GROUP	GROUP
SECOND	SECOND
SECOND	SECOND
JOIN	JOIN
ESCAPE	ESCAPE
AVG	AVG
REFS	REFS
STRING	STRING
SUM	SUM
TYPE	TYPE
VALUETYPE	VALUETYPE
THEN	THEN
ONLY	ONLY
DESC	DESC

Russian	English
DROP	DROP
УПОРЯДОЧИТЬ	ORDER
HOUR	Hour
NUMBER	Number

NOTE. The keywords are not case-sensitive.

8.4.6. Main sections of the query text

A query text can be described by the following rule:

<Query Text>

```
<Query description>
[<Query merging>]
[<Ordering results>]
[АВТОУПОРЯДОЧИВАНИЕ]
[<Description of totals>]
```

As can be seen from this rule, the query text consists of several parts or sections:

- <Query description> - Is the only required section in the query text; in many cases it is enough to specify it only. The section defines sources of a query data, sample fields, groupings, etc. This section, in turn, is described by a whole set of rules and is discussed in detail below.
- <QueryMerging> - Query Language allows for combining the results of executing multiple queries.
- In the <OrderingResults> section the conditions for ordering strings as a result of a query can be defined.
- The AUTOORDERING section allows for enabling a mode of the automatic ordering of strings as a result of a query.
- In the <Description of Totals> section, there is a possibility to specify which totals should be calculated in query.

8.4.7. Description of a query

8.4.7.1. General information

As already mentioned, the query text must contain a section describing query, which defines:

- fields that will be contained in there as a result of query
- data sources for a query - original tables;
- conditions affecting the selection of data in a query;
- An order of grouping of the results of a query.

The section descripting query consists of several interrelated sentences:

<Query description>

```
ВЫБРАТЬ [РАЗРЕШЕННЫЕ] [РАЗЛИЧНЫЕ] [ПЕРВЫЕ <Количество>]
  <Selection fields list>
[FROM <Source list>]
[WHERE <Selection condition>]
[GROUP BY <Grouping fields>]
[HAVING <Selection condition>]
[FOR CHANGING [<Top level tables list>]]
```

A description of query begins with the obligatory SELECT (ВЫБРАТЬ) keyword.

The keyword ALLOWED (РАЗРЕШЕННЫЕ) means that a query will select only those entries to which a current user has rights. If this word is not specified, a query will operate with an error when it selects records for which a user does not have rights. This keyword can only be present in the top-level SELECTclause and applies to the entire query including subqueries. It should be kept in mind that the use of the keyword ALLOWED affects an operation only if the tables are subject to data access restrictions. Access rights to the table itself are not taken into account when using the keyword ALLOWED. For example, if a table does not have the READING (ЧТЕНИЕ) right, then a query using such a table will be executed with an error regardless whether the keyword ALLOWED was used or not in query.

Using the keyword DIFFERENT, there is a possibility to specify that duplicate strings should not fall into the result.

A construct FIRST <Quantity> allows for setting a limit on the number of strings as a result of query. The very first (according to the rules for ordering query results) strings will be selected. A quantity is specified as a whole number. In the query language, an ability to perform ordering in subqueries has been added in case the subquery contains the FIRST (ПЕРВЫЕ) construction.

The <List of sample fields> (<Список полей выборки>)section describes the fields that should be contained in there as a result of query.

In the clause FROM <List of sources> (<Список источников>), the data sources - the information database table are specified, the contents of which is processed in the request. A description of sources can be omitted only if they are fully defined in the list of selection fields.

The clause WHERE <Selection Condition> allows for filtering the result of a query. Only those records for which the specified condition turns out to be true fall into a result.

The clause GROUP allows for describing the order in which the results of query are grouped.

A HAVING (ИМЕЮЩИЕ) clause allows for imposing conditions on the results of grouping.

The clause FOR CHANGES (ДЛЯ ИЗМЕНЕНИЯ) is intended to indicate a need to block the data read in a transaction. The read data becomes inaccessible for reading in other sessions. For the file variant, the specified tables are blocked and for the client-server variant - only the selected records. A lock is released after transaction is completed.

All sample queries in this chapter contain both the text and result of a query. This implies that a query text is transferred as a parameter to the Execute method () of the Query object.

Let's provide an example of a rather simple query consisting of one SELECT statement and a list of selection fields.

Example:

```
// The report must show a list of invoices.

Select
Document.SalesInvoice.Reference
```

Result:

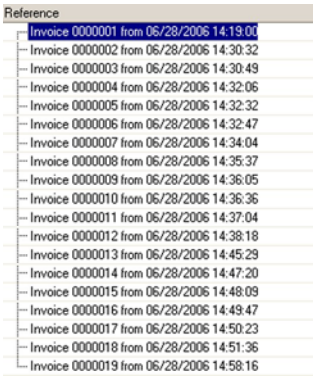


Fig. 151. Query result

8.4.7.2. Using the word "DIFFERENT"

In many situations, it is desirable that the same strings are not repeated in the report.

Example:

```
// There is a need to find out to which counterparts, in general
// shipped goods for the period.

Select
Document.SalesInvoice.Counterparty
```

Result:

Fig. 152. Query result

It can be seen that as a result of a query there are many duplicate strings which reduces its visibility. To avoid repetition, in the query's description the keyword `DIFFERENT (РАЗЛИЧНЫЕ)` should be specified.

Example:

```
Select Different
Document.SalesInvoice.Counterparty
```

Result:

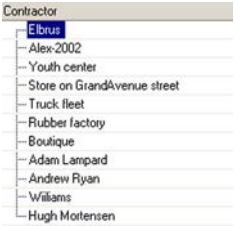


Fig. 153. Query result

8.4.7.3. Using word "FIRST"

In some cases it is necessary to output a limited number of strings to a report. For this purpose, the keyword FIRST and after it - the required number of strings should be indicated in description of a query.

Example:

```
// It is necessary to select five most expensive products.
// Sampling should be carried out based on price for goods in
descending order.

Select first five
Catalog.Products.Name,
Catalog.Products.WholePrice

Order By
Catalog.Products.WholePrice Desc
```

Result:

Description	PurchasePrice
Omega copy duplicator machine	3,500.00
Vega 700 telephone set	2,500.00
Vega 300 telephone set	2,200.00
Hewlett-Packard processor Vectra VL420	1,699.00
Hewlett-Packard processor Brio BA410	1,633.00

Fig. 154. Query result

8.4.8. Description of sampling fields

After the mandatory keyword SELECT (and the qualifying words DIFFERENT and FIRST), a list of sampling fields is specified in the query text. These fields will be processed when sampling data in a query. The result of a query will also have a set of fields defined in this list. Sampling fields are described according to the following rules:

<Selection fields list>

```
<Sampling field>[, <Sampling field>[, ...]] | *
```

<Sampling field>

```
<Описание поля> | <Логическое выражение> [[КАК] <Псевдоним поля>]
```

<Field description>

```
<Expression>[.<Group filds>]|<Description of blank table>
```

<Description of blank table>

```
BLANKTABLE. (<AliasList>)
```

<AliasList>

```
[<FieldAlias>][, <Alias list>]
```

A list of sampling fields consists of one or more items separated by commas. Each sampling field consists of a description of a sampling field and an optional field alias.
Instead of listing fields in the sampling list, there is a possibility to specify an asterisk (*). This will mean that the query result must contain all the fields that are in the source tables - the original query data described in the source list.

NOTE 1. When specifying an asterisk (*) in the list of sampling fields, the result does not include the virtual fields of the original tables.
NOTE 2. Receiving samples of very large size (more than **64 MB**) requires a sufficient amount of free disk space used to accommodate temporary server and client files.

A field description determines how the field values should be formed. In the simplest case, the sampling field is a reference to a field of the original table. A reference can be specified indicating the table containing this field or without specifying the table itself.
In general, the sampling field can be not only a reference to a field of the source table but a certain expression or a logical expression.
Query results can be grouped using aggregate functions, specified as expressions in the sampling fields.
An alias can be assigned to each sampling field. In future, the alias field can be used to more conveniently refer to this field.
A group of fields can be specified only when the sampling field points to a nested table. In this case, there is a possibility to specify which fields should be processed in a sample by nested table. If a group of fields is not specified, all fields of the nested table will be processed in the selection.

8.4.8.1. Field alias in a selection list

If an alias is assigned to the sampling field then in future it will be possible to refer to this field using its alias, in the `ORDER BY` and `RESULTS (ИТОГИ)`

sentences, as well as when working with the result of a query. Such treatment may be more convenient and visible, and in some cases the only possible one.

The keyword AS can precede the field alias. This word can be not specified at all but if it is specified, the visibility and readability of the query text is increased. Field aliases are defined according to the rules for assigning variable IDs. Alias in queries cannot coincide.

Assigning aliases to fields does not in itself affect the sampling of data in a query.

Example:

```
// There is a need to select from the catalog of goods
// names of goods and groups.

Select
Catalog.Products.Parent.Name As Goods,
Catalog.Products.Parent.Name As Group
From
Catalog.Products
```

Result:

Product	Group
SW installation	Services
OS Windows setup consulting	Services
1C setup consulting	Services
Notebooks	
Telephone set	
Copy duplicator machine	
Omega copy duplicator machine	Copy duplicator machine
Notebook Rover Computers Navigator KT7	Notebooks
Notebook Rover Computers Explorer	Notebooks
Vega 700 telephone set	Telephone sets
Vega 300 telephone set	Telephone sets
Siemens SL45 telephone set	Telephone sets
LG W7200 telephone set	Telephone sets

Fig. 155. Query result (fragment)

It should be noted that the fields as a result of a query are called the Goods (Товар) and Group (Группа). If the fields' aliases were not specified, the fields as a result of a query would be called Name (Наименование) and Name1 (the names of fields as a result of a query cannot be the same, therefore "1" is automatically added to the name of the second field), which is less obvious.

8.4.8.2. Nested tables in the list of selecting fields

A field in the sampling list can refer to a nested data source table of a query. In this case, the query result field will be of the QueryResult kind, that is, it will contain a nested query result formed based on the nested table -source.

By default, all fields of a nested data - source table are included in the nested result. It is possible to explicitly define a group of fields that should be contained in the nested query result. A group of fields of the nested result is described according to the following rule:

<A group of fields>

```
( <A list of nested fields> ) | *
```

<A list of nested fields>

```
<Nested field>[,      <Nested field>[, ...]]
```

<Nested field>

```
<Expression> [[КАК] (AS) <Field alias>]
```

A list of nested fields consists of one or more items separated by commas. If the list consists of one item, it is not necessary to enclose it in brackets.
Instead of listing nested fields, there is a possibility to specify an asterisk (*); this will mean that the query result must contain all the fields that are in the nested table.
A nested field can represent some expression. In the simplest case, the expression -is a reference to a field of the nested table.
Each nested field can be assigned an alias. In future, the field's alias can be used to more conveniently refer to this field, similarly to the fields' alias in the sampling list.
Aliases can be assigned to the nested fields, regardless of whether the alias is set to the nested table itself.

Example:

```
// In a report, it is necessary to display the specification of sales
invoices
 // a document itself, nomenclature and quantity.

Select
Document.SalesInvoice.Reference,
Document.SalesInvoice.Composition.(Products AS Goods, Quantity)
(Документ.РасходнаяНакладная.Состав.(Номенклатура Как Товар,
Количество))
```

Result:

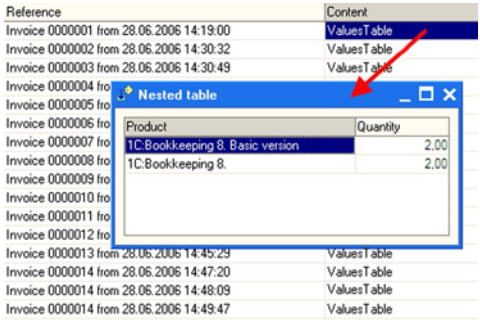


Fig. 156. Query result (fragment)

It should be noted that the field Composition (Состав) of the query result is a nested table that has the fields Products (Номенклатура) and Quantity (Количество).

8.4.8.3. Blank nested tables in the selection list

If a query uses a join and some parts of joining contain nested tables and some do not -, it becomes necessary to fill the selection list by fields - the empty nested tables. This is done using the keyword BLANKTABLE (ПУСТАЯТАБЛИЦА). After it, the aliases of the fields which the nested table will be composed of, are indicated in parentheses.

Example:

```
SELECT Reference.Number, BLANKTABLE.(Num,Goods,Quant) AS Composition
(ВЫБРАТЬ Ссылка.Номер, ПУСТАЯТАБЛИЦА.(Ном, Тов, Кол) КАК Состав))
ИЗ Document.SalesInvoice (ИЗ Документ.РасходнаяНакладная)
UNITE ALL
SELECT Reference.Number, Composition.(LineNumber, Product, Quantity)
ИЗ Document.SalesInvoice (ИЗ Документ.РасходнаяНакладная)
```

8.4.9. Descripting sources of query

A task of the clause FROM (ИЗ) is to designate a list of the source tables - data sources used in this SELECT statement.

It should be noted that the FROMclause is optional in the query language. It may be omitted if the data sources are clearly identified in the description of the list of sampling fields contained in the SELECT clause. It should be noted that a number of examples in the previous sections did not contain the clause FROM.

After the keyword FROM the list of sources is indicated. In general, the list of sources is described by the following set of rules:

<A list of sources>

```
<Source>[, <Source>[, ...]] (<Источник>[, <Источник>[, …]])
```

<Source>

```
<Source description> [<List of connections>] (<Описание источника> [
<Перечень соединений> ])
```

<Source description>

```
<Table> [[КАК](AS)] <source alias>] (<Таблица>  <Псевдоним
источника>])
```

<Table>

```
<Table name> [(<Parameters>)] | (<Query description>) (<Имя
таблицы>[(<Параметры>)] | (<Описание запроса>))
```

Query data sources are listed in the source list and are separated by commas. Each source in the list of sources must include a description of source; in addition, a list of connections - the rules for connecting sources with other sources may be indicated. For the description of connection features, se p. 665

If the infobase table serves as a data source, the source description contains a name of the table.

If the source table is virtual, the parameters for its formation can be specified. Detailed parameters of virtual tables are described in the section 1C:Enterprise Language - Working with queries - Query tables of the embedded reference.

A subquery can represent a data source for quering. In this case, the source description contains a description of query.

In a description of the data source, its alias can also be assigned. In future, the source alias can be used to more conveniently refer to this source.

8.4.9.1. Specifications of connections

8.4.9.1.1. General information

When defining several sources, for each record in the list from the first source table, a sampling is made from the second source table, etc. Thus, as a result of quering, all possible combinations of all records from all specified sources are formed.

Example:

```
SELECT
Counterparties.Reference as Counterparty,
PriceKinds.Reference AS PriceKind
FROM
Catalog.Counterparties AS Counterparties,
Catalog.PriceKinds AS PriceKinds
```

Result:

Contractor	PriceType
Temp_plus	Wholesale
Temp_plus	Purchase
Temp_plus	Retail
Temp_plus	Individual wholesale
Vendors	Wholesale
Vendors	Purchase
Vendors	Retail
Vendors	Individual wholesale
Maximus	Wholesale
Maximus	Purchase
Maximus	Retail
Maximus	Individual wholesale
Novation	Wholesale
Novation	Purchase
Novation	Retail

Fig. 157. Query result (fragment)

The result of query contains combinations of all counterparties with all types of prices. As a rule, such a result in itself does not make sense. Usually, combinations of records from different source tables need to be limited by any conditions. In the query language, it is possible to describe such a connection of sources, specifying the sources themselves and determining the conditions under which combinations of records from these sources need to be included in the query result.

Connections are of several kinds; they are described by the following rules:

<List of connections>

```
<Join> [<List of joins>] (<Соединение> [<Перечень соединений>])
```

<Join>

```
[ВНУТРЕННЕЕ] СОЕДИНЕНИЕ <Описание источника> ПО <Условие отбора> |
ЛЕВОЕ [ВНЕШНЕЕ] СОЕДИНЕНИЕ <Описание источника> ПО <Условие отбора> |
ПРАВОЕ [ВНЕШНЕЕ] СОЕДИНЕНИЕ <Описание источника> ПО <Условие отбора> |
ПОЛНОЕ [ВНЕШНЕЕ] СОЕДИНЕНИЕ <Описание источника> ПО <Условие отбора>
```

In an overall case, the list of connections contain and describe not only a single connection (of two sources) but also several connections of several sources at once.

Description of source (Описание источника) contains a description of the source table.

A sampling condition contains the conditions according to which in the sample it is necessary to combine data from the original tables - source of a query.

Keywords LEFT (ЛЕВОЕ), RIGHT (ПРАВОЕ) and FULL (ПОЛНОЕ) shape the character of connection. Words INNER (ВНУТРЕННЕЕ) or OUTER (ВНЕШНЕЕ) can be omitted altogether, they increase a visibility and readability of the query text.
The connected sources are not equivalent to each other, and in some cases the result depends on which table is listed first, until the keyword CONNECTION (СОЕДИНЕНИЕ) (to the left of it), and which - second (to the right).
To describe the connections we will use two simple tables. One table, called Companies, consists of two fields: The Name and Phone contain the following data:

Description	Phone
ConfetProm	
ServiceLog	2-12-85-03

Fig. 158. The table "Companies"

Another table called Contacts (Контакты) consists of three fields: Name (Наименование), Phone (Телефон) and Company (Компания) (reference to the table item of a Company). A table contains the following data:

Description	Phone	Company
John JonesT	222-3-22	
William Smith		ServiceLog
Peter Jackson	222-3-22	ConfetProm

Fig. 159. The table "Contacts"

NOTE. In terms of the 1C:Enterprise both tables are catalogs.

8.4.9.1.2. inner join

Inner join means that, from both original tables -source data only those combinations of records that meet the specified condition should be included in the query result. The rest of records do not make to the result.

Example:

```
SELECT
  Companies.Name AS Company,
  Contacts.Name AS Contact
FROM
  Catalog.Contacts AS Contacts
    INNER JOIN Catalog.Companies AS Companies
    ПО (software) Contacts.Company = Companies.Reference (ПО
Контакты.Компания = Компании.Ссылка)
```

Result:

Company	Contact
ConfetProm	Peter Jackson
ServiceLog	William Smith

Fig. 160. Query result

8.4.9.1.3. Left outer join

The left outer join means that the result of query must include combinations of records from both original tables meeting the specified condition. But in contrast to the internal connection the result of query must also include records from the first ((indicated to the left of the word `CONNECTION (СОЕДИНЕНИЕ)`)) source for which no matching records from the second source were found.
Thus, the result of query will include all records from the first source; they will be connected to records from the second source when the specified condition is met. The query result strings, for which no matching records from the second source are found, will contain `NULL` in the fields generated based on the records from this source.

Example:

```
SELECT
  Contacts.Name AS Contact (Контакты.Наименование КАК Контакт),
  Companies.Name AS company (Компании.Наименование КАК Компания)
FROM
  Catalog.Contacts AS Contacts
    LEFT JOIN Catalog.Companies AS Companies
    ПО (software) Contacts.Company = Companies.Reference (ПО
Контакты.Компания = Компании.Ссылка)
```

Result:

Contact	Company
Peter Jackson	ConfetProm
John Jones	NULL
William Smith	ServiceLog

Fig. 161. Query result

8.4.9.1.4. Right outer join

The right outer join means that the result of query must include combinations of records from both original tables that meet the specified condition. In addition, the result of query should also include records from the second (indicated to the right of the word CONNECTION) source for which no matching records from the first source were found.
Thus, the result of query will include all records from the second source; they will be connected to records from the first source when the specified condition is met. The query result lines, for which no matching conditional records from the first source are found, will contain NULL in the fields generated based on the records from this source.

NOTE. In fact, the right outer connection can be expressed through the left outer connection. This is exactly what the Designer's query does.

Example:

```
SELECT
  Contacts.Name AS Contact (Контакты.Наименование КАК Контакт),
  Companies.Name AS company (Компании.Наименование КАК Компания)
FROM
  Catalog.Contacts AS Contacts
    RIGHT JOIN Catalog.Companies AS Companies
    ПО (software) Contacts.Phone= Companies.Phone (ПО Контакты.Телефон
= Компании.Телефон)
```

Result:

Contact	Company
Peter Jackson	ConfetProm
John Jones	NULL
William Smith	ServiceLog

Fig. 162. Query result

8.4.9.1.5. Full outer join

A full outer join means that the result of query must include combinations of records from both original tables that match the specified condition. In addition, the query result must also include those records from both sources for which no matches were found.
Thus, the result of query will include all records from both sources; they will be connected to each other when the specified condition is met. The strings of query result for which no matching records from any source were found will contain NULL in the fields formed on the basis of records from this source.

Example:

```
SELECT
  Contacts.Name AS Contact (Контакты.Наименование КАК Контакт),
  Companies.Name AS company (Компании.Наименование КАК Компания)
FROM
  Catalog.Companies AS companies (Справочник.Компании КАК Компании)
    A FULL JOIN Catalog.Contacts AS contacts (ПОЛНОЕ СОЕДИНЕНИЕ
Справочник.Контакты КАК Контакты)
    PO (software) (Contacts.Phone = Companies.Phone) (ПО
(Контакты.Телефон = Компании.Телефон))
```

Result:

Contact	Company
NULL	ServiceLog
William Smith	ConfetProm

Fig. 163. Query result

8.4.9.2. Alias of the data sources

If the data source is assigned an alias, then in future it will be possible to access this source using this alias (and it will no longer be possible to access it by specifying the name of the table). Such treatment may be more convenient and visible, and in some cases the only possible one.

The alias is set in accordance with the rules for assigning variable IDs. Alias in queries cannot coincide.

A keyword `HOW (КАК)` may precede the source alias. This word can be not specified at all but if it is specified, the visibility and readability of the query text is increased.

The assignment of aliases to sources does not in itself have any effect on the selection of data in query.

Example:

```
SELECT
  Goods.Name,
  Goods.Parent
FROM
  Catalog.Products AS Goods
```

This example demonstrates a use in the sampling fields list of the `Goods` alias assigned to the source table `Reference.Products`.

8.4.9.3. Nested tables in the list of sources

The list of sources may include nested tables - tabular parts of catalogs and documents.

Example:

```
// The reports needs to contain the specification of invoices
// show the document itself, the products and quantity.
// In the list of sources there is a nested table "Composition"
(Состав)
// a table part of the invoice.
// Sampling is limited to eight entries in order not to overload the
example.

SELECT FIRST EIGHT
  SalesInvoiceComposition.Reference,
  SalesInvoiceComposition.Products,
  SalesInvoiceComposition.Quantity
FROM
  Document.SalesInvoice.Contents AS SalesInvoiceContents
```

Result:

Reference	Nomenclature	Quantity
Invoice 0000001 from 28.06.2006 14:19:00	1C:Bookkeeping 8. Basic version	2.00
Invoice 0000001 from 28.06.2006 14:19:00	1C:Bookkeeping 8.	2.00
Invoice 0000002 from 28.06.2006 14:30:32	Monitor 15' LG Studioworks 575N	5.00
Invoice 0000002 from 28.06.2006 14:30:32	Mouse 2-button A4 Tech PS/2	10.00
Invoice 0000002 from 28.06.2006 14:30:32	Mouse LOGITECH M-S48 PS/2	10.00
Invoice 0000002 from 28.06.2006 14:30:32	Mouse A4 Tech OK-720 PS/2	2.00
Invoice 0000003 from 28.06.2006 14:30:49	1C:Payroll and Human Resources 8	1.00
Invoice 0000004 from 28.06.2006 14:32:06	Windows XP Home Edition Russian CD	1.00

Fig. 164. Query result

It should be noted that when specifying a nested table in the list of sources, it is possible to refer both to the fields of the nested table itself and to the fields of the top level table (the one that contains a nested table) through the field `Reference`. In this case, the field `Reference.FieldName` of the document itself is called.

8.4.9.4. Nested queries in the list of sources

In the source list of a query, the subquery can be used as the source table. In this case, the source description contains a description of the subquery. A description of the subquery is made in the same way as a normal one.

Using a subquery as a data source is no different from using the infobase table. As fields of such a source, all the fields described in the list of the sampling fields of a nested query are available.

Example:

```
SELECT
  NestedQuery.Reference,
  NestedQuery.Products,
  NestedQuery.Quantity
FROM
  (SELECT FIRST EIGHT
    SalesInvoiceComposition.Reference AS reference
(РасходнаяНакладнаяСостав.Ссылка КАК Ссылка),
    SalesInvoiceContents.Products AS Products,
    SalesInvoiceContents.Quantity AS Quantity
  FROM
Document.SalesInvoice.Composition AS SalesInvoiceComposition) AS
Subquery ((Документ.РасходнаяНакладная.Состав КАК
РасходнаяНакладнаяСостав) КАК ВложенныйЗапрос))
```

The result will be exactly the same as in the previous example.

8.4.10. Filtration query results

The clause `WHERE <selection condition>` allows for specifying a condition for selecting data from the source tables - source of a query; it will process only those records for which this condition is true.

Example:

```
// It is necessary to find out which counterparties are private
individuals.

SELECT
  Counterparties.Name
FROM
  Catalog.Counterparties AS Counterparties
WHERE Counterparties.Kind = Value (Enumeration. CounterpartiesKinds.
PrivatePerson)
```

Result:

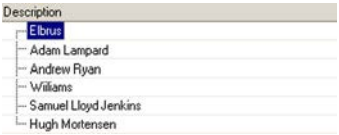


Fig. 165. Query result

NOTE. It is not necessary at all that the field appearing in the `WHERE (ГДЕ)` clause be included in the sampling list.

The selection filter can be defined both as a simple logical expression and as more complex, in which simple logical expressions are interconnected by `AND (И)`, `Or (ИЛИ)` and `No (Не)` logical operators.

When selecting data from the table that has table parts, the conditions on records of the table part affect both composition of the records selected from table and composition of the records of the table parts.
In general: the selection of records containing tabular sections is performed as follows:

- Tabular sections are linked to each table in the source list by the left-hand external connection.
- All the connections defined in the source list are made.
- Records are selected that meet the condition of the `WHERE` section.
- The records are grouped by the fields of the tables from the list of sources, including reference fields, by placing the records of the tabular sections as the values of the corresponding fields of the table records from the selection list.

8.4.11. Query results grouping

8.4.11.1. Simple grouping

The source data in the query can be grouped using aggregate functions specified as fields in the selection list. This means that the rows as a result of the query will contain the results of the calculation of the specified aggregate functions, calculated (grouped) by the records of the source tables.
The aggregate functions themselves are specified in the list of selection fields. It is necessary to specify a list of fields which it is required to group by in `GROUP BY <Grouping fields>` expression. The query will group the source table records containing the same values of the specified fields.
The list of grouping fields contains references to the fields of the source tables - which are request sources, separated by commas:

<Grouping fields>

<Field dereferencing> [, <Field dereferencing> [, …]]

IMPORTANT! When grouping the results of a query, aggregate functions must be specified in the list of selection fields, and in addition to those, only the fields which the grouping is performed by are allowed to be specified. In queries containing grouping by any field, this field must not be used inside the aggregate functions. Using the field simultaneously for grouping and within the aggregate function - must be considered an error. Such queries can be interpreted differently, depending on which DBMS is used. If this rule is not complied with, errors may occur that cause program termination.

The exception is when the aggregate functions are applied to the fields of the nested table. In this case, is it possible to refer to the fields of the top-level table in the list of selection fields without grouping the results by these fields.

When using aggregate functions, GROUP BY statement may not be indicated at all; all query results will be grouped into one single line.

Example:

```
// Getting statistics on the sale of goods is required:
// maximum, minimum and average prices in sales invoices.

Select
  Invoice.Products,
  Avg (Invoice.Price) As Avg,
  Max(Invoice.Price) As Max,
  Min(Invoice.Price) As Min

From
  Document.SalesInvoice.Content As Invoice

Group By
  Invoice.Products
```

Result:

Nomenclature	Average	Maximum	Minimum
Monitor 15' LG Studioworks 575N	151,666667	155,00	145,00
Monitor 17' Philips 107S20	210	210,00	210,00
Monitor 19' Hitachi CM715ET	350	350,00	350,00
Monitor LCD 22' M8537ZM/A	2 000	2 000,00	2 000,00
Hewlett-Packard base unit Brio BA410	2 000	2 000,00	2 000,00
Hewlett-Packard base unit Vectra VL420	2 100	2 100,00	2 100,00
IBM NetVista A22p base unit	1 816,666667	2 500,00	1 350,00
IBM NetVista M41 base unit	1 450	1 450,00	1 450,00
Laser printer Canon LBP-810	300	300,00	300,00

Fig. 166. Query result (fragment)

8.4.11.2. Grouping data sets

Simple grouping (GROUP BY) allows it to calculate pre-defined aggregate functions per each data set which contains a unique combination of values for fields listed in GROUP BY sentence. However, a need may arise to get several aggregated results from a single data set with different grouping field sets.

For instance, let's take ProductsAccounting accumulation register. This register has Products and Warehouse dimensions and Amount resource. You need to get quantitative stock for the following groupings:

1. By warehouse.
2. By product.
3. By product and warehouse.

To address the first task, you can use the following query:

```
SELECT
  ProductAccountingStock.Warehouse AS Warehouse,
  SUM (ProductAccountingStock.AmountStock) AS AmountStock
FROM
  AccumulationRegister.ProductAccounting.Stock AS
ProductAccountingStock

GROUP BY
  ProductAccountingStock.Warehouse
```

To address the second task, you can use the following query:

```
SELECT
  ProductAccountingStock.Products AS Products,
  SUM (ProductAccountingStock.AmountStock) AS AmountStock
FROM
  AccumulationRegister.ProductAccounting.Stock AS
ProductAccountingStock

GROUP BY
  ProductAccountingStock.Products
```

The last task can be addressed as follows:

```
SELECT
  ProductAccountingStock.Products AS Products,
  ProductAccountingStock.Warehouse AS Warehouse,
  SUM (ProductAccountingStock.AmountStock) AS AmountStock
FROM
  AccumulationRegister.ProductAccounting.Stock AS
ProductAccountingStock

GROUP BY
  ProductAccountingStock.Products, ProductAccountingStock.Warehouse
```

All three tasks listed hereinabove can be addressed either by executing the said three queries in a row and further processing the three results in 1C:Enterprise language or by executing the following merged query:

```
SELECT
  NULL AS Products,
  ProductAccountingStock.Warehouse AS Warehouse,
  SUM (ProductAccountingStock.AmountStock) AS AmountStock
FROM
  AccumulationRegister.ProductAccounting.Stock AS
ProductAccountingStock

GROUP BY
  ProductAccountingStock.Warehouse
MERGE
SELECT
  ProductAccountingStock.Products,
  NULL,
  SUM(ProductAccountingStock.AmountStock)
FROM
  AccumulationRegister.ProductAccounting.Stock AS
ProductAccountingStock

GROUP BY
  ProductAccountingStock.Products
MERGE
SELECT
  ProductAccountingStock.Products,
  ProductAccountingStock.Warehouse,
  SUM(ProductAccountingStock.AmountStock)
FROM
  AccumulationRegister.ProductAccounting.Stock AS
ProductAccountingStock

GROUP BY
  ProductAccountingStock.Products,
  ProductAccountingStock.Warehouse
```

Both methods described hereinabove are applicable. However, they have the following significant drawback: each of them executes a very similar query three times in a row. So far as significant data volumes are concerned, it may be time-consuming.

To reduce time required to handle the said task, use `GROUPING SETS` extension in `GROUP BY`. `GROUP BY GROUPING SETS` allows it to calculate all groupings you need in a single search of query tables.

The foregoing sentence uses the following syntax:

GROUP BY GROUPING SETS ((SetGroupFields1)[, (SetGroupFields2)[,...]])

SetGroupFieldsN

<Field dereferencing> [, <Field dereferencing> [, …]]

Technically, each `SetGroupFields` is a grouping set. Aggregate functions available in a query will be similar for all grouping sets. If a query field is not involved in aggregation (i.e. not an aggregate function parameter), make sure that it is included at least in one grouping set. Grouping sets are supported for upper level tables only (including virtual tables).

Therefore, a merged query (see above) can be written as follows:

```
SELECT
  ProductAccountingStock.Products AS Products,
  ProductAccountingStock.Warehouse AS Warehouse,
  SUM (ProductAccountingStock.AmountStock) AS AmountStock
FROM
  AccumulationRegister.ProductAccounting.Stock AS
ProductAccountingStock

GROUP BY GROUPING SETS
(
(ProductAccountingStock.Warehouse),
(ProductAccountingStock.Products),
(ProductAccountingStock.Products, ProductAccountingStock.Warehouse)
)
```

The query result will include the results of all three queries (or one combined query). Thus, it is not referred to other ways of obtaining results. First of all, it is about **increasing the speed** of obtaining the result and improving productivity. Note that parentheses available in grouping set description are of material importance. Thus, the following expression (let it be B1):

```
GROUP BY GROUPING SETS
(
(ProductAccountingStock.Products, ProductAccountingStock.Warehouse)
)
```

is dissimilar to the following (let it be B2):

```
GROUP BY GROUPING SETS
(
ProductAccountingStock.Products, ProductAccountingStock.Warehouse
)
```

B1 is equivalent to the following expression:

```
GROUP BY ProductAccountingStock.Products,
ProductAccountingStock.Warehouse
```

B2 is equivalent to the following expression:

```
GROUP BY GROUPING SETS
(
(ProductAccountingStock.Products),
(ProductAccountingStock.Warehouse)
)
```

In other words, B2 describes two grouping sets, and each of them contains grouping per a single field (see below): `ProductAccountingStock.Products` and `ProductAccountingStock.Warehouse`.

8.4.12. Conditions for aggregate function values

`HAVING <Filter condition>` expression allows to set conditions for the aggregate function values. In other query language structures, for example, in the `WHERE` expression, it is impossible to specify aggregate functions as conditions.

Example:

```
// It is necessary to select products that sold more than 20 pieces.

SELECT
  Invoice.Products,
  SUM(Invoice.Quantity) AS Quantity
FROM
  Document.SalesInvoice.Content As Invoice

Group By
  Invoice.Products

HAVING
  SUM(Invoice.Quantity) > 20
```

Result:

Nomenclature	Quantity
Laser printer HP LaserJet 2200	26
Mouse 2-button A4Tech PS/2	35
Mouse Ice Mouse MUS-2	30
Mouse LOGITECH M-S48 PS/2	41
Mouse GENIUS "EASY" (3 buttons).	21
Keyboard Apple Pro Keyboards	23
Keyboard LK-601 KB-2000 PS/2	23

Fig. 167. Query result

IMPORTANT! Only aggregate functions and fields, which are grouped by, can be used in filter condition.

8.4.13. Query merging

It is possible to merge several queries in the query language. In this case, the records obtained using each of the merged queries will be combined into one result of the query.

When merging, each query collects data independently, and such operations as ordering the results and calculating totals are performed on the query merging result already.

The fields of the query result will be named as described in the list of fields for selecting the first of the merged queries. The fields of the remaining queries selection are matched with the result fields in accordance with the order in which

they appear in the list of selection fields. The merged queries must have the same number of fields in the list of selection fields.
If the selection fields of the merged queries have different types, the query result fields will have a composite type.
Query merging is described by the following rule:

<Query merging>

```
MERGE [BCE]
<Query description>
[<Query merging>]
```

The merging of queries begins with the mandatory `MERGE` keyword, followed by a description of the joined query. Then one more query can be added, etc.
By default, when merging queries, completely identical strings in the query result, generated by different queries, are replaced with one string. If it is necessary to leave different lines, the `ALL` keyword must be specified.

Example:

```
SELECT
  Receipt.Products AS Goods,
  SUM(Receipt.Quantity) AS Receipt,
  SUM(0) AS Expense
FROM
  Document.Invoice.Content AS Receipt

Group By
  Receipt.Products

MERGE

SELECT
  Expense.Products,
  SUM(0),
  SUM(Expense.Quantity)
FROM
  Document.SalesInvoice.Content AS Expense

Group By
  Expense.Products
```

Result:

Product	Receipt	Expense
Keyboard LK-601 KB-2000 PS/2	160	
Windows XP Home Edition Russian CD		3
Windows XP Home Edition Russian CD	25	
Windows XP Home Edition Russian UPG CD		3
Windows XP Home Edition Russian UPG CD	42	
Windows XP Professional Russian CD		6
Windows XP Professional Russian CD	15	
1C:Bookkeeping 8. Basic version		2
1C:Bookkeeping 8. Basic version	16	
1C:Bookkeeping 8.		2

Fig. 168. Query result (fragment)

8.4.14. Query results ordering

8.4.14.1. General information

`ORDER BY` expression allows to sort strings in the query results.

<Ordering results>

```
ORDER BY <Ordering conditions>
```

<Ordering conditions>

```
<Ordering field> [<Order>]
[, <Ordering field> [<Order>][, …]]
```

<Ordering field>

```
<Expression> <ORDER> ASCENDING | DESCENDING | HIERARCHY | DESCENDING
HIERARCHY
```

A comma separated list of conditions according to which it is necessary to sort the result of the query is specified in `ORDER BY` expression. The selections are ordered first by the first condition, then by the second one, and so on.

In general case, the ordering condition may be some kind of expression. The query result rows will be ordered by the values of this expression calculated for each row.

The ordering can be carried out in ascending or descending order of values, and for tables for which the hierarchy property is set, - based on the hierarchy also. The order can be set for each field independently.

The field that appears in the ordering conditions does not have to be included in the query result.

Example:

```
// It is necessary to select five most expensive products,
// having arranged them in descending order of price.

SELECT FIRST FIVE
  Products.Name,
  Products.PurchasePrice
FROM
  Catalog.Products AS Products

ORDER BY
  Products.PurchasePrice DESCENDING
```

Result:

Description	PurchasePrice
Omega copy duplicator machine	3,500.00
Vega 700 telephone set	2,500.00
Vega 300 telephone set	2,200.00
Hewlett-Packard base unit Vectra VL420	1,699.00
Hewlett-Packard base unit Brio BA410	1,633.00

Fig. 169. Query result

8.4.14.2. Ordering by hierarchy

For catalogs it is possible to assign the ordering by the catalog hierarchy.

Example:

```
SELECT
  Catalog.Products.Name,
  Catalog.Products.FullDescr
ORDER BY
  Catalog.Products.Name Hierarchy
```

Result:

Description	FullDescription
⊞ Keyboards	
⊟ Copy duplicator machine	
└ Omega copy duplicator machine	Omega copy duplicator machine
⊟ Monitors	
├ 15' LG Studioworks 575N monitor	15' LG Studioworks 575N monitor <0.28, 50-160Hz, 128...
├ 17' Philips 107S20 monitor	17' Philips 107S20 monitor (0.27), Flat SM, HC, AGRAS, ...
├ 19' Hitachi CM715ET monitor	19' Hitachi CM715ET monitor FST 0.21, 1600x1200@7...
└ LCD 22' M8537ZM/A monitor	LCD 22' M8537ZM/A monitor Apple Cinema HD Display ...
⊞ Mouses	

Fig. 170. Query result (fragment)

Hierarchical sorting is only possible by field, but not by some operation on it: the ordering field must contain a reference to the field of the source table - which is the query data source.

IMPORTANT! It makes sense to set the ordering by the hierarchy if it is the catalog table that is defined as the source, and not some other table containing only a reference to the catalog.

Example:

```
SELECT
  SalesInvoice.Counterparty.Name,
  SalesInvoice.Number,
  SalesInvoice.Warehouse
FROM
  Document.SalesInvoice AS SalesInvoice
```

In this example, hierarchical ordering will not work, because there is no connection with the catalog, and the groups from this catalog will not be included in the query result.
To order by hierarchy it is necessary to arrange a connection with the catalog.

Example:

```
SELECT
  Counterparties.Name AS Name,
  SalesInvoice.Number,
  SalesInvoice.Warehouse
FROM
  Catalog.Counterparties AS Counterparties
LEFT CONNECTION Document.SalesInvoice AS SalesInvoice
BY Counterparties.Reference = SalesInvoice.Counterparty
ORDER BY
  Name HIERARCHY
```

Result:

Description	Number	Warehouse
Buyers		
Truck fleet N:34	0000012	Sales department warehouse
Alex-2002	0000003	Shopwindow in office
Alex-2002	0000016	Main warehouse
Alex-2002	0000008	Main warehouse
Alex-2002	0000013	Sales department warehouse
Rubber factory	0000005	Shopwindow in office
Rubber factory	0000018	Sales department warehouse
Store on GrandAvenue street	0000019	Sales department warehouse
Store on GrandAvenue street	0000011	Sales department warehouse
Store on GrandAvenue street	0000009	Main warehouse
Boutique	0000015	Main warehouse
Youth center	0000007	Main warehouse
Individuals		
Andrew Ryan	0000004	Main warehouse
Hugh Mortensen	0000010	Sales department warehouse

Fig. 171. Query result (fragment)

8.4.14.3. Ordering in nested tables

In the `ORDER BY` expression, is it also possible to define conditions for ordering of records from nested tables; where these can be combined with the ordering conditions of the top-level table.

While doing this, the order of specifying the one level table fields (nested or top level one) relative to each other is important, but it does not matter in what order the fields of a one level table are specified in relation to the fields of another level table: ordering is always performed first by the top-level table, and then by the nested table.

Example:

```
// The reports needs to contain the specification of invoices
// show the document itself, the products and quantity.
// Documents must be ordered by number,
// and the contents - by product name.

Select
  Document.SalesInvoice.Reference,
  Document.SalesInvoice.Composition.(Products AS Goods, Quantity)
(Документ.РасходнаяНакладная.Состав.(Номенклатура Как Товар,
Количество))
Order By
  Document.SalesInvoice.Number,
  Document.SalesInvoice.Contents.Products.Name
```

8.4.14.4. Auto-ordering results

The AUTOORDERING expression allows to enable the automatic generation of fields to order the query result.

Autoordering works based on the following principles:

- If the query contains the ORDER BY expression, then each reference to the table in this expression will be replaced by fields by which the table is sorted by default (for catalogs this is a code or name, for documents - the document date). If the field for ordering refers to a hierarchical catalog, then hierarchical sorting by this catalog will be applied.
- If the query does not contain the ORDER BY expression, but there is a RESULT expression, then the query result will be ordered by the fields present in the TOTALS expression after the BY keyword in the same sequence; and if the totals were calculated by reference fields, then by default sorting fields of the tables which were referenced.
- If the query does not contain ORDER BY and TOTALS expressions, but there is GROUP BY expression, then the query result will be ordered by the fields present in the expression, in the same sequence; and if the grouping was carried out by reference fields, then by default sorting fields of the tables which were referenced.
- If the query does not contain ORDER BY, TOTALS and GROUP BY expressions, the result will be sorted by default sorting fields for the tables from which the data is selected in the order in which they appear in the query.

8.4.15. Query totals calculation

TOTALS expression allows to determine the calculation of which totals is required in the request. When calculating the totals, the values of aggregate functions are calculated by the selection with the same values of control point fields-. Totals are added to the query result as totals rows.

The procedure for query totals calculating is described in accordance with the following rules:

<Totals description>

```
<Totals> [<Totals description>]
```

<Totals>

```
TOTALS [<Totals fields list>] BY [ОБЩИЕ] <Control points list>
```

<Totals fields list>

```
<Totals field> [,<Totals fields list> [, …]]
```

<Totals field>

```
<Aggregate function> | <Expression> [[КАК] <Field alias>]
```

<Control points list>

```
<Control point> [, <Control point> [, …]]
```

<Control point>

```
<Expression> [[ТОЛЬКО] HIERARCHY] | [IN PERIODS(Second | Minute | Hour
| Day | Week | Month | Quarter | Year | TenDays | HalfYear
[,<DATE type literal> | <Parameter ID>]
[,<DATE type literal> | <Parameter ID>])] [[КАК] Field alias]
```

Description of the totals begins with the mandatory `TOTALS` keyword.
The list of aggregate functions contains a list of aggregate functions, that must be calculated in the totals.
The `GRAND TOTALS` keyword means that it is required to generate a totals row for the entire query result.
In addition to the grand totals, it is possible to set the calculation of totals by control points. To do this, after the mandatory `BY` keyword, the `<Control points list>` must be specified. Each control point contains an expression that is calculated when executing the query. For each combination of values of these expressions, the totals rows will be calculated and added to the query result.
If the control point is a reference to the catalog, it is possible to calculate the totals by the catalog hierarchy. To do this, after such reference, a mandatory `HIERARCHY` keyword must be specified.

8.4.15.1. Calculation of totals in nested tables

The current software version does not support the calculation of totals by nested tables.

8.4.15.2. Totals by hierarchy

It is possible to calculate the totals by hierarchy. To do this, after the name of the field for which the totals are calculated, the `HIERARCHY` keyword must be specified**.** As a result, the totals for the control points and the totals by the hierarchy for the control points will be calculated.

Example:

```
SELECT
  Document.Products AS Products,
  Document.Quantity AS Quantity,
  Document.Reference.Number,
  Document.Reference.Counterparty
FROM
  Document.SalesInvoice.Contents AS Document
ORDER BY
  Document.Products
TOTALS
  SUM(Quantity)
TO
  Products HIERARCHY
```

Result:

Nomenclature	Quantity	Number	Contractor
⊟ Computer base units and components	24		
⊞ Hewlett-Packard base unit Brio BA410	9		
⊟ Hewlett-Packard base unit Vectra VL420	7		
├ Hewlett-Packard base unit Vectra VL420	1	0000010	Hugh Mortensen
├ Hewlett-Packard base unit Vectra VL420	3	0000008	Alex-2002
├ Hewlett-Packard base unit Vectra VL420	1	0000007	Youth center
├ Hewlett-Packard base unit Vectra VL420	1	0000018	Rubber factory
└ Hewlett-Packard base unit Vectra VL420	1	0000016	Alex-2002
⊟ IBM NetVista A22p base unit	4		
├ IBM NetVista A22p base unit	2	0000012	Truck fleet N:34
├ IBM NetVista A22p base unit	1	0000007	Youth center
└ IBM NetVista A22p base unit	1	0000014	Elbrus
⊞ IBM NetVista M41 base unit	4		

Fig. 172. Query result (fragment)

If necessary, it is possible to only calculate the totals of values by hierarchy, without calculating totals at control points. To do this, specify the `ONLY` keyword before the `HIERARCHY` keyword.

Example:

```
SELECT
  Document.Products AS Products,
  Document.Quantity AS Quantity,
  Document.Reference.Number,
  Document.Reference.Counterparty
FROM
  Document.SalesInvoice.Contents AS Document
ORDER BY
  Document.Products
TOTALS
  SUM(Quantity)
TO
  Products ONLY HIERARCHY
```

Result:

Nomenclature	Quantity	Number	Contractor
IBM NetVista A22p base unit	2	0000012	Truck fleet N:34
IBM NetVista A22p base unit	1	0000007	Youth center
IBM NetVista A22p base unit	1	0000014	Elbrus
IBM NetVista M41 base unit	1	0000012	Truck fleet N:34
IBM NetVista M41 base unit	1	0000007	Youth center
IBM NetVista M41 base unit	1	0000008	Alex-2002
IBM NetVista M41 base unit	1	0000016	Alex-2002
⊟- Monitors	38		
15' LG Studioworks 575N monitor	1	0000014	Elbrus
15' LG Studioworks 575N monitor	1	0000016	Alex-2002
15' LG Studioworks 575N monitor	1	0000004	Andrew Ryan
15' LG Studioworks 575N monitor	3	0000018	Rubber factory
15' LG Studioworks 575N monitor	5	0000011	Store on GrandAvenue street

Fig. 173. Query result (fragment)

8.4.15.3. Addition of dates

If the field for which the totals are calculated is a `Date` type field, it is possible to supplement the totals with dates in a given period. This is performed using the `BY PERIODS` keyword, after which the type of period (`Second`, `Minute`, `Hour`, `Day`, `Week`, `Month`, `Quarter`, `Year`, `TenDays`, `HalfYear`), the starting and ending dates of the period of interest are specified in parentheses. If the start and end dates are not specified, the first and last dates involved in the result will be used.

Example:

```
// Retrieve the number of purchases by customers in the hours of the
selected day
SELECT
  Invoice.Counterparty,
  BEGINOFPERIOD(Invoice.Date, HOUR) AS Period,
  COUNT(Invoice.Reference) AS PurchaseQuantity
FROM
  Document.Invoice AS Invoice
Group By
  Invoice.Counterparty,
  BEGINOFPERIOD(Invoice.Date, HOUR)
TOTALS
  SUM(PurchaseQuantity)
TO
  Period BY PERIODS(MINUTE, DATETIME(2006,6,28), DATETIME(2006,6,28))
```

Result:

Contractor	Period	QuantityPurchased
	06/28/2006 12:00:00	11
Temp_plus	06/28/2006 12:00:00	4
Maximus	06/28/2006 12:00:00	3
Novation	06/28/2006 12:00:00	2
Nielsen	06/28/2006 12:00:00	2
	06/28/2006 14:00:00	2
Temp_plus	06/28/2006 14:00:00	1
Maximus	06/28/2006 14:00:00	1

Fig. 174. Query result

This presentation of the result will be obtained only if all the records of the period are used as the source of measurement, when bypassing the result by `Period` grouping.

8.4.15.4. Grand totals calculation

To calculate the totals for the entire table, the `GRAND TOTALS` word must be specified in the `TOTALS` expression. In this case, the values of the aggregate functions will be calculated for all the records in the table.

Example:

```
SELECT
  Document.Products,
  Document.Quantity AS Quantity,
  Document.Reference.Number,
  Document.Reference.Counterparty
FROM
  Document.SalesInvoice.Contents AS Document
TOTALS
  SUM(Quantity)
BY
  OVERALL
```

Result:

Nomenclature	Quantity	Number	Contractor
	332		
1C:Bookkeeping 8. Basic version	2	0000001	Elbrus
1C:Bookkeeping 8.	2	0000001	Elbrus
15' LG Studioworks 575N monitor	5	0000002	Elbrus
Mouse 2-button A4Tech PS/2	10	0000002	Elbrus
Mouse LOGITECH M-S48 PS/2	10	0000002	Elbrus
Mouse OK-720 Mouse A4Tech PS/2	2	0000002	Elbrus
1C:Payroll and Human Resources 8	1	0000003	Alex-2002
Windows XP Home Edition Russian CD	1	0000004	Andrew Ryan
Keyboard Apple Pro Keyboards	1	0000004	Andrew Ryan

Fig. 175. Query result (fragment)

8.4.15.5. Joint use of totals and grouping

If totals are used together with grouping and no list of aggregate functions is specified for the totals, it will be automatically generated from the aggregate fields of the selection list. If the query contains a merger, the aggregate functions will be taken from the first query.

Example:

```
SELECT
  Document.Products AS Products,
  Document.Reference.Counterparty AS Counterparty,
  SUM(Document.Quantity) AS Quantity
FROM
  Document.SalesInvoice.Contents AS Document
Group By
  Document.Products,
  Document.Reference.Counterparty
TOTALS BY
  Products,
  Counterparty
```

Result:

Nomenclature	Contractor	Quantity
15' LG Studioworks 575N monitor		1
15' LG Studioworks 575N monitor	Elbrus	6
15' LG Studioworks 575N monitor	Elbrus	6
15' LG Studioworks 575N monitor	Alex-2002	1
15' LG Studioworks 575N monitor	Alex-2002	1
15' LG Studioworks 575N monitor	Store on GrandAvenue street	5
15' LG Studioworks 575N monitor	Store on GrandAvenue street	5
15' LG Studioworks 575N monitor	Rubber factory	3
15' LG Studioworks 575N monitor	Rubber factory	3
15' LG Studioworks 575N monitor	Andrew Ryan	1
15' LG Studioworks 575N monitor	Andrew Ryan	1
17' Philips 107S20 monitor		4
17' Philips 107S20 monitor	Alex-2002	2

Fig. 176. Query result (fragment)

8.4.15.6. Totals aliases

It is possible to assign an alias to the totals - control points fields for which the totals are counted, for subsequent accessing them from the 1C:Enterprise language. To do this, after the expression - control point, it is necessary to specify the name of the alias in the same way as it is done in the list of selection fields.

Example:

```
SELECT
  Document.Products AS Products,
  Document.Quantity AS Quantity,
  Document.Reference.Number,
  Document.Reference.Counterparty
FROM
  Document.SalesInvoice.Contents AS Document
ORDER BY
  Document.Products
TOTALS
  SUM(Quantity)
TO
  Products ONLY HIERARCHY AS Products
```

If the alias is not specified, the 1C:Enterprise itself will provide the name of the totals so that it is unique. In the example above, the totals will be named `Products`.

8.4.16. Expressions in query language

8.4.16.1. General information

Expressions can be used in many query language structures. Query language expressions are described by the following set of rules:

<Expression>

```
<Field dereferencing> |
<Aggregate function> |
<Built-in function> |
<Selection operation> |
<Type casting>[.<Field dereferencing>] |
<Value> |
<Expression> <Binary operation> <Expression> |
<Unary operation> <Expression> |
( <Expression> )
```

<Binary operation>

```
+ | - | * | /
```

<Unary operation>

```
- | +
```

In the simplest case, the expression is a reference to the field of the original table - query data source. A reference can be specified indicating the table containing this field or without specifying the table itself.
The lines of unlimited length are not supported during the addition + operations. Strings longer than 1024 characters are treated as strings of unlimited length.

Expressions in the list of selection, in `HAVING`, `TOTALS`, `ORDER BY` expressions, can be aggregate functions.
An expression can be a built-in query language function. Selection operations and value type cast operations can be used.
The expressions can directly indicate the values of logical, numeric, string and other constants; the values of the query parameters can also be used. Binary and unary operations can be applied to values of the corresponding types in the expressions. When using the division operation, it is recommended to perform a check of the divisor for difference from zero.
The type of `SUM()`, `AVG()`, `CAST()`, `*`, `+`, `-` operations, applied to numeric types will be a number with arbitrary precision and bitness.
In the arithmetic calculations, the platform tries to provide the accuracy not lower than 8 decimal places. If the DBMS allows higher accuracy (in any situation) the platform does not force the rounding to 8 decimal places. It means that the use of a request to another DBMS (including the file variant) may result in another accuracy of an arithmetic calculation. If an arithmetic operation applied to the constant expressions is used in the request language expression it may be calculated without sending to the DBMS and its accuracy may be different as compared to the calculation result obtained at the DBMS.

8.4.16.2. Fields dereferencing

In the simplest case, expressions of the query language are references to the fields of the tables of the infobase. In general, the references are described by the following rules:

<Field dereferencing>

```
[<Table>.]<Field name>[.<Field name>[…]]
```

<Table>

```
<Table name> | <Source alias>
```

The dereferencing of a field begins with the name of the table containing this field. If the field name uniquely -exists in only one table among the indicated in the list of sources, the table can be omitted.
If the field has a reference type, the query language allows to access the fields of the table referenced by the field, and so on. Field names are specified separated by a dot.
If the source table in the source list is assigned a source alias, it can be used instead of the table name in the field dereferencing of this table. Otherwise, the table name is specified.

8.4.16.3. Aggregate functions of query language

The query language provides aggregate functions that are used when grouping query results and when calculating totals. Aggregate functions are intended to

summarize the values of the specified parameter. The following aggregate functions are defined:

<Aggregate function>

```
SUM ( <Expression> ) |
AVG ( <Expression> ) |
MIN ( <Expression> ) |
MAX ( <Expression> ) |
COUNT ( [РАЗЛИЧНЫЕ] <Expression> | * )
```

Example:

```
SELECT
  Invoice.Products.Name,
  SUM(Invoice.Sum) AS Sum,
  AVG(Invoice.Sum) AS Avg,
  MAX(Invoice.Sum) AS Max,
  MIN(Invoice.Sum) AS Min,
  Count(Invoice.Sum) AS Count
FROM
  Document.SalesInvoice.Content As Invoice
Group By
  Invoice.Products,
  Invoice.Products.Name
TOTALS BY
  OVERALL
```

Result:

NomenclatureDescription	Amount	Average	Maximum	Minimum	Quantity
	104,086.7	873.606641178571...	16,600.00	1.10	28
Keyboard Apple Pro Keyboards	1,690	241.428571	375.00	75.00	7
17" Philips 107S20 monitor	840	210	210.00	210.00	4
1C:Bookkeeping 8.	280	280	280.00	280.00	1
IBM NetVista M41 base unit	5,800	1,450	1,450.00	1,450.00	4
Windows XP Home Edition Russian UPG CD	255	127.5	170.00	85.00	2
15" LG Studioworks 575N monitor	2,400	400	750.00	155.00	6
Laser printer Canon LBP-810	2,100	350	600.00	300.00	6
Hewlett-Packard base unit Brio BA410	18,000	3,000	6,000.00	2,000.00	6
Mouse Ice Mouse MUS-2	39.8	5.685714	13.00	1.40	7

Fig. 177. Query result (fragment)

Aggregate functions can be used in the list of selection fields, in the `HAVING`, `TOTALS`, `ORDER BY` expressions. Aggregate functions should be used in the list of selection fields to obtain a result only. It is not recommended to use aggregate functions in the list of selection fields for comparison.

Correct query:

```
SELECT
    COUNT(*),
    Vendor
FROM
  Document.SalesInvoice As Invoice
Group By
  Vendor
HAVING
  COUNT(*) > 1
```

Incorrect query:

```
SELECT
    WHEN COUNT(*) > 1 THEN True ELSE False END,
    Vendor
FROM
  Document.SalesInvoice As Invoice
Group By
  Vendor
```

Despite the fact that an incorrect query can operate successfully, such structures (use of the aggregate function not for obtaining the result, but for checking the condition) in the `SELECT` section have their own specifics on different DBMS and can cause errors.

SUM

Description:

The function calculates the arithmetic sum of all the field values that were included in the selection.

It is possible to specify only the fields that contain a numeric value as the function parameter.

If a field cannot contain numeric values, then using the `SUM` function to such field will cause an error. This function can be applied to such field only if the field can contain numeric values (has a composite data type). But if there is a non-numeric value (in addition to `NULL` values) among the values of the field in the selection, this will cause an error.

AVG

Description:

The function calculates the average value of all the field values that were included in the selection.

It is possible to specify only the references to the fields that contain a numeric value as the function parameter.

If a field cannot contain numeric values, then using the `AVERAGE` function to such field will cause an error. This function can be applied to such field only if the field can contain numeric values (has a composite data type). But if there is

a non-numeric value (in addition to NULL values) among the values of the field in the selection, this will cause an error.

MIN

Description:

The function calculates the minimum value of all the field values that were included in the selection.

It is possible to specify the references to the fields that contain any value types as the function parameter.

When determining the minimum value, the rules for comparing values are applied, as described in the “Value comparison rules” section.

MAX

Description:

The function calculates the maximum value of all the field values that were included in the selection.

You can specify an expression containing values of any type as a function parameter. Cannot be applied to expressions with ValueStorage, PointInTime type and strings of unlimited length.

When determining the maximum value, the rules for comparing values are applied, as described in the “Value comparison rules” section.

COUNT

Description:

The function counts the number of parameter values that included in the selection. The COUNT function has three usage scenarios in contrast to the other aggregate functions:

- The function allows to count the number of values of the specified field, that are not equal to NULL.
- The function allows to count the number of different values of the specified field, that are not equal to NULL. To do this, before specifying the field, the DISTINCT keyword must be set.
- The function allows to count the number of rows in the query result. To do this, it is necessary to specify an asterisk “*” as a function parameter.

It is possible to specify the references to the fields that contain any value types as the function parameter.

Example:

```
SELECT
  COUNT(*) AS Total,
  Count(DISTINCT Invoice.Products) AS Distinct
FROM
  Document.SalesInvoice.Content As Invoice
```

Result:

Total	Different
120	28

Fig. 178. Query result

8.4.16.4. Built-in functions of query language

Built-in functions are defined in the query language that can be used in expressions in the list of selection fields and in the selection condition in the `WHERE` expression.

The following built-in functions are defined:

<Built-in function>

```
SUBSTRING ( <Expression>, <Value>, <Value> ) |
YEAR ( <Expression> ) |
QUARTER ( <Expression> ) |
MONTH ( <Expression> ) |
DAYOFYEAR ( <Expression> ) |
DAY ( <Expression> ) |
WEEK ( <Expression> ) |
WEEKDAY ( <Expression> ) |
HOUR ( <Expression> ) |
MINUTE ( <Expression> ) |
SECOND ( <Expression> ) |
BEGINOFPERIOD(<Expression>, Minute | Hour | Day | Week | Month |
Quarter | Year | TenDays | HalfYear) |
ENDOFPERIOD(<Expression>, Minute | Hour | Day | Week | Month | Quarter
| Year | TenDays | HalfYear) |
DATEADD (<Expression>, Minute | Hour | Day | Week | Month | Quarter |
Year | TenDays | HalfYear, <Expression>) |
DATEDIFF(<Expression>, <Expression>, Minute | Hour | Day | Week |
Month | Quarter | Year) |
VALUETYPE(<Expression>) |
PRESENTATION(<Expression>) |
ISNULL(<Expression>, <Expression>) |
AUTORECORDNUMBER( )
GROUPEDBY(<Expression>)
```

SUBSTRING

Description:

This function extracts a substring from a string. Three parameters are passed to the function:

- The string to extract a substring from. `String` type expression.
- The character position where the extracted substring begins. `Number` type value.
- The substring length. `Number` type value.

If the first parameter is a string, the result is a string (it can have zero length). If the first parameter is `NULL`, the result is also `NULL`. Other values are considered invalid and will cause an error.

Example:

```
SELECT FIRST EIGHT
  Counterparties.Name,
  SUBSTRING(Counterparties.Name, 3, 5) AS Substring
FROM
  Catalog.Counterparties AS Counterparties
```

Result:

Description	Substring
Temp_plus	Mp pl
Vendors	Endo
Maximus	Aximu
Novation	Vation
Nielsen	Elsen
Electronica	Ctroni
Buyers	Uyer
Elbrus	brus

Fig. 179. Query result

YEAR

Description:

This function calculates the year number from a `Date` type value.

Function parameter - is a `Date` type expression.

If the parameter is the `Date` type value, then the result of the function will be the `Number` type value. If the parameter is `NULL`, the result is also `NULL`. Other values are considered invalid and will cause an error.

QUARTER

Description:

This function returns a quarter number from a `Date` type value. The quarter number is in the range of 1–4.

Function parameter - is a `Date` type expression.

If the parameter is the `Date` type value, then the result of the function will be the `Number` type value. If the parameter is `NULL`, the result is also `NULL`. Other values are considered invalid and will cause an error.

MONTH

Description:

This function calculates a month number from a `Date` type value. The month number is in the range of 1–12.

Function parameter - is a `Date` type expression.

If the parameter is the `Date` type value, then the result of the function will be the `Number` type value. If the parameter is `NULL`, the result is also `NULL`. Other values are considered invalid and will cause an error.

DAYOFYEAR

Description:

This function returns a day of the year from a `Date` type value. The day of the year is in the range of 1–366.
Function parameter - is a `Date` type expression.
If the parameter is the `Date` type value, then the result of the function will be the `Number` type value. If the parameter is `NULL`, the result is also `NULL`. Other values are considered invalid and will cause an error.

DAY

Description:

This function returns a day of the month from a `Date` type value. The day of the month is in the range of 1–31.
Function parameter - is a `Date` type expression.
If the parameter is the `Date` type value, then the result of the function will be the `Number` type value. If the parameter is `NULL`, the result is also `NULL`. Other values are considered invalid and will cause an error.

WEEK

Description:

This function returns a week number from a `Date` type value.
Function parameter - is a `Date` type expression.
If the parameter is the `Date` type value, then the result of the function will be the `Number` type value. If the parameter is `NULL`, the result is also `NULL`. Other values are considered invalid and will cause an error. The calculation considers the value of the First weekday property of the infobase regional settings.

WEEKDAY

Description:

This function returns a day of the week from a `Date` type value. The day of the week is in the range from 1 (Monday) - to 7 (Sunday).
Function parameter - is a `Date` type expression.
If the parameter is the `Date` type value, then the result of the function will be the `Number` type value. If the parameter is `NULL`, the result is also `NULL`. Other values are considered invalid and will cause an error.

HOUR

Description:

This function returns the hour of the day from a `Date` type value. The hour of the day is in the range of 0–23.

Function parameter - is a Date type expression.
If the parameter is the Date type value, then the result of the function will be the Number type value. If the parameter is NULL, the result is also NULL. Other values are considered invalid and will cause an error.

MINUTE

Description:

This function returns the minute of an hour from a Date type value. The minute of the hour is in the range of 0–59.
Function parameter - is a Date type expression.
If the parameter is the Date type value, then the result of the function will be the Number type value. If the parameter is NULL, the result is also NULL. Other values are considered invalid and will cause an error.

SECOND

Description:

This function returns the second of a minute from a Date type value. The second of a minute is in the range of 0–59.
Function parameter - is a Date type expression.
If the parameter is the Date type value, then the result of the function will be the Number type value. If the parameter is NULL, the result is also NULL. Other values are considered invalid and will cause an error.

BEGINOFPERIOD

Description:

This function extracts a certain date from a given date.
Function parameters - are an expression of the Date type and the period type - one of the values: Minute, Hour, Day, Week, Month, Quarter, Year, TenDays, HalfYear. If the period type is set to Week, then the value of the First weekday property of the infobase regional settings is considered.

Example:

```
SELECT BEGINOFPERIOD(DATETIME (2002, 10, 12, 10, 15, 34), MONTH)
```

Result:

Field 1
10/01/2002

Fig. 180. Query result

Example:

```
SELECT BEGINOFPERIOD(DATETIME(2002, 10, 12, 10, 15, 34), DAY)
```

Result:

Field 1
01/12/2002

Fig. 181. Query result

ENDOFPERIOD

Description:

This function extracts a certain date from a given date.

Function parameters - are an expression of the `Date` type and the period type - one of the values: `Minute`, `Hour`, `Day`, `Week`, `Month`, `Quarter`, `Year`, `TenDays`, `HalfYear`. If the period type is set to `Week`, then the value of the First weekday property of the infobase regional settings is considered.

Example:

```
SELECT ENDOFPERIOD(DATETIME(2002, 10, 12, 10, 15, 34), MONTH)
```

Result:

Field 1
10/31/2002 23:59:59

Fig. 182. Query result

Example:

```
SELECT ENDOFPERIOD(DATETIME(2002, 10, 12, 10, 15, 34), YEAR)
```

Result:

Field 1
12/31/2002 23:59:59

Fig. 183. Query result

DATEADD

Description:

This function adds a value to a date.

The first parameter - is the initial date, to the value of which the specified value determined by the second and third parameters is added; expression of `Date` type.

The second parameter - is the type of increase, one of the values: `Second`, `Minute`, `Hour`, `Day`, `Week`, `Month`, `Quarter`, `Year`, `TenDays`,

HalfYear. Note that the Decade term means exactly 10 days, regardless of the date specified in the first parameter of the function.
The third parameter - is the value by which it is necessary to increase the date specified by the first parameter; Number type (decimal section is ignored).
It is recommended to specify the third parameter (dimension of magnification) so that it corresponds to the applied task being solved. So, if an applied task operates with days or months, then days and months must be added to the date. Using large numbers as the values of the second and third parameters can cause the following problems:

- Different implementation of daylight saving transition in different DBMS.
- Going beyond the range of dates supported by the DBMS.
- Going beyond the permissible parameter values.

Example:

```
SELECT DATEADD(DATETIME(2002, 10, 12, 10, 15, 34), MONTH, 1)
```

Result:

Fig. 184. Query result

Example:

```
SELECT DateAdd(DateTime(2002, 10, 12, 10, 15, 34), Day, 5)
```

Result:

Fig. 185. Query result

DATEDIFF

Description:

This function returns an interval between two dates.
The first parameter - is an expression of Date type. The second parameter - is an expression of Date type. The third parameter - is the type of difference, one of the values: Second, Minute, Hour, Day, Month, Quarter, Year.
It is recommended to specify the third parameter (type of difference) so that it corresponds to the applied task being solved. Thus, if the applied task operates with days or months, then it is necessary to calculate the difference of dates in days or months. It is not recommended to retrieve inadequate (from the applied task point of view) difference of dates, and then bring it to the desired difference using the means of the query language or the 1C:Enterprise language. This recommendation is due to the fact that the calculation of the difference with a very

high discreteness (seconds, minutes, hours) can take different account of the daylight saving transitions on different DBMS.
The use of this function is not supported when setting the `Period` virtual table parameter value.

Example:

```
SELECT DATEDIFF(DATETIME(2002, 10, 12, 10, 15, 34), DATETIME(2002, 10,
14, 9, 18, 06), DAY)
```

Result:

Field 1
2

Fig. 186. Query result

Example:

```
SELECT DATEDIFF(DateTime(2002, 10, 12), DateTime(2002, 11, 03), MONTH)
```

Result:

Field 1
1

Fig. 187. Query result

IMPORTANT! The function returns the calendar difference between two dates, so it cannot be used where it is necessary to calculate the number of banking or business days between two dates.

VALUETYPE function

Description:

The function returns the type of value in the query.

Parameters:

The function parameter - is an expression of any type.
Return value - is a `Type` value.
If the function parameter is `Undefined`, then the return value will also be `Undefined`.

Example:

```
SELECT
  VALUETYPE(ProductsAccounting.Recorder) AS Document
FROM
  AccumulationRegister.ProductsAccounting AS ProductsAccounting
```

PRESENTATION function

Description:

This function returns a string presentation of an arbitrary type value.

The function parameter - is an expression of any type.

Return value - is a representation of the value, `String` type.

The function result cannot be used inside other functions, except for the `PRESENTATION` function.

Example:

```
SELECT
  PRESENTATION(Document.Counterparty) AS Recipient,
  PRESENTATION(Document.Date) AS Date
FROM
  Document.SalesInvoice AS Document
```

Result:

Recipient	Date
Elbrus	06/28/2006 14:19:00
Elbrus	06/28/2006 14:30:32
Alex-2002	06/28/2006 14:30:49
Andrew Ryan	06/28/2006 14:32:06
Rubber factory	06/28/2006 14:32:32
Williams	06/28/2006 14:32:47
Youth center	06/28/2006 14:34:04
Alex-2002	06/28/2006 14:35:37
Store on GrandAvenue street	06/28/2006 14:36:05
Hugh Mortensen	06/28/2006 14:36:36

Fig. 188. Query result

ISNULL function

Description:

The function replaces the `NULL` value with another value.

Function parameters:

- the first parameter - is an expression of any type;
- the second parameter - is an expression of any type.

Returns: the value of the first parameter, if the first parameter does not contain a `NULL` value; otherwise, the value of the second parameter.

The second parameter will be converted to the type of the first one if the type of the first parameter is a string or a number.

Example:

```
// Retrieve the sum of the Quantity field. If there are no
 // entries, return 0
SELECT
  ISNULL(SUM(SalesInvoiceContents.Quantity), 0)
FROM
  Document.SalesInvoice.Contents AS SalesInvoiceContents
```

AUTORECORDNUMBER

Description:

The function generates a field with a unique, consistently increasing value in the temporary table. The function may be used if it is necessary to create a key field in a temporary table. The value generated by the function will be unique within the temporary table, during the generation of which the function was used. The counter initial value depends on the used DBMS and, in the general case, can be any. It is not guaranteed that the counter initial value will be 1 for any temporary table.

The use of the `AUTORECORDNUMBER()` function is not supported in the following cases:

- in queries containing `MERGE` at the top level,
- in queries that do not generate a temporary table,
- outside the selection list;
- in query language expressions.

Example:

```
SELECT
  PaymentsToVendor.Vendor AS Vendor,
  PaymentsToVendor.Currency AS Currency,
  SUM(PaymentsToVendor.Sum) AS Sum,
  AUTORECORDNUMBER() AS Key
PLACE Payments
FROM
  Document.PaymentsToVendor AS PaymentsToVendor

GROUP BY
  PaymentsToVendor.Vendor,
  PaymentsToVendor.Currency
```

Result:

As a result, a `Payments` temporary table will be created, where there will be a sequentially increasing numerical value as a value of the `Key` field.

GROUPEDBY

Description:

This function can be used to define grouping in this string by a field specified as a function parameter.

Parameters:

Parameter - verified field.

The functions returns `True`, if a function parameter available in this string is used for grouping purposes and otherwise it will be `False`.

Example:

```
SELECT
  CASE
    WHEN GROUPEDBY(ProductAccountingStock.Goods)
    ProductAccountingStock.Goods
    OTHERWISE "<< All goods >>"
  END AS Goods,
  CASE
    WHEN GROUPEDBY(ProductAccountingStock.Warehouse)
    ProductAccountingStock.Warehouse
    OTHERWISE "<< All warehouses >>"
  END AS Warehouse,
  SUM (ProductAccountingStock.AmountStock) AS AmountStock
FROM
  AccumulationRegister.Inventory.Stock AS ProductAccountingStock

GROUP BY GROUPING SETS
(
  (ProductAccountingStock.Warehouse),
  (ProductAccountingStock.Goods)
)
```

Result:

Upon execution of this query the fields which are not used in each result string for grouping purposes specify `<< All goods >>` or `<< All warehouses >>` values (depending on a field this function is applied to).

8.4.16.5. Selection operations in query language

Query language expressions can use selection operations that allow one of the possible values to be retrieved in accordance with the specified conditions.

The selection operation is described by the following set of rules:

<Selection operation>

```
CASE
<Selection alternatives>
[ELSE <Expression>]
END
```

<Selection alternatives>

```
<Single choice>
[<Selection alternatives>]
```

<Single choice>

```
WHEN <Logical expression>
THEN <Expression>
```

An unlimited number of alternative single choices can be specified in a `WHEN ... THEN` select operation. These are processed sequentially in the querry. If the logical expression is `True`, the selection operation processing is completed. The result of the operation is the value of the expression specified after the `THEN` word.

The value of the expression specified after the `ELSE` word is used as the result of the selection operation, if the predicate was `False` in all the previously specified alternative single choices.

Example:

```
SELECT
  Products.Name,
  CASE
    WHEN Products.IsFolder = True
      THEN "This is group"
    WHEN Products.PurchasePrice > 1000
      THEN "1000  "
    WHEN Products.PurchasePrice > 100
      THEN "100 - 1000"
    When Products.PurchasePrice > 10
      THEN "10 - 100"
    WHEN Products.PurchasePrice > 0
      THEN "0 - 10"
    ELSE "Not specified"
  END AS Price
FROM
  Catalog.Products AS Products
```

Result:

Description	Price
Monitors	This group
Printers	This group
Mouses	This group
Keyboards	This group
Software	This group
Services	This group
15' LG Studioworks 575N monitor	100 – 1000
17' Philips 107S20 monitor	100 – 1000
19' Hitachi CM715ET monitor	100 – 1000
LCD 22' M8537ZM/A monitor	1000 -
Hewlett-Packard base unit Brio BA410	1000 -
Hewlett-Packard base unit Vectra VL420	1000 -

Fig. 189. Query result (fragment)

8.4.16.6. Type casing in query language

Fields in source tables can be of a composite type. For such fields, it is necessary to bring the field values to a specific type. The query language provides the feature to cast the type. It can be used in the list of selection fields and in the selection condition in the `WHERE` expression.

<Type casting>

```
CAST ( <Expression> AS <Value type> )
```

<Value type>

```
Boolean |
Number [(Length[, Precision])]|
String [(Length)]|
Date |
<Table name>
```

<Length>

```
Number
```

<Precision>

```
Number
```

The expression is cast to one of the primitive types or to the reference data type. In the latter case, the table name referes to the corresponding infobase table.
If an expression contains a required type of value within a composite type, then the type casting is considered feasible, and for each value of the specified type, the result will be the same value. For values of other types, the result of the cast will be `NULL`.
If the expression does not contain the required type of value within the composite type, then the execution of this query will fail with an error due to the impossibility to perform a type casting.

8.4.16.7. Constants and parameters in query language

Query language expressions can directly indicate values of the `Boolean`, `Number`, `String` or `Date` type. Query parameter values may also be used.

<Value>

```
TRUE |
FALSE |
<NUMBER type literal> |
<STRING type literal> |
<DATE type literal> |
<TYPE type literal> |
<Parameter name> |
UNDEFINED |
NULL
```

<Number type literal>

```
<Integer>[.<Integer>]
```

<String type literal>

```
"<Character sequence>"
```

<Date type literal>

```
DATETIME ( <Integer>, <Integer>, <Integer>[, <Integer>, <Integer>,
<Integer>] )
```

Values of the `Boolean`, `Number`, `String` type in the query language are set in the same way as in the 1C:Enterprise language.
Values of the `Date` type are specified using the `DATETIME` keyword, after which the year, month, day, hour, minute, second are successively indicated in parentheses. The last three values are optional.

IMPORTANT! The maximum date that can be set using `DATETIME` literal is - `31.12.3999 23:59:59`.

Parameters can be passed to the query (see the description of the `Query` object). Parameter values can be used in query language expressions. To do this, it is necessary to specify the "`&`" character followed by the parameter name.

<Type type literal>

```
TYPE(<Type name>)
```

`<Type name>` - the name of the primitive type or the name of the table which link type is to be retrieved. The result of this structure is the value of the `Type` type for the specified type.

Example:

```
// Retrieving String type
TYPE(String)

// Retrieving type - reference to the Products catalog
TYPE(Catalog.Products)
```

`Type` type values in the query language can be used in comparison, ordering and grouping operations.

Example:

```
SELECT
  VALUETYPE(Balance.Recorder)
FROM
  AccumulationRegister.ProductsAccounting AS Balance
WHERE
VALUETYPE(Balance.Recorder) = TYPE(Document.SalesInvoice)
```

It is possible to pass a `Type` type value as a query parameter.

Example:

```
SELECT
  VALUETYPE(Balance.Recorder)
FROM
  AccumulationRegister.ProductsAccounting AS Balance
WHERE
  VALUETYPE(Balance.Recorder) = &Type
```

When comparing values of `Type` type, the values are ordered as follows (the first type is considered the smallest):

- `NULL`,
- `Undefined`,
- `Boolean`
- `Number`
- `Date`
- `String`
- reference to a table,
- other types.

8.4.17. Conditions in query language

8.4.17.1. General information

In the query language, the selection conditions are used, in accordance with which the data are filtered in `WHERE`, `HAVING` and `CONNECTION` expressions. The conditions are described according to the following rules:

<Filter condition>

```
<Logical addend> [Or <Logical addend>]
```

<Logical addend>

```
<Logical multiplier> [And <Logical multiplier>]
```

<Logical multiplier>

```
Not <Logical multiplier> |
( <FIlter condition> ) |
<Logical expression>
```

In the simplest case, the condition is an expression the result of which is a logical type value. Logical expressions are described in the next section.
Conditions can also be defined as more complex logical expressions, where simple logical expressions appear, interconnected by means of logical operators `And`, `Or`, `Not`.
Logical operators have the following priorities:

- Logical operator `Not` has the highest priority.
- Operator `And` in the next in priority.
- The operator `Or` has the lowest priority.
- In the conditions, simple logical expressions are calculated first, then `Not` operations, then `And` operations, and, finally, - `Or` operations. Parentheses `( )` can be used to ensure a different calculation order.

8.4.17.2. Logical expressions in query language

Logical expressions are used in the query language in the selection operations and in the filter conditions.

<Logical expression>

```
<Expression> |
(<Expression> | <Logical expression>) <Comparison operation>
(<Expression> | <Logical expression>) |
<Expression> [He] IN [ИЕРАРХИИ] (<Value list>) |
<Expression> [He] IN [ИЕРАРХИИ](<Query description>) |
<Expression> [He] BETWEEN <Expression> AND <Expression> |
<Expression> IS [He] NULL |
<Expression> REFERENCE <Table name> |
<Expression> [He] LIKE <STRING type literal>
     [ESCAPE <STRING type literal>]
```

<Comparison operation>

```
> | < | = | >= | <= | <>
```

<Value list>

```
<Expression>[, <Expression> [, …]]
```

A logical expression can be:

- the usual expression of the query language, if its result has a logical type;
- the operation of comparing two expressions of the query language; performed in accordance with the rules for comparing values described in the section “Value comparison rules”;
- operator checking the match/mismatch of the expression value with one of the listed values or with the values contained in the result of another query;
- operator checking the occurrence of the expression in the range;
- operator checking the value of an expression for `NULL`;
- operator checking the reference value of the expression for a reference to a specific table;
- operator checking the string value by template.

When comparing values, the rules for comparing values are described below.

8.4.17.3. Value comparison rules

This section describes the rules for comparing values of different types in a query, as the values of different types may be compared in the query language.
These rules are used:

- to compare values in comparison operators;
- to determine the maximum and minimum values in the aggregate functions `MIN` and `MAX`;
- to arrange the records of the query result in accordance with the order specified in the `ORDER BY` expression.

If value types differ from each other, the relationship between the values is defined based on type priorities:

- `NULL` type (the lowest priority);
- `Неопределено` type;
- `Boolean` type;
- `Number` type;
- `Date` type;
- `String` type;
- reference types

The relationship between different reference types is defined based on their internal table reference numbers corresponding to one type or another.
If the data types are identical, their values are compared according to the following rules:

- For `Boolean` type, `True` is greater than `False`.
- For `Number` type, common number comparison rules are applied.
- For `Date` type, earlier dates are less than later dates.
- For `String` type, the comparison is performed in accordance with the established national characteristics of the database and the specified sorting order. The comparison is case insensitive. The string comparison operation

does consider trailing spaces, unlike string comparisons in the 1C:Enterprise language, where trailing spaces are included during the comparison operation. For example, comparing strings “bb” and “bb ” in the 1C:Enterprise language will return False (strings are not equal), and True in the query language (strings are equal). At the same time, it is not guaranteed that the sorting of the same data set performed on different DBMS will return the same result.

- Reference types and values of UUID type are compared based on their values. At the same time, the repeatability of the comparison result is guaranteed only within one database.
- Comparison of fields of UUID type with fields of other types is not supported;
- It is not allowed to compare fields of unlimited length (strings of unlimited length, ValueStorage, ValueType field from the chart of characteristic types table).

If the operation of comparing two values is used, where at least one NULL value is involved; NULL will be the result of the operation. If the result of a conditional expression (for example, in the WHERE, BY, WHEN expressions) is NULL, the condition is considered not to be satisfied.

8.4.17.4. Value matching operator

The form of the "IN" operator to check the coincidence with one of the listed

IN operator allows to check whether the value of the expression specified to the right from it matches one of the values described to the left. If at least one of them matches, the result of the operator will be True, otherwise - False. Application of Not changes the operator effect to the opposite one. Comparison of values is performed according to the rules described in the “Value comparison rules” section.

Example:

```
SELECT
  Products.Name
FROM
  Catalog.Products AS Products
WHERE
  Products.Parent.Name IN
    ("Household appliances", "Office equipment")
```

The form of the "IN" operator to check the belonging by hierarchy

For catalogs, the check can also be carried out for belonging to the hierarchy. The result of the IN HIERARCHY operator will be True if the value of the expression to the left is a reference to the catalog item and is included in the set

of values to the right or hierarchically belongs to some group contained in this set.

Example:

```
// As a Group parameter, the reference is passed to the request
// to some group of Products catalog.

SELECT
  Products.Name
FROM
  Catalog.Products AS Products
WHERE
  Products.Reference IN HIERARCHY(&Group)
```

The result of the query may also appear as a set of values for which a check is performed. In this case, it is necessary to specify a query description to the right from the `IN` operator.

Example:

```
SELECT
  Products.Name
FROM
  Catalog.Products AS Products
WHERE
  Products.Reference IN HIERARCHY
    (SELECT
      Products.Reference
    FROM
      Catalog.Products AS Products
    WHERE
      Products.Name = "Clothes")
```

The form of the "IN" operator to check the coincidence with one of the query results

Operator use example:

Example:

```
// Select the names of the products that were present
// in sales invoices
SELECT
  Products.Name
FROM
  Catalog.Products AS Goods
WHERE
  Products.Reference IN
    (SELECT
      SalesInvoiceContents.Products
    FROM
      Document.SalesInvoice.Contents AS SalesInvoiceContents)
```

Result:

Description
15' LG Studioworks 575N monitor
17' Philips 107S20 monitor
19' Hitachi CM715ET monitor
LCD 22' M8537ZM/A monitor
Hewlett-Packard base unit Brio BA410
Hewlett-Packard base unit Vectra VL420
IBM NetVista A22p base unit
IBM NetVista A22p base unit
Laser printer Canon LBP-810
Laser printer 5250197-203 Minolta-QMS
Laser printer HP LaserJet 2200
Mouse 2-button A4Tech PS/2

Fig. 190. Query result (fragment)

To retrieve the opposite result, that is, if it is necessary to determine that the value does not match any of the query results, the query will look as follows.

Example:

```
// Select the names of the products that were present
// in sales invoices
SELECT
  Products.Name
FROM
  Catalog.Products AS Goods
WHERE
  (Not Products.Reference IN
    (SELECT
      SalesInvoiceContents.Products
    FROM
      Document.SalesInvoice.Contents AS SalesInvoiceContents))
```

Result:

Description
Monitors
Printers
Mouses
Keyboards
Software
Services
Windows
Delivery
SW installation
OS Windows setup consulting
1C setup consulting
Notebooks

Fig. 191. Query result (fragment)

Note that from the `IN` operation query, it is possible to access the fields of the tables that were present in the external query before the operation appeared.

Example:

```
// Select the names of the products that were present
// in sales invoices
SELECT
  Products.Name
FROM
  Catalog.Products AS Goods
WHERE
  Products.Reference IN
  (SELECT
    SalesInvoiceContents.Products
  FROM
    Document.SalesInvoice.Contents AS SalesInvoiceContents
  WHERE
    SalesInvoiceContents.Products = Goods.Reference)
```

Result:

Description
15' LG Studioworks 575N monitor
17' Philips 107S20 monitor
19' Hitachi CM715ET monitor
LCD 22' M8537ZM/A monitor
Hewlett-Packard base unit Brio BA410
Hewlett-Packard base unit Vectra VL420
IBM NetVista A22p base unit
IBM NetVista A22p base unit
Laser printer Canon LBP-810
Laser printer 5250197-203 Minolta-QMS
Laser printer HP LaserJet 2200
Mouse 2-button A4Tech PS/2

Fig. 192. Query result (fragment)

The result of the `IN` operator expression with the subquery can have two values (`True`/`False`). This means that the third variant of the result (`NULL` - result of comparison with `NULL`) can be interpreted as false.

If the `IN` operator with the subquery has an argument to the left from `IN` and the argument in the subquery of different types, the `NULL` result is excluded from the possible results of `IN` operator. The `NULL` result in this case is interpreted as false.

Using "IN" ("Not IN") operation for several fields

Syntax for nested query:

`(expression1, expression2, …, expressionN) IN (SELECT expression1, expression2, …, expressionN …)`

Syntax for values table:

`(expression1, expression2, …, expressionN) IN (&Parameter)`

The first N columns of table of values to be used for `IN` operation must be passed as a parameter.

8.4.17.5. Operator checking the occurrence of the value in the range

The BETWEEN operator allows to check whether the value of the expression specified to the left from it is in the range specified to the right. If it is included, the result of the operator will be True, otherwise - False. Application of Not changes the operator effect to the opposite one.

Example:

```
SELECT
  Products.Name,
  Products.PurchasePrice
FROM
  Catalog.Products AS Products
WHERE
  Products.PurchasePrice BETWEEN 100 AND 1000
```

8.4.17.6. Operator checking the value for NULL

The ISNULL operator allows to check the value of the expression to its left for NULL. If the value is NULL, the result of the operator will be True, otherwise - False. Application of Not changes the operator effect to the opposite one.

Example:

```
SELECT
  Goods.Name AS Name,
  Prices.Price AS Price
FROM
  Catalog.Products AS Goods
    LEFT CONNECTION InformationRegister.ProductPrices AS Prices
    BY Goods.Reference = Prices.Products
WHERE
  Prices.Price IS NULL
```

8.4.17.7. Operator to check reference value

The REFERENCE operator allows to check whether the value of the expression specified to the left from it is a reference to a table specified to the right. If yes, the result of the operator will be True, otherwise - False.

Example:

```
SELECT
  SalesInvoice.Number,
  SalesInvoice.Date
FROM
  Document.SalesInvoice AS SalesInvoice
WHERE
  SalesInvoice.Counterparty REFERENCE Catalog.Counterparties
```

8.4.17.8. Operator to check the string for template similarity

The `LIKE` operator allows to compare the value of the expression specified to its left with the template string specified to the right. The value of the expression must be of `String` type. If the expression value matches the template, the result of the operator will be `True`, otherwise - `False`.

The following characters in the template string are service ones and have a meaning different from the string character:

- "`%`" (percent sign): a sequence containing any number of arbitrary characters.
- «`_`» (underscore): Any one character.
- «`[...]`» (one or more characters in brackets): any single character from the ones listed in brackets. A character set can contain ranges, such as `a-z`, meaning any character within the range, including the limits of the range.
- "`[^...]`" (a negation sign followed by one or more characters in brackets). any character except those listed after the negation sign.

Any other character represents itself only and does not have any additional meaning.

If it is necessary to write one of the listed characters as itself, then a special character must precede it. The special character itself (any suitable character) is defined in the same operator after the `ESCAPE` keyword.

For example, the template:

```
"%АБВ[0-9][абвг][\[\]\    ]\_abc%" SPECIAL SYMBOL "\"
```

describes a substring that contains the sequence of characters:

- letter A,
- letter B,
- letter C,
- one digit,
- one of the letters a, b, c or d,
- one of the symbols [,] or ^,
- underscore character,
- letter a,
- letter b,
- letter c.

At the same time an arbitrary character set may be placed before this sequence. When running the client/server mode, it is possible that the request containing the LIKE operator is not executed. This situation is possible if the length of the string with the template exceeds a certain number depending on the DBMS applied.

8.5. Execution and working with queries in the 1C:Enterprise language

For the generation of queries, selection and processing of query results the language provides a special set of objects. Using these objects, a query is generated, query records are iterated, etc.
The type of the query result column is determined based on the types of expression values included in the column and always includes NULL type.

8.5.1. Working with queries

8.5.1.1. Operational data

The basic techniques for working with queries in the 1C:Enterprise language are most conveniently considered using examples. Here is a typical example of a query.

```
// Create a Query
Query = New Query("SELECT Products.Name Name,
  |Products.Parent.Name ParentName
  |FROM Catalog.Products Goods");
// Run the query and write the result to QueryResult variable.
QueryResult = Query.Execute();
// Retrieve the selection from the query result.
Selection = QueryResult.Select();
// While the selection contains records ...
While Selection.Next() Loop
  // ... display the fields from result to the message window.
  Goods = Selection.Name;
  Parent = Selection.ParentName;
  Message("Product: " + Goods + " Parent: " + Parent);
EndDo;
```

As we see from this example, query operations are performed using three main objects:

- Query - an object performing the query itself. This object is represented by the Query variable.
- QueryResult - is an object that contains the data retrieved when executing the query. This object is represented by the QueryResult variable.

- `QueryResultSelection` - is an object that can be used to iterate (i.e. look over) the records from the result. This object is represented by the `Selection` variable.

Let's study the `QueryResultSelection` object more carefully.

We will need to use the following query for this purpose:

```
SELECT
  SalesInvoiceContents.Products AS Products,
  SalesInvoiceContents.Quantity AS Quantity
FROM
  Document.SalesInvoice.Contents AS SalesInvoiceContents

ORDER BY
  SalesInvoiceContents.Products
TOTALS
  SUM(Quantity)
TO
  Products HIERARCHY
```

Result:

№	Nomenclature	Quantity
1	**Sanitaryware**	**104**
2	*Faucet*	*84*
3	Faucet	10
4	Faucet	8
5	Faucet	44
6	Faucet	22
7	*Mixer*	*20*
8	Mixer	5
9	Mixer	1
10	Mixer	14
11	**Furniture**	**134**
12	*Table*	*26*
13	Table	1
14	Table	15
15	Table	10
16	*Chair*	*108*
17	Chair	55
18	Chair	5
19	Chair	32
20	Chair	16

Fig. 193. Operational selection

In this table, we added the first column, which is not included in the query text, but will be used by us later to identify the record in the result. Totals records in the table are shown in italics, and totals values for hierarchical levels of the catalog are shown in bold.

8.5.1.2. Query result iteration methods

8.5.1.2.1. Linear result iteration

The linear method of result iteration - is the simplest. Linear method selects records in the same sequence as in the query result. In our example, these will be records 1, 2, 3, 4, 5, and so on, up to 20.

To retrieve a linear selection from the result, call the `Select()` method of the `QueryResult` object with no parameters or with the `QueryResultIteration.Linear` parameter.

Example:

```
SelectionMethod = QueryResultIteration.Linear;
Selection1 = QueryResult.Select(SelectionMethod);
// which is equivalent to the following line
Selection1 = QueryResult.Select();
```

8.5.1.2.2. Hierarchical result iteration

The next method for result iteration - is hierarchical. With this method, only records that are on the same level are iterated. To retrieve a hierarchical selection from the result, call the `Select()` method of the `QueryResult` object with the `QueryResultIteration.ByGroupsWithHierarchy` parameter.

Example:

```
SelectionMethod = QueryResultIteration.ByGroupsWithHierarchy;
Selection2 = QueryResult.Select(SelectionMethod);
```

In our example, only records 1 and 11 will be selected by the selection from the result with hierarchical iteration, as these are the only two records on the highest level. To illustrate this, let's assume that our result is a tree, where summary records are nodes, and detailed records - are leaves. Here is what we will get:

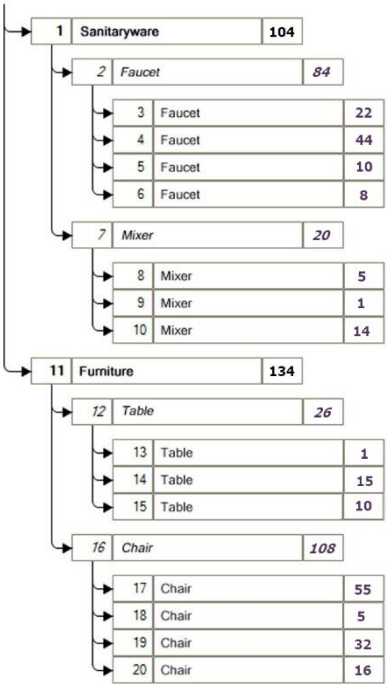


Fig. 194. Hierarchical query result iteration

From this figure, we can see that only records 1 and 11 are located at the first level of the tree and that only these records will be included in the first pass of hierarchical iteration.

The question is: how to obtain other records of the query result? To do this, it is possible to retrieve another selection from the `QueryResultSelection` object to be used for iteration over subordinate records of the current selection. In our example, when the `Selection2` object is placed on the record number 1, we will request hierarchical selection from it. Thus, we will get a selection that will return records with numbers 2 and 7. And if `Selection2` is placed on the record number 11, the resulting hierarchical selection will return records with numbers 12 and 16. This is how the hierarchical iteration for query results works.

Note, that it is possible to retrieve nested selections of any type from the selection. Thus, if we requested a linear selection from the `Selection2` object, placed in record 1, it would return records form 2 to 10. Let's show an example of the above method.

Example:

```
Procedure ExecuteQuery()
  // Create a query.
  Query = New Query;
  // Set a query text
  Query.Text = "SELECT
  |SalesInvoiceContents.Products AS Products,
  |SalesInvoiceContents.Quantity AS Quantity
  |FROM
  |Document.SalesInvoice.Contents AS SalesInvoiceContents
  |ORDER BY
  |SalesInvoiceContents.Products
  |TOTALS
  |SUM(Quantity)
  |BY
  |Products HIERARCHY";
  // Execute a query and write its result to a variable
  // QueryResult.
 QueryResult = Query.Execute();
  // Retrieve the selection from the query result.
  Selection =
QueryResult.Select(QueryResultIteration.ByGroupsWithHierarchy);
  OutputRecursively(Selection);
EndProcedure

Procedure OutputRecursively(Selection)
  // While the selection contains records ...
  While Selection.Next() Loop
    // ... display the fields from result to the message window
    Goods = Selection.Products;
    Quantity = Selection.Quantity;
    Message("Product: " + Goods + " Quantity: " + Quantity);
    // Continue subordinate records selection
    SelectionMethod = QueryResultIteration.ByGroupsWithHierarchy;
    OutputRecursively(Selection.Select(SelectionMethod,
Selection.Grouping()));
  EndDo;

EndProcedure
```

8.5.1.2.3. Result iteration by groups

The third and the final method of result iteration - is by groups. It is similar to hierarchical iteration, but there is a difference: records with hierarchical totals during iteration are treated as detailed records, rather than as totals records. To retrieve a selection by groups from the query result, call the `Select()` method

of the `QueryResult` object with the `QueryResultIteration.ByGroups` parameter.

Example:

```
SelectionMethod = QueryResultIteration.ByGroups;
Selection2 = QueryResult.Select(SelectionMethod);
// After going through it, will get entries with numbers:
// 1, 2, 7, 11, 12, 16.
```

Example:

```
Procedure ExecuteQuery()
  // Create a query.
  Query = New Query;
  // Set a query text
  Query.Text = "SELECT
  |SalesInvoiceContents.Products AS Products,
  |SalesInvoiceContents.Quantity AS Quantity
  |FROM
  |Document.SalesInvoice.Contents AS SalesInvoiceContents
  |ORDER BY
  |SalesInvoiceContents.Products
  |TOTALS
  |SUM(Quantity)
  |BY
  |Products HIERARCHY";
  // Execute a query and write its result to a variable
  // QueryResult.
  QueryResult = Query.Execute();
  // Retieve the selection from the query result
  Selection = QueryResult.Select(QueryResultIteration.ByGroups);

  // While the selection contains records ...
  While Selection.Next() Loop
    // ... display the fields from result to the message window
    Goods = Selection.Products;
    Quantity = Selection.Quantity;
    Message("Product: "+Goods+" Total for product: "+Quantity);
    OutputChildRecords(Selection.Select());
  EndDo;

EndProcedure

Procedure OutputChildRecords (Selection)

  // While the selection contains records ...
  While Selection.Next() Loop
    // ... display the fields from result to the message window
    Goods = Selection.Products;
    Quantity = Selection.Quantity;
    Message("Product: "+Goods+" Quantity: "+Quantity);
  EndDo;

EndProcedure
```

8.5.1.3. Operations with selection

The `QueryResultSelection` object is intended to iterate query result records. Selection is some object that contains a reference to the current result record and gives access to all the fields of the current record. Three methods are used for query record navigation:

- `Next()` - go to the next result record according to the selection iteration order. The selection is positioned on the first record at the first call. When all records are selected, this method will return the `False` value.
- `NextByFieldValue()` - get the next record with a value in the specified field that differs from the value in the same field of the current record.
- `FindNext()` - find a record with the specified values in certain fields.

8.5.1.3.1. Using NextByFieldValue() method

This method can be used to group results by field values.

Example:

```
SELECT
  Doc.Products,
  Doc.Reference.Counterparty AS Counterparty,
  Doc.Quantity
FROM
  Document.SalesInvoice.Contents AS Doc

ORDER BY
  Doc.Products.Name,
  Counterparty
```

Result:

№	Nomenclature	Contractor	Quantity
1	Faucet	"Giant" store	10.00
2	Faucet	"Master" store	8.00
3	Faucet	Mosgortorg	44.00
4	Faucet	"Furniture" store	22.00
5	Mixer	"Giant" store	5.00
6	Mixer	"Master" store	1.00
7	Mixer	Mosgortorg	14.00
8	Mixer	Mosgortorg	13.00
9	Table	"Giant" store	1.00
10	Table	"Master" store	15.00
11	Chair	Mosgortorg	10.00
12	Chair	"Giant" store	55.00
13	Chair	"Master" store	5.00
14	Chair	Mosgortorg	32.00
15	Chair	"Furniture" store	16.00

Fig. 195. Query result

We retrieve a linear selection from the query result and we will use the `NextByFieldValue()` method for iteration.

Example:

```
Selection = QueryResult.Select();
While Selection.NextByFieldValue("Products") Do
  // here we get records 1, 5, 9, 12
  While Selection.NextByFieldValue("Counterparty") Do
    // here we first get records 1, 2, 3, 4
    // then records 5, 6, 7
    // then records 9, 10, 11
    // then records 12, 13, 14, 15
  EndDo;
EndDo;
```

Note that we do not select record 8 in the internal cycle, since it has the same `Recipient` field value as the previous record.

Note that if we use the `Next()` method to get records by field value in the cycle, all the records with the same field value specified during the previous call of the `NextByFieldValue()` method will be selected.

Example:

```
Selection = QueryResult.Select();
While Selection.NextByFieldValue("Products") Do
  // here we get records 1, 5, 9, 12
  While Selection.Next() Loop
    // here we first get records 1, 2, 3, 4
    // then records 5, 6, 7, 8
    // then records 9, 10, 11
    // then records 12, 13, 14, 15
  EndDo;
EndDo;
```

8.5.1.3.2. Methods for defining the current record type

When a selection is placed in a record, we can use this selection to get the attributes of the record. The following methods are used to get record attributes:

- `Level()` - determines the record level in the query result.
- `RecordType()` - determines whether a record belongs to one of the following types:
  - group total,
  - total by hierarchy,
  - detailed record,
  - grand total.
- `Grouping()` - determines the name of the field used to calculate totals.

To illustrate how these methods operate, let's see what they will return for a query that was used as an example in the beginning of the chapter.

```
SELECT
  SalesInvoiceContents.Products AS Products,
  SalesInvoiceContents.Quantity AS Quantity
FROM
  Document.SalesInvoice.Contents AS SalesInvoiceContents

ORDER BY
  SalesInvoiceContents.Products
TOTALS
  SUM(Quantity)
TO
  Products HIERARCHY
```

Result:

№	Nomenclature	Quantity	Level	Record type	Grouping
1	**Sanitaryware**	**117**	**0**	**Totals by hierarchy**	**Nomenclature**
2	*Faucet*	*84*	*1*	*Total by grouping*	*Nomenclature*
3	Faucet	10	2	Detail record	
4	Faucet	8	2	Detail record	
5	Faucet	44	2	Detail record	
6	Faucet	22	2	Detail record	
7	*Mixer*	*33*	*1*	*Total by grouping*	*Nomenclature*
8	Mixer	5	2	Detail record	
9	Mixer	1	2	Detail record	
0	Mixer	14	2	Detail record	
10	Mixer	13	2	Detail record	
11	**Furniture**	**134**	**0**	**Totals by hierarchy**	**Nomenclature**
12	*Table*	*26*	*1*	*Total by grouping*	*Nomenclature*
13	Table	1	2	Detail record	
14	Table	15	2	Detail record	
15	Table	10	2	Detail record	
16	*Chair*	*108*	*1*	*Total by grouping*	*Nomenclature*
17	Chair	55	2	Detail record	
18	Chair	5	2	Detail record	
19	Chair	32	2	Detail record	
20	Chair	16	2	Detail record	

Fig. 196. Records hierarchy

8.5.2. Managing temporary tables

8.5.2.1. General information

1C:Enterprise query language allows to use temporary tables in queries. Using temporary tables will improve query performance and simplify reading the query text.

Managing temporary tables is ensured by two components:

- `TempTablesManager` 1C:Enterprise language object that stores temporary tables data;
- query language syntax that allows creating new and reusing current temporary tables.

8.5.2.2. Temporary tables manager

A temporary tables manager is intended to manage existence time of temporary tables created during the application operations.

Each application supports creation of multiple instances of temporary tables manager, where each of them stores its own set of temporary tables. Within a single temporary tables manager each temporary table should have a unique name that is also a temporary table ID.

NOTE. The names of temporary tables must meet the requirements to the names of the 1C:Enterprise language variables.

Temporary tables manager instance may be created using the `New` constructor.

Example:

```
TempTablesManager = New TempTablesManager;
```

When a transaction is rolled back, all the temporary tables that were created during the transaction are deleted from the temporary table manager, and it is impossible to access them.
Each temporary table created in a given manager instance exists until this temporary tables manager instance exists. Deleting a temporary tables manager instance also deletes all the temporary tables contained in it.
Force a temporary tables manager to close with the `Close()` method. All the tables created in it will be deleted. Further operating this temporary tables manager instance is not possible.

8.5.2.3. Creating temporary tables

Temporary tables are created using `Query` object of 1C:Enterprise language.
A query is linked to a temporary tables manager using the `TempTablesManager` property of the query, which specifies the manager instance where the temporary tables must be created.

Example:

```
TempTablesManager = New TempTablesManager;
Query = New Query;
Query.TempTablesManager = TempTablesManager;
```

A temporary table can be created from database data or external data source (for example, a value table).
In order to create a temporary table based on database data, set a temporary tables manager to the `Request` object, and then perform a database query using the `PLACE` keyword, followed by the name of the temporary table to be created. The `PLACE` keyword must be specified after the query selection list.

Example:

```
SELECT
  Products.Code,
  Products.Name
PLACE TempTable
FROM
  Catalog.Products AS Products
```

The values of the `Code` and `Name` attributes for all the elements of the `Products` directory will be placed in the temporary table. At the same time, from the point of view of the 1C:Enterprise language, the result of executing such a query will be the `QueryResult` object, the selection from which will contain one line with one column `Quantity`. This number will indicate how many records were placed in the created temporary table.

If the temporary table manager is not set, has been closed or a table with the specified name already exists in the set temporary tables manager, - an error will be displayed.

Use the `AUTORECORDNUMBER()` query language function to add a column in a created temporary table that will be filled with a unique number.

When creating temporary tables not based on an external source using the `FOR CHANGING` structure is possible. This is necessary in cases where it is required to retrieve data into a temporary table and simultaneously block it from reading by other transactions.

Example:

```
SELECT
  SalesInvoice.Reference,
  SalesInvoice.Number,
  SalesInvoice.Date
PLACE TempTable
FROM
  Document.SalesInvoice AS SalesInvoice
WHERE
  SalesInvoice.Reference IN(&Documents)

FOR UPDATE
```

To create an index for a temporary table, add the `INDEX BY` keyword to the query and specify the fields for building the index after this keyword.

Example:

```
SELECT
  Products.Code AS Code,
  Products.Name
PLACE TempTable
FROM
  Catalog.Products AS Products

INDEX BY
  Code
```

Fields used for indexing must be included in the selection list.

If a table of values is used as a source, then this table of values must explicitly indicate the types of values contained in the columns.

In order to create a temporary table and block the data of the tables based on which the temporary table is created, use the `FOR CHANGING` structure.

Example:

```
SELECT
  SalesInvoice.Reference,
  SalesInvoice.Number,
  SalesInvoice.Date
PLACE TempTable
FROM
  Document.SalesInvoice AS SalesInvoice
WHERE
  SalesInvoice.Reference IN(&Documents)

FOR UPDATE
```

In order to create a temporary table based on an external source, the parameter name where the external source will be placed must be specified in the source list of the query text. Then create a temporary table as usual.

The following external data sources are available:

- value table,
- tabular section
- a query result,

An example of creating a temporary table based on the external data source is given below:

```
SELECT
  Source.Code,
  Source.Name
PLACE TempTable
FROM
  &ExternalSource AS Source
```

In this example, the contents of the `Code` and `Name` columns from an external source, for example, the value table that was passed as an `ExternalSource` parameter, will be placed in the `TempTable` temporary table.

IMPORTANT! If a temporary table is created based on an external source, the query cannot use mergers and connections, as well as fields that are attributes of the table fields based on which the temporary table is created.

8.5.2.4. Using temporary tables

To use existing temporary tables, set a temporary table manager to the `Request` object in a query, after which it becomes possible to access temporary tables contained in this temporary table manager by name similar to regular query tables.

8.5.2.5. Dropping temporary tables

To drop a temporary table from the temporary table manager, use the `DROP` keyword of the query language followed by the name of the table being dropped, for example:

```
DROP TempTable
```

If the table being dropped does not exist, an error will be generated.

8.5.2.6. Debugging a query with temporary tables

When debugging complex queries containing temporary tables, it becomes necessary to analyze the content of temporary tables. To do this, the `Query.ExecuteBatchWithIntermediateData()` method or the temporary tables manager can be used.

NOTE. The considered query has no application meaning, but serves solely to demonstrate the managing of temporary tables in the process of query debugging.

Consider an example of such use:

```
Query = New Query;
Query.TempTablesManager = New TempTablesManager;
Query.Text = "SELECT
   | SalesTurnover.Goods,
   | SalesTurnover.TurnoverQuantity
   |PLACE Sales
   |FROM
   | AccumulationRegister.Sales.Turnover AS SalesTurnover
   |;
   |
   |////////////////////////////////
   |SELECT
   | ProductReservesBalance.Goods,
   | ProductReservesBalance.BalanceQuantity
   |PLACE Reserves
   |FROM
   | AccumulationRegister.ProductReserves.Balance AS
ProductReservesBalance
   |;
   |
   |////////////////////////////////
   |SELECT
   | Products.Name,
   | Sales.TurnoverQuantity,
   | Balance.BalanceQuantity
   |FROM
   | Catalog.Goods AS Goods
   |  INTERNAL CONNECTION Balance AS Balance
   |  BY Goods.Reference = Balance.Goods
   |  INTERNAL CONNECTION Sales AS Sales
   |  BY Goods.Reference = Sales.Goods";
Result = Query.Execute();
SalesVT = Query.TempTablesManager.Tables["Sales"].GetData().Export();
BalanceVT = Query.TempTablesManager.Tables[1].GetData().Export();
```

As a result of the query, the `Query.TempTablesManager.Tables` property will contain a collection of temporary tables containing the results of the batch query. Using the `Table` property, it is possible to access each temporary table, and this can be done both by name and by index of the temporary table. For example, the value table `SalesVT` has been retrieved as a result of accessing the `Sales` temporary table by name (`Tables ["Sales"]`). At the same time, the generation of a value table `BalanceVT` is performed by accessing by index (`Tables[1]`).

8.5.3. Working with batch queries

The 1C:Enterprise platform allows to operate the query batches. In a batch request, query texts are separated by “;”. Queries are executed sequentially, while temporary tables that were created during the execution of a query will exist until the end of the execution of the entire query batch or until the operation in the batch query that destroys this temporary table, for example:

```
Query = New Query;
Query.Text =
  "SELECT
  |ProductsAccountingBalanceAndTurnovers.Products,
  |ProductsAccountingBalanceAndTurnovers.ReceiptQuantity,
  |ProductsAccountingBalanceAndTurnovers.SalesQuantity,
  |ProductsAccountingBalanceAndTurnovers.EndBalanceQuantity
  |PLACE ProductsAccounting
  |FROM
  |AccumulationRegister.ProductsAccounting.BalanceAndTurnovers(, ,
Auto, , ) AS ProductsAccountingBalanceAndTurnovers
  |;
  |
  |SELECT
  |ProductsAccounting.Products,
  |ProductsAccounting.SalesQuantity,
  |ProductsAccounting.EndBalanceQuantity
  |FROM
  |ProductsAccounting AS ProductsAccounting
  |";

Result = Query.Execute();
```

The first query creates a temporary table, the data from which is used in the second query.

If the `Query` object executing a batch query is set up with a temporary tables manager, temporary tables that have not been dropped as part of the batch query will be saved to the set manager. It is possible to use and dropping temporary tables that existed in the set temporary tables manager at the time the batch was executed in the text of a batch query.

In addition to the `Execute()` method, which sequentially executes all the queries in the batch and returns the result of the last query of the batch, the 1C:Enterprise platform provides another method - `ExecuteBatch()`. This method sequentially executes all the queries and returns an array of results for each query from the batch in the sequence of queries placed in the text of the batch. The result of the query for the dropping of a temporary table is the `Undefined` value , which is also placed in an array of results.

If it is necessary to debug a batch query, it is possible to use the `ExecuteBatchWithIntermediateData()` method. In this case, the result of the operation is similar to the `ExecuteBatch()` method with the difference that the temporary tables generated in the query batch will not be represented by the number of records in the corresponding temporary table (as when calling the `ExecuteBatch()` method), but by the contents of such temporary tables.

Thus, it is possible to modify the above example as follows:

```
Query = New Query;
Query.Text =
  "SELECT
  |ProductsAccountingBalanceAndTurnovers.Products,
  |ProductsAccountingBalanceAndTurnovers.ReceiptQuantity,
  |ProductsAccountingBalanceAndTurnovers.SalesQuantity,
  |ProductsAccountingBalanceAndTurnovers.EndBalanceQuantity
  |PLACE ProductsAccounting
  |FROM
  |AccumulationRegister.ProductsAccounting.BalanceAndTurnovers(, ,
Auto, , ) AS ProductsAccountingBalanceAndTurnovers
  |;
  |
  |SELECT
  |ProductsAccounting.Products,
  |ProductsAccounting.SalesQuantity,
  |ProductsAccounting.EndBalanceQuantity
  |FROM
  |ProductsAccounting AS ProductsAccounting
  |";

BatchResult = Query.ExecuteBatchWithIntermediateData();
TempTable = BatchResult[0].Export();
Result = BatchResult[1].Export();
```

In this case, the value table will be available in the `TempTable` variable, which contains the data of the `ProductsAccounting` temporary table, and the `Result` variable will have an accessible temporary table with the result of the last batch query execution (the actual result of the batch).

8.5.4. Software operations with queries

The 1C:Enterprise provides the possibility to create or modify a query from the 1C:Enterprise language. For this, there is a special `QuerySchema` object. Using the `QueryBatch` property, the developer gets access to all queries that are currently being edited using the `QuerySchema` object. The `SetQueryText()` method is used to populate the `QuerySchema` object based on the passed query text, and the `GetQueryText()` method is used for the inverse operation.
As an example of using the object model of working with queries, the program code needed to generate the following query is given:

```
SELECT
  ProductReserveBalance.Goods AS Goods,
  ProductReserveBalance.Warehouse AS Warehouse,
  ProductReserveBalance.BalanceQuantity AS BalanceQuantity,
  CASE
    WHEN ProductReserveBalance.BalanceQuantity > 0
      THEN "Sufficient"
    ELSE "Insufficient"
  END AS Status
FROM
  AccumulationRegister.ProductReserves.Balance AS
ProductReservesBalance
WHERE
  ProductReserveBalance.Warehouse = &Warehouse
ORDER BY
  Goods,
  Warehouse,
  BalanceQuantity
TOTALS
  SUM(BalanceQuantity)
TO
  GRAND TOTALS,
  Goods,
  Warehouse
```

Code in 1C:Enterprise language:

```
Constructor = New QuerySchema;
Batch1 = Constructor.QueryBatch[0];
Operator1 = Batch1.Operators[0];
// add a source to a query
RegisterTable =
Operator1.Sources.Add("AccumulationRegister.ProductReserve.Balance",
"ProductReserveBalance");
// add fields to a query
FieldGoods =
Operator1.SelectedFields.Add("ProductReserveBalance.Goods", 0);
AvailableFieldWarehouse =
RegisterTable.Source.AvailableFIelds.Find("Warehouse");
FieldWarehouse = Operator1.SelectedFields.Add(AvailableFieldWarehouse,
1);
FieldsQuantity = Operator1.SelectedFields.Add("BalanceQuantity", 2);
SelectionExpression = New QuerySchemaExpression("SELECTION
       |WHEN ProductReserveBalance.BalanceQuantity > 0
       | THEN "Sufficient"
       | ELSE "Insufficient"
       |END");
ComboBox = Operator1.SelectedFields.Add(String(SelectionExpression),
3);
// set aliases
Result = Batch1.Columns.Find(FieldQuantity);
Result.Alias = "BalanceQuantity";
Result = Batch1.Columns[1];
Result.Alias = "Warehouse";
Result = Batch1.Columns.Find(SelectionExpression);
Result.Alias = "Status";
// add a condition
Operator1.Filter.Add(String(FieldWarehouse) + " = &Warehouse");
// add ordering
Batch1.Order.Add(FieldGoods);
Batch1.Order.Add("Warehouse");
Batch1.Order.Add("BalanceQuantity");
// set totals control points
Batch1.Overall = True;
Batch1.TotalCalculationFields.Add(FieldGoods);
Batch1.TotalCalculationFields.Add("Warehouse");
// set aggregate functions for totals
ColumnQuantity = Batch1.Columns.Find(FieldQuantity);
Batch1.TotalExpressions.Add("Sum(" + ColumnQuantity.Alias +")");

Text = Constructor.GetQueryText();
```

The `QuerySchema` object supports the query extension for the data composition system. It is defined by the property `QuerySceme.DataCompositionMode`. If this property is set in the value `True` then the properties `TotalExpressions`, `TotalCalculationFields`, and `Overall` of the object `QuerySchemaSelectQuery` become accessible. If the query text is set with the help of the method `QueryScheme.SetQueryText()`, the property

`QueryScheme.DataCompositionMode` is set in the value `True`, and the query text contains `TOTALS` expression -the exception will be displayed.
The client application allows opening the query editor during operation. The object `QueryWizard` is intended for it. The wizard is opened by calling the method `Show()` where the callback processor shall be transferred. The callback will be performed after the query editing completion. The query editor can be opened in two modes: in the query edit mode and in the query edit for the data composition mode. This behavior is managed with the help of the property `QueryWizard.DataCompositionMode`. The query text being in process of editing (or edited) is accessible with the help of the property `QueryWizard.Text`.

Chapter 9. Data operations

9.1. Object locks function

9.1.1. General information

When working with object data (catalogs, documents, accounts, etc.), the 1C:Enterprise provides two types of object locks: - **pessimistic** and **optimistic**. These allow to make holistic changes to objects when several users working simultaneously.

9.1.2. Pessimistic lock

The function of database objects pessimistic lock is intended to prohibit changing of object data by other sessions or this session until the lock is released (automatically or using the methods of the 1C:Enterprise language).
Basically, the pessimistic lock function is used by the 1C:Enterprise to lock objects edited in the form. At the same time, the developer has the possibility to apply this function using the tools of the 1C:Enterprise language.
The 1C:Enterprise applies a pessimistic lock function using applied object form extensions. When the user begins modifying an object in a form, the form extension sets a pessimistic lock. When the user who edited the object closes the object form, the form extension will remove the pessimistic lock. If during an object editing, another user (or the same user but from another session) tries to start editing the same object the system will notify about it. Several solutions can be offered depending on the situation:

- In file mode:
  - The collaboration system is unavailable: the detail information on the user who has blocked the object is shown.
  - The collaboration system is available: the information on the user who has blocked the object is shown and the editing is enabled. In the case of editing, the changes made by the other user will be lost.
    If the object is blocked by another user -it is possible to write a message to such user using the collaboration system.
- Client/server mode:

- If the object is blocked by the same user in another session- it is possible to start editing with the loss of the changes made in another session.
- If the object is in process of editing by another user it is possible to start editing with the loss of the changes made by another user. In this case, it is possible to write a message to another user if the collaboration system is accessible.

If it is necessary to provide the same behavior in the non-standard object form as in the standard object form, it is possible to use the `LockFormDataForEdit()` form method to set a pessimistic lock and the `UnlockFormDataForEdit()` form method to remove the lock.

The developer, in order to enable pessimistic lock, can use the `LockDataForEdit()` global context method. There are two options for setting pessimistic lock:

- With the specification of the form indicator, -in this case, the locking is set for the form life period. When the form is closed it will be unlocked at once if the usual connection is used and after a time if a slow connection is used. In all the following cases the unlock will take place at once:
  - End of the session in which the form is opened.
  - 1 minute has elapsed after the form modification attribute was cleared.
  - During the form display, other locks have been set on behalf of this form (with interactive editing or using the `LockFormDataForEdit()` method).
  - The form on behalf of which the background jobs were run (search in the dynamic list or report generation) is closed.
  - The `UnlockDataForEdit()` global context method with the same form ID specified for setting the lock is used.
- Without specifying the form ID - in this case, the lock to be set is not bound to any form. The lock will be automatically removed at the end of the session, when control is returned from the server or at the end of the transaction (if the lock was set in the transaction). A lock can also be removed using the `UnlockDataForEdit()` global context method without specifying a form ID.

However, it should be noted that the fact of setting a lock itself does not preclude modifying or deleting an object in the database. Therefore, in order to ensure that it is impossible to modify a locked object, an object change operation in another session must also be preceded by an attempt to lock this object. Locking a locked database object causes an exception that can be handled by the `Try ... Exception ... EndTry`.

```
&OnServer
Function ModifyingExample()

  GoodsReference = Catalogs.Goods.FindByCode("000000001");

  Try

    LockDataForEdit(GoodsReference);

    // Object data may be modified
    // ...
    GoodsObject = GoodsReference.GetObject();
    GoodsObject.Name = "New name";
    GoodsObject.Write();

    Return True;

  Exception

    // Object data cannot be modified
    Message = New UserMessage;
    Message.Text = "Object data are already locked";
    Message.Message();

    Return False;

  EndTry;

EndFunction
```

It should be noted, that attempts to lock the same object with the indication of the form ID and without specifying the ID are incompatible with each other.

```
GoodsReference = Catalogs.Products.FindByCode(1);

LockDataForEdit(GoodsReference);

Try

  LockDataForEdit(GoodsReference, , FormID);

Exception

  // exception due to locks incompatibility

EndTry;
```

To remove the pessimistic lock, the developer can use the `UnlockDataForEdit()` global context method.

9.1.3. Pessimistic lock and transactions

Object lock operations only affect the execution of other object lock operations and do not affect data operations or the transaction flow process.
Locking a locked database object throws an exception that can be handled and does not necessarily causes the transaction to roll back. If an exception is thrown in a transaction during the execution of the `LockDataForEdit()` method, it can be handled by the `Try ... Exception ... EndTry` structure and does not require the transaction rollback.
Object locks set during a transaction are removed at the end of the transaction, if the lock was set without specifying a form ID.

9.1.4. Optimistic lock

An optimistic lock prevents the object from being written to the database if, after reading an object, it has been modified in the database.
Optimistic locking is a basically check that is performed before an object is written to the database.
When a 1C:Enterprise language object reads data from a database, among other things, it reads the version of the object stored in the database.
If the data of the object in the database have been changed (for example, by another user) before the user starts editing the data (before setting the pessimistic lock), the version number of the object stored in the database will also change. When a user attempts to write this object, the check will be performed whether the version of the object in memory matches the version of the object stored in the database. Since the versions differ, a warning will be displayed that the version of the object has changed or the object was deleted, that is, an optimistic lock will activate.
Optimistic lock ensures that if the user changes an object, its changes will not “overwrite” the changes made by other sessions or other software objects of the same session.

9.2. Transaction function

9.2.1. General information

Regardless of the operation mode chosen (file or client/server), the 1C:Enterprise provides working with information stored in the database using the transaction function.

Transaction -is indivisible in terms of the impact on the database, sequence of data manipulation operations. It is executed on the “all or nothing” principle and transfers a database from one holistic state to another holistic state. If, for any reason, one of the actions of the transaction is not feasible or any system malfunction has occurred, the database returns to the state it was in before the transaction began (the transaction is rolled back).
Transactions can be used both by the 1C:Enterprise itself and by the developer when writing modules.
The 1C:Enterprise defines an implicit transaction when performing any actions related to the modification of information stored in the database. For example, all the event handlers located in object modules and record sets that are associated with modifying database data are called during a transaction. At the same time, reading database objects (accessing object properties “separated by dot”, executing a `.GetObject()` method, opening an object form or a set of records), as a rule, is performed outside the transaction in order to improve the 1C:Enterprise operation parallelism. Within such reading, the 1C:Enterprise ensures that the state of the read object in memory will be exactly as it was in the database at the time of the reading start. However, to ensure read integrity, the 1C:Enterprise can open implicit transactions, the occurrence conditions of which are described below.
Along with this, the developer can use the transactions operations explicitly. `BeginTransaction()`, `CommitTransaction()` and `RollbackTransaction()` global context methods are used for this.

9.2.2. Using explicit transaction call

The `BeginTransaction()` method allows to open a transaction. After that, all the changes to the database data performed by subsequent operators can be either fully accepted or fully rejected.
The `CommitTransaction()` method is used to accept all the changes made. In order to undo all changes made in an open transaction, use the `RollbackTransaction()` method. If the number of the `BeginTransaction()` method calls exceeds the number of `CommitTransaction()` or `RollbackTransaction()` method calls, the 1C:Enterprise will execute an implicit `RollbackTransaction()` method call in the following cases:

- at the end of the 1C:Enterprise language execution (event handler, external connection, automation server);
- when transferring control from server to client.

If the number of `CommitTransaction()` or `RollbackTransaction()` method calls exceeds the number of `BeginTransaction()` method calls, then if an extra call of the `CommitTransaction()` or `RollbackTransaction()` method is performed, an exception will be thrown.

Thus, the general transaction operation scheme may look as follows:

```
Try

  BeginTransaction();

  // Sequence of operators
  …

  CommitTransaction();

Exception

  RollbackTransaction();

EndTry;
```

When using such a scheme, it should be note that not all the errors that occur when working with a database are handled in the same way.

In general, all the database errors can be divided into two categories:

- non-recoverable,
- recoverable.

Non-recoverable errors - are the errors, which prevent the normal functioning of the 1C:Enterprise in case of occurrence, for example, data may be corrupted. If a non-recoverable error occurs, the running of 1C:Enterprise is terminated in any case.

If a non-recoverable error occurred during the execution of a transaction, then all the changes made within this transaction are discarded by 1C:Enterprise.

Recoverable errors - are the errors that do not cause serious violations in the 1C:Enterprise operation. In case of a recoverable error, the further operation of 1C:Enterprise can be continued. In this case, operation, which caused the error, stops, and an exception is thrown, which can be caught and handled by the `Try ... Exception ... EndTry` structure.

If the recoverable error occurred during the execution of transaction, 1C:Enterprise does not automatically cancel the transaction, giving the developer the chance to independently handle the situation.

Depending on the nature of the error that occurred, various scenarios for handling such situation are possible.

If the error that occurred is not related to the database, then the transaction and module operation can be continued. If the developer considers it necessary, they

can cancel the transaction or, conversely, continue the transaction if the error occurred does not violate the atomic nature of the transaction.
If the exception was caused by a database error, 1C:Enterprise records the occurrence of an error in this transaction, and further continuation of the transaction or its committing becomes impossible. The only database operation that the developer can perform in this situation - is to cancel the transaction. After that, they may attempt to perform this transaction again.
For example, a code fragment that implements this approach when writing some data to a database may be as follows.

```
// Write attempts completion flag
Written = False;

// Write attempts are performed in a loop.
While Not Written Do

  Try

    BeginTransaction();
    Data.Write();
    CommitTransaction();

    // In case of committing a transaction stop write attempts
    Written = True;

  Exception

    // If unsuccessful, cancel the current transaction and
    // start the next attempt with a new transaction
    RollbackTransaction();

  EndTry;

EndDo;
```

9.2.3. Nested transaction call

It is possible to call the `BeginTransaction()`, `CommitTransaction()` and `RollbackTransaction()` procedures within the ongoing transaction. For example, the following call pattern may be used:

```
BeginTransaction();
…

  // Nested transaction call
  BeginTransaction();
  …
  CommitTransaction();
…

  // Nested transaction call
  BeginTransaction();
  …
  CommitTransaction();
…
CommitTransaction();
```

However, such call does not mean the beginning of a new transaction within an already running one.

IMPORTANT! 1C:Enterprise does not support nested transactions.

This means that only the top-level transaction is always in effect. All the transactions that are called inside an already open transaction are actually related to the same transaction, and do not form a nested transaction. Thus, the cancellation of changes performed in a nested transaction will not ultimately cause the cancellation of changes to the nested transaction itself, but the cancellation of all the changes to the top-level transaction. At the same time, committing changes made in a nested transaction is ignored.

9.2.4. The impact of transactions on the software objects operation

In general, the software objects used by 1C:Enterprise are absolutely “transparent” for database transactions. In other words, database transactions can be called when executing various methods of software objects, however, for example, the actions performed by the database when a transaction is rolled back generally do not affect the corresponding software objects in any way.
From this it follows that when canceling database transactions, the developer (if necessary) must independently ensure an adequate change in the data of the corresponding software objects. This can be done by re-reading all the object data or by changing some attributes of the software object (if, for example, it is necessary to display it in the interface).
There are exceptions to this rule. Due to the significant application specificity of the 1C:Enterprise software objects, in some cases, rollback of changes made in

the database can still affect the property values of the corresponding software objects. This happens in the following cases:

- when a transaction is canceled, the document posting flag restores the value it had before the transaction start;
- if the object was created and recorded in a transaction, then if the transaction is rolled back, the reference value is cleared;
- if the object was created outside the transaction and the code/number generated automatically was used when writing it in the transaction, then the code/number is cleared when the transaction is canceled.

9.3. Managed lock function

9.3.1. General information

Ideally, transactions in any DBMS must provide isolation of changes performed in the database. In other words, multiple transactions that perform data modification must not interfere with each other.

The easiest way to solve this problem is the sequential execution of transactions. The next transaction is executed after the previous one has ended.

However, in a real situation, with multi-user work, this approach leads to a sharp decrease in 1C:Enterprise performance. Therefore, in practice, the features are used that allow to perform multiple transactions simultaneously.

To make simultaneous execution of transactions possible, several levels of transaction isolation are used. At the lowest isolation level, transactions can strongly influence each other. At the highest level, they are completely isolated.

Thus, large overhead and 1C:Enterprise slowdown is the price to pay for the for higher isolation of transactions.

The possibility to isolate one transactions from others is usually implemented through locks set for the data they use. Depending on the isolation level, different types of locks are set on different database objects for different times.

From the 1C:Enterprise point of view, data operations can be performed in one of two modes:

- in transaction,
- outside transaction.

The **outside transaction** data operation mode allows **data reading** operations only. This mode is introduced in order to ensure maximum speed and parallel data reading. Therefore, any data read operation performed outside transaction is considered **irresponsible**. This means that such read operation may return obsolete data or even uncommitted changes made by another transaction, i.e., reading is performed without looking at data locks set by the other transactions.

The in transaction data operation mode allows any data read and modify operations.
The following rules must be observed:

- Data reading must be reproducible, i.e., any subsequent data read operation with the same selection condition must return the same result.
- The read result must contain the most up-to-date data. This means that no other transaction can change the data read in this transaction until the transaction has been completed.
- The read result must not contain uncommitted changes to the database data.

9.3.2. Managed locks

The 1C:Enterprise allows to use two modes of working with database within a transaction: automatic lock mode and managed lock mode.
The main difference between these modes is as follows. The automatic lock mode does not require the developer to take any action to manage locks in a transaction in order to ensure the above rules for working with data in a transaction are met. These rules are provided by the 1C:Enterprise platform by using certain levels of transaction isolation in a particular DBMS. This mode of operation is the easiest for the developer, but in some cases (for example, with intensive simultaneous operations of a large number of users), the transaction isolation level in the DBMS cannot provide sufficient parallelism, which is manifested in the form of a large number of lock conflicts during users operations.
When operating in the managed lock mode, the 1C:Enterprise uses a much lower level of transaction isolation in the DBMS, which can significantly increase the parallelism of the application user operations. However, in contrast to the automatic lock mode, this transactions isolation level can no longer ensure that all the rules for working with data in a transaction are met (in particular, the reproducibility of data reading in a transaction is not ensured). Therefore, when using the managed mode, the developer is required to independently manage the locks that are set in a transaction.
In summary, the differences when using automatic lock mode and the managed lock mode are shown in the table below:

	Lock type	**Transaction isolation level**
Automatic locks		
File DB	Tables	Serializable
MS SQL Server	Records	Repetable Read or Serializable
IBM DB2	Records	Repetable Read or Serializable
PostgreSQL	Tables	Serializable

	Lock type	Transaction isolation level
Oracle Database	Tables	Serializable
Managed locks		
File DB	Tables	Serializable
MS SQL Server 2000	Records	Read Commited
MS SQL Server 2005 or higher	Records	Read Commited Snapshot
IBM DB2	Records	Read Commited
PostgreSQL	Records	Read Commited
Oracle Database	Records	Read Commited

NOTE. It is recommended that the application development is carried out in a managed lock mode in order to maximize the use of the DBMS capabilities. Using an application in automatic mode is used for compatibility with the previous versions of application and is not recommended for actual operation.

The 1C:Enterprise behavior during irresponsible reading (for example, when dynamic lists are running, reports are executed) also differs in different lock modes. When working with objects that do not have tabular sections, the differences between irresponsible reading in different lock modes (automatic or managed one) are shown in the following table:

	Reading outside transaction
Automatic locks	
File DB	"Dirty" read
MS SQL Server	"Dirty" read
IBM DB2	"Dirty" read
PostgreSQL	Consistent read
Oracle Database	Consistent read
Managed locks	
File DB	"Dirty" read
MS SQL Server 2000	"Dirty" read
MS SQL Server 2005 or higher	Consistent read
IBM DB2 below version 9.7	"Dirty" read
IBM DB2 version 9.7 and higher	Consistent read
PostgreSQL	Consistent read
Oracle Database	Consistent read

Irresponsible reading of objects with tabular sections and record sets is always consistent. For this, the platform can define an implicit transaction and set implicit managed locks:

	Implicit transaction
Automatic locks	
File DB	Yes
MS SQL Server	Yes
IBM DB2	Yes
PostgreSQL	No
Oracle Database	No
Managed locks	
File DB	Yes
MS SQL Server 2000	Yes
MS SQL Server 2005 or higher	Yes - in version 8.2.16 and earlier compatibility mode No - in version 8.2.17 and earlier compatibility mode
IBM DB2	Yes
PostgreSQL	No
Oracle Database	No

When reading an object with tabular sections without an implicit definition of a transaction, the object version is read twice in the managed lock mode (the `DataVersion` object field) at the beginning and at the end of the reading. If the versions differ, a transaction lock error is displayed.
Information register record sets are always read in an implicit transaction, while in the managed lock mode, a shared lock is set by the recorder value, and for an independent information register - by the values of the main filter.

9.3.3. Setting the lock mode in the configuration

The configuration has a Data lock management mode property. Each configuration applied object also has a Data lock management mode property.
The data lock management mode for the entire configuration if general can be set to Automatic, Managed (set by default for the new configuration), and Automatic and managed. The Automatic or Managed values mean that this lock mode may be used for all the configuration objects regardless of values set for each individual object. The Automatic and managed value means that the particular configuration object will use the mode specified in its Data lock management mode property: Automatic or Managed.

It should be noted that the data locking management mode specified for the metadata object is set for those transactions that are initiated by the 1C:Enterprise when working with the data of this object (for example, when modifying the object data).
If, for example, an object write operation is performed in a transaction initiated by the developer (`BeginTransaction()` method), then the data lock management mode will be determined by the value of the `LockMode` parameter of the `BeginTransaction()` method, and not by the value of the Data lock management mode metadata object property .
By default, the `LockMode` parameter is set to `DataLockControlMode.Automatic`, therefore, in order to use the managed lock mode in an explicit transaction, it is necessary to set this parameter value to `DataLockControlMode.Managed`.

9.3.4. Operations with managed locks using 1C:Enterprise language tools

To manage locks in transaction, the 1C:Enterprise language `DataLock` object is used. An instance of this object can be created using the constructor and allows to describe the necessary lock spaces and modes. To set all the created locks, use the `Lock()` method of the `DataLock` object. If this method is run in transaction (explicit or implicit one), locks are set and will be removed automatically at the end of the transaction. If the `Lock()` method is executed outside a transaction or in an automatic lock management mode in a transaction, an exception will be thrown.
The `DataLock` object is a collection of data lock elements, each of which describes locks of one lock space. The lock spaces are defined in the 1C:Enterprise platform and correspond to the structure of the applied configuration objects. For each lock space in the platform, field names are defined, the values of which can be analyzed during setting of certain locks.
The following lock space names and lock space field names are available:

Lock space name	Lock field name
`Catalog.<name>`	`Reference`; `<field name>`
`Document.<name>`	`Reference`; `<field name>`

Lock space name	Lock space field name
ExchangePlan.<name>	Reference; <field name>
ChartOfAccounts.<name>	Reference; <field name>
BusinessProcess.<name>	Reference; <field name>
Task.<name>	Reference; <field name>
ChartOfCalculationTypes.<name>	Reference; <field name>
ChartOfCharacteristicTypes.<name>	Reference; <field name>
InformationRegister.<name>.RecordSet - for the information register subordinate to the recorder only	Recorder
InformationRegister.<name>	Period - if any; <dimension name>
AccumulationRegister.<name>.RecordSet	Recorder
AccumulationRegister.<name>	Period; <dimension name>
AccountingRegister.<name>.RecordSet	Recorder
AccountingRegister.<name>	Period; <dimension name>; <record type> - systen enumeration valu AccountingRecordType Account; Extra dimension<N>; <extra dimensio type>.
CalculationRegister.<name>.RecordSet	Recorder
CalculationRegister.<name>	RegistrationPeriod; ActionPeriod; <dimension name>
Recalculation.<name>.RecordSet	RecalculationObject

Lock space name	Lock field name
Sequence.<name>.RecordSet	Recorder
Sequence.<name>	<dimension name>
Constant.<name>	
ExternalDataSource.<source name>.Table.<table name>	Reference; <field name>

Where <field name> - name of field, which can be used to set managed lock by. The list of available fields (which a managed lock can be set by) is set in the Data Lock Fields property for the following objects:

- Catalogs
- Documents
- Charts of characteristic types
- Charts of accounts
- Charts of calculation types
- Exchange plans
- Business processes
- tasks;
- external data source tables.

It is forbidden to set locks by attributes (and general attributes) of the following types: string of unlimited length, value storage, value type of chart of characteristic types, composite type including one of the above types. The ban includes the following objects: catalogs, documents, charts of characteristic types, charts of calculation types, charts of accounts, business processes, tasks, exchange plans. It is also forbidden to set locks by the Predefined and PredefinedDataName fields in catalogs, charts of calculation types, charts of calculation types, charts of accounts.

When setting a managed transaction lock on the AccountingRegister.<Name> space, if the value of the extra dimension is specified by reference to the characteristic, then this characteristic in the chart of accounts must be one of the extra dimensions of this account. If the corresponding extra dimension is not specified for the account, an error will be generated.

For each lock space, an arbitrary number of conditions can be specified for the fields, which will determine the records to be locked. Conditions are set to whether the field value is equal to the specified value or whether the field value is within the specified range. Conditions can be defined in two ways:

- by explicitly specifying the field name and value (SetValue() method of the DataLockItem object);

- by specifying a data source containing the required values (the `DataSource` property of the `DataLockItem` object).

When explicitly specifying a field value, the field name and value are passed to the parameters of the `SetValue()` method;

```
Lock = New DataLock;
LockElement = Lock.Add("Catalog.Store");
LockElement.SetValue("Code", 100);
LockElement.Mode = DataLockMode.Exclusive;
Lock.Lock();

Lock = New DataLock;
LockElement = Lock.Add("AccumulationRegister.GoodsInWarehouses");
LockElement.SetValue("Quality", Catalogs.Quality.FindByCode("1"));
```

It should be noted that the same register entry can be locked twice: the first time by locking the record itself, and the second time - by locking the set which this record is included in.

When deleting or modifying an object for which locking by fields is available, one lock is performed, which includes locking on the `Reference` field and by all the fields specified in the Data lock fields property. When a change is made, the object is locked both by the “old” field values (which were effective before the writing start) and by the “new” values (which are in placed the written object).

The value of the 1C:Enterprise language `Range` object, which is created using the constructor and allows to specify the top and bottom limits of the range (the boundaries of the range are included in the range), can also be passed as a value.

When using a data source, the value of the `DataSource` property is set, and then, using the `UseFromDataSource()` method, the lock area fields are set to match the data source fields:

```
Lock = New DataLock;
LockElement = Lock. Add("AccumulationRegister.GoodsInWarehouses");
LockElement.DataSource = DocumentObject.ReusablePackage;
LockElement.UseFromDataSource("Products", "Products");
LockElement.UseFromDataSource("Warehouse", "Warehouse");
```

The data source can be:

- a query result,
- tabular section
- record set,
- value table.

Accordingly, when matching the fields, the names of the source fields will be:

- query result column names,
- tabular section attribute names,

- dimension names,
- value table column names.

The `Range` object can also be the value of a data source field.
For each lock element, one of two lock modes can be specified:

- shared,
- exclusive.

The lock mode is set using the `Mode` property of the `DataLockItem` object.
The managed lock compatibility table is as follows:

	Shared	Exclusive
Shared	+	-
Exclusive	-	-

Shared lock mode implies that locked data cannot be changed by another transaction until the end of the current transaction.
The exclusive lock mode implies that the locked data cannot be changed by another transaction until the end of the current transaction, and also cannot be read by another transaction that sets a shared lock on this data.
Speaking about the compatibility of actions performed in a transaction, it should be noted that in addition to explicit managed locks set by the user, 1C:Enterprise also implicitly controls exclusive locks when writing data in a transaction. When reading object data from the database (retrieving the object and accessing the reference), the transactional lock of the object is not performed. If a lock is necessary, then it must be set by means of the language before accessing the object. Thus, the compatibility table of actions performed in a transaction when using the managed lock mode is as follows.

Lock	**Mode**	**NL**		**SL**		**EL**	
		R	W	R	W	R	W
NL	R	+	+	+	+	+	+
	W	+	-	-	-	-	-
SL	R	+	-	+	-	-	-
	W	+	-	-	-	-	-
EL	R	+	-	-	-	-	-
	W	+	-	-	-	-	-

Where:

- R - read,
- W - write,
- NL - no lock,
- SL - shared lock,

- EL - exclusive lock.

9.3.5. Features of "Automatic and managed" mode

Automatic and managed lock control mode is intended to solve problems that arise when using individual configuration objects in parallel.
For example, there is a large configuration, which is difficult to transfer into a managed lock mode at a single time, however there are one or two documents that are a “stumbling block” when a large number of users use them simultaneously. In this case, the entire configuration can be transferred to the Automatic and managed mode, and then the document itself and the configuration objects affected when it is written - can be transferred to the Managed mode. This will allow to set up operations with this document in the managed lock mode, while the bulk of the configuration will continue to operate in the automatic lock management mode.
When using the Automatic and managed lock management mode, two features must be considered:

- Regardless of the mode specified for this transaction, 1C:Enterprise will set the appropriate managed locks.
- The lock control mode is determined by the transaction of the “highest” level. In other words, if by the moment of the transaction start another transaction was started, then the started transaction can be executed only in the mode that is set for the transaction that is already running.

Consider these features in more detail.
The first feature is that even if the automatic lock management mode is used for a transaction, 1C:Enterprise will additionally set the corresponding managed locks when writing data in this transaction. It follows from this that transactions that are executed in the managed lock mode may conflict with transactions that are executed in the automatic lock management mode.
The second feature is that the lock management mode specified for the metadata object in the configuration or explicitly indicated at the start of a transaction (as a parameter of the `BeginTransaction()` method) is only the “desired” mode. The actual lock management mode in which the transaction will be executed depends on whether this transaction start call is the first one, or by this time another transaction has already started in this session of 1C:Enterprise.
For example, if it is necessary to manage locks when writing sets of register records, while posting a document, then the managed lock mode must be set for both register and the document, since the recording of the sets of register records will be performed in the transaction opened when the document is written.

Four different combinations of lock management modes are possible in a transaction, which are presented in the table below:

Existing transaction mode	**Started transaction mode**	**Result**
Automatic	Automatic	The started transaction will be executed in automatic mode.
Automatic	Managed	The started transaction will be executed in automatic mode.
Managed	Automatic	An exception will be thrown
Managed	Managed	The started transaction will be executed in managed mode.

9.3.6. Features of using a large number of managed locks

It should be noted that if more than 100,000 locks are superimposed on one space, lock escalation may occur. The entire space is blocked in this case. When using independent separators, escalation occurs for one set of separator values:

- The entire space is locked only within the limits of the separator values;
- Sessions with different values of separators are not affected by escalation.

When using the managed locks, it should be noted that 1C:Enterprise can take over the set locks. A lock, where the values of not all the dimensions of the lock spaces are indicated, will take over a lock, where the values are indicated for a larger number of dimensions of the lock spaces, provided that the values for the coincident dimensions match the lock spaces. Both locks set from the 1C:Enterprise language using the `DataLock.Lock()` method and locks set by the platform features can be taken over.

Thus, for example, if the accumulation register has the `Warehouse` and `Products` dimensions, then the lock, which only has the `Warehouse` dimension, will take over any locks that have both dimensions (`Warehouse` and `Products`).

If a transaction with a large number of managed locks is used, which differ, for example, by the value of one dimension, it is recommended to set one managed lock for which the value of the different dimension will not be set instead of a large number of locks.
If a conflict with already set locks occurs during a lock escalation, the escalation is not performed, but an attempt is made to set the requested lock. In this case, the situation is possible when there will be more than 100,000 locks per space in 1C:Enterprise.

9.3.7. Modifying configurations during the transition to the managed lock mode

9.3.7.1. General information

In the transition to partial or full operation in the managed lock mode, the application needs some improvement. The improvement implies that it is necessary to identify portions of code that require the presence of managed locks, and then set the desired mode of managed data lock in these portions of the code. This can be done in two ways:

- using `DataLock` object;
- using `LockForUpdate` property of the record sets in accumulation registers and accounting registers. The required managed lock will be automatically set by the platform in the event that a set of registry records is written for which this property is `True`.

9.3.7.2. Identifying the portions of code that require improvement

Managed locks must be set in those portions of the code that satisfy one of the following conditions:

- Data is read, which must be changed in the future.
- Some consistent data set (contained in several objects) is read, and the consistency of the read data needs to be maintained. In this case, the data from this set must be locked either at a time before reading, or as individual elements are being read. If a data element is read once, not connected logically with the other data elements and will not be changed in the same transaction, then it may not be locked.
- It is important to ensure that the read data remains unchanged until the end of the transaction. For example, some of the data read in a transaction can be read again, and it is important to ensure that they are unchanged.

A typical example of such operations may be a query for product balances in the document posting module.

9.3.7.3. Selecting a managed lock mode

For the operations described above, the locking mode must be set based on the following considerations:

- **Exclusive** locking mode must be set for the data that must be changed within the same transaction. This will prevent lock conflicts.
- **Shared** lock mode must be set for data that is only read and is not intended to change, but is still required to be locked from modification.

9.3.7.4. Configuration improvement examples

Below there is an example of setting managed locks in the record set module of the `GoodsInWarehouses` accumulation register.

First of all, a managed lock must be created:

```
Lock = New DataLock;
```

Then the text of the query generated in the module must be analyzed and the necessary locks must be created.

Exclusive lock of "GoodsInWarehouses" register

During the execution of the record set module, a query is generated containing the following fragment:

```
// …
LEFT CONNECTION
AccumulationRegister.GoodsInWarehouses.Balance(,
  Products IN (SELECT DISTINCT
    Products
    FROM
    Document.GoodsServicesSale.Goods
    WHERE
    Reference = &DocumentRef))
// …
```

In this case, the following locks must be set:

```
LockGoodsInWarehouses1 = Lock.
     Add("AccumulationRegister.GoodsInWarehouses");

LockGoodsInWarehouses1.Mode = DataLockMode.Exclusive;

LockGoodsInWarehouses1.DataSource = DocumentObject.Goods;

LockGoodsInWarehouses1.UseFromDataSource("Products", "Products");
LockGoodsInWarehouses1.UseFromDataSource("ProductsCharacteristic",
"ProductsCharacteristic");
LockGoodsInWarehouses1.UseFromDataSource("Warehouse", "Warehouse");
```

Shared lock of the “GoodsReservedInWarehouses” register

During the execution of the module, a query is generated containing the following fragment:

```
// …
LEFT CONNECTION
AccumulationRegister.GoodsReservedInWarehouses.Balance(,
  Products IN (SELECT DISTINCT
    Products
    FROM
    Document.GoodsServicesSale.Goods
    WHERE
    Reference = &DocumentRef))
// …
```

In this case, the following locks must be set:

```
LockGoodsReservedInWarehouses1 =
Lock.Add("AccumulationRegister.GoodsReservedInWarehouses");

LockGoodsReservedInWarehouses1.Mode = DataLockMode.Shared;

LockGoodsReservedInWarehouses1.DataSource = DocumentObject.Goods;

LockGoodsReservedInWarehouses1.UseFromDataSource("Products",
"Products");
LockGoodsReservedInWarehouses1.UseFromDataSource(
"ProductsCharacteristic", "ProductsCharacteristic");
LockGoodsReservedInWarehouses1.UseFromDataSource("Warehouse",
"Warehouse");
```

Shared lock of the “GoodsForTransferFromWarehouses” register

During the execution of the module, a query is generated containing the following fragment:

```
// …
LEFT CONNECTION
AccumulationRegister.GoodsForTransferFromWarehouses.Balance(,
  Products IN (SELECT DISTINCT
    Products
    FROM
    Document.GoodsServicesSale.Goods
    WHERE
    Reference = &DocumentRef))
// …
```

In this case, the following locks must be set:

```
LockGoodsForTransferFromWarehouses1 =
Lock.Add("AccumulationRegister.GoodsForTransferFromWarehouses");

LockGoodsForTransferFromWarehouses1.Mode = DataLockMode.Shared;

LockGoodsForTransferFromWarehouses1.DataSource = DocumentObject.Goods;

LockGoodsForTransferFromWarehouses1.UseFromDataSource("Products",
"Products");
LockGoodsForTransferFromWarehouses1.UseFromDataSource("ProductsCharact
eristic", "ProductsCharacteristic");
LockGoodsForTransferFromWarehouses1.UseFromDataSource("Warehouse",
"Warehouse");
```

Locks by "ReusablePackage" tabular section. Exclusive lock of "GoodsInWarehouses" register

During the execution of the module, a query is generated containing the following fragment:

```
// …
LEFT CONNECTION
AccumulationRegister.GoodsInWarehouses.Balance(,
  Products IN (SELECT DISTINCT
    Document.GoodsServicesSale.ReusablePackage.Products
    FROM
    Document.GoodsServicesSale.ReusablePackage
    WHERE
    Document.GoodsServicesSale.ReusablePackage.Reference =
      &DocumentRef)
    AND Quality = &New)
// …
```

In this case, the following locks must be set:

```
LockGoodsInWarehouses2 =
Lock.Add("AccumulationRegister.GoodsInWarehouses");

LockGoodsInWarehouses2.Mode = DataLockMode.Exclusive;

LockGoodsInWarehouses2.DataSource = DocumentObject.ReusablePackage;

LockGoodsInWarehouses2.UseFromDataSource("Products", "Products");
LockGoodsInWarehouses2.UseFromDataSource("Warehouse", "Warehouse");
LockGoodsInWarehouses2.SetValue("Quality",
Catalogs.Quality.FindByCode("1"));
```

Locks by "ReusablePackage" tabular section. Shared lock of the “GoodsReservedInWarehouses” register

During the execution of the module, a query is generated containing the following fragment:

```
// …
LEFT CONNECTION
AccumulationRegister.GoodsReservedInWarehouses.Balance(,
  Products IN (SELECT DISTINCT
    Document.GoodsServicesSale.ReusablePackage.Products
    FROM
    Document.GoodsServicesSale.ReusablePackage
    WHERE
    Document.GoodsServicesSale.ReusablePackage.Reference =
      &DocumentRef)) AS Reserves
// …
```

In this case, the following locks must be set:

```
LockGoodsReservedInWarehouses2 =
Lock.Add("AccumulationRegister.GoodsReservedInWarehouses");

LockGoodsReservedInWarehouses2.Mode = DataLockMode.Shared;

LockGoodsReservedInWarehouses2.DataSource =
DocumentObject.ReusablePackage;
LockGoodsReservedInWarehouses2.UseFromDataSource("Products",
"Products");
LockGoodsReservedInWarehouses2.UseFromDataSource("Warehouse",
"Warehouse");
```

Locks by "ReusablePackage" tabular section. Shared lock of the “GoodsForTransferFromWarehouses” register

During the execution of the module, a query is generated containing the following fragment:

```
// …
LEFT CONNECTION
AccumulationRegister.GoodsForTransferFromWarehouses.Balance(,
  Products IN (SELECT DISTINCT
    Document.GoodsServicesSale.ReusablePackage.Products
    FROM
    Document.GoodsServicesSale.ReusablePackage
    WHERE
    Document.GoodsServicesSale.ReusablePackage.Reference =
      &DocumentRef)) AS GoodsForTransfer
// …
```

In this case, the following locks must be set:

```
LockGoodsForTransferFromWarehouses2 =
Lock.Add("AccumulationRegister.GoodsForTransferFromWarehouses");

LockGoodsForTransferFromWarehouses2.Mode = DataLockMode.Shared;

LockGoodsForTransferFromWarehouses2.DataSource =
DocumentObject.ReusablePackage;

LockGoodsForTransferFromWarehouses2.UseFromDataSource("Products",
"Products");
LockGoodsForTransferFromWarehouses2.UseFromDataSource("Warehouse",
"Warehouse");
```

After all the necessary locks have been created, the listed data should be locked:

```
Lock.Lock();

// ... followed by the old text of the module
```

9.4. Exclusive mode

Exclusive mode - is a specific database access mode, using which it is possible to perform operations with the database using just one session. The exclusive mode can be used in cases when it is necessary to make substantial, consistent changes with the infobase that are not required to be made as part of a transaction. At the same time it is desirable to exclude the effect of other sessions on the results of the change. Setting the exclusive mode requires some methods of the 1C:Enterprise language (for example, the `DeleteObjects()` method), as well as the operation of the Designer.

To set or remove the exclusive mode, it is required to use the `SetExclusiveMode()` global context method. For checking the setting of the exclusive mode us the `ExclusiveMode()` method. It is not possible to change the status of the exclusive mode if the transaction is active (explicit or implicit one).

A session can set exclusive mode if there are no other sessions with this infobase. It is prohibited to create new sessions with this infobase except for launching one background job in the exclusive mode. If a background job is started from the session that set the exclusive mode, then the started background job will “take away” the exclusive mode from the parent session. During the background job, the parent session is not able to change the data in the infobase. The exclusive mode will “return” to the parent session after the running background job is completed.

In the case of using the exclusive mode, managed locks are not set (attempts to set them are ignored).

Chapter 10. Data composition system

The data composition system is intended to create 1C:Enterprise reports based on their declarative description. Using declarative report description allows to implement the following features:

- creating a report without programming;
- capability to create different versions of the report;
- capability to set various versions of user settings;
- use of automatically generated forms for report viewing and customization;
- splitting the execution of the report into stages;
- execution of individual stages of report generation on different computers;
- independent use of individual parts of the data composition system;
- programmatic control of the report execution process;
- report structure settings;
- combining multiple tables in a report;
- creation of nested reports, etc.

Capabilities of the data composition system are used:

- when running dynamic lists;
- when preparing data for further processing (for example, in data processors).

10.1. General data composition information

The data composition system is a set of elements, each of which corresponds to a certain stage of the report execution. Thus, the entire process of executing a report in the data composition system is reduced to a sequential transition from one element to another, eventually reaching the final report.

Each element of the data composition system has its own declarative description, the capability of programmatic access and the feature to serialize to/from XML. Such approach allows to flexibly manage the various stages of the report execution.

The main elements of the data composition system are shown in fig. 197.

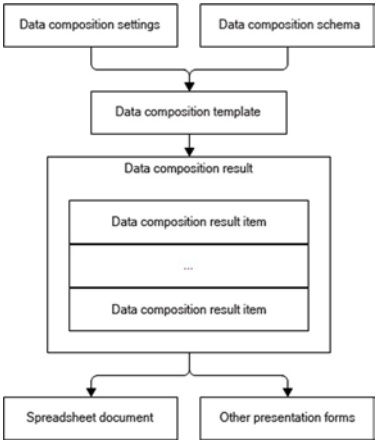


Fig. 197. Main elements of data composition system

Data composition scheme - describes the essence of the data that is provided to the report (where to get the data from and how to manage the data composition). It is a base which various reports can be generated from. It may contain:

- query text with data composition system instructions;
- description of several data sets;
- description of the available fields;
- description of the links between multiple data sets;
- description of data retrieval parameters;
- description of field and grouping templates, etc.

Data composition settings - describe everything that a developer or user can customize in some established data composition scheme. These may contain:

- filter;
- ordering;
- conditional appearance;
- report structure (components of the future report);
- data retireval parameters;
- data output parameters, etc.

Data composition template - is a ready description of how the report must be generated. It combines the composition and composition settings. It actually represents the result of applying specific settings to the composition

scheme and is a ready job for the composition processor to generate a report of the desired structure with regard to specific settings.

Data composition result element - data composition result is represented by a set of data composition result elements. As an independent logical entity, the result of the data composition does not exist, only its elements do. The elements of the data composition result can be output to a spreadsheet document for presentation to the end user or to other types of documents.

The data composition process consists of several stages, which are shown in fig. 198.

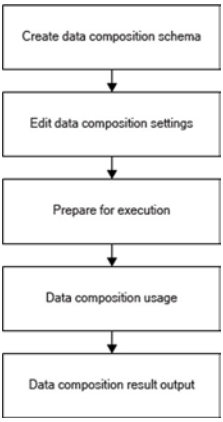


Fig. 198. Stages of data composition process

Creating a data composition scheme can be performed:

- visually, using the data composition wizard;
- visually, using any editor that allows editing XML text;
- programmatically, using the objects of 1C:Enterprise language.

Editing data composition settings - 1C:Enterprise provides a number of 1C:Enterprise language objects and table field extensions to edit composition settings.

Preparing for execution - the process of generating the data composion template. In this process, queries are generated that are necessary to retrieve the data specified in the settings, and the report area templates are generated.

Execution of data composition - is the process of retrieving, aggregating, processing the data.

Outputting the data composition result - the obtained result of the composition can be output to a document that will be shown to the user. The report can be output in various formats.

The diagram in fig. 199 in a generalized form represents the data composition system objects used at various stages of report creation.

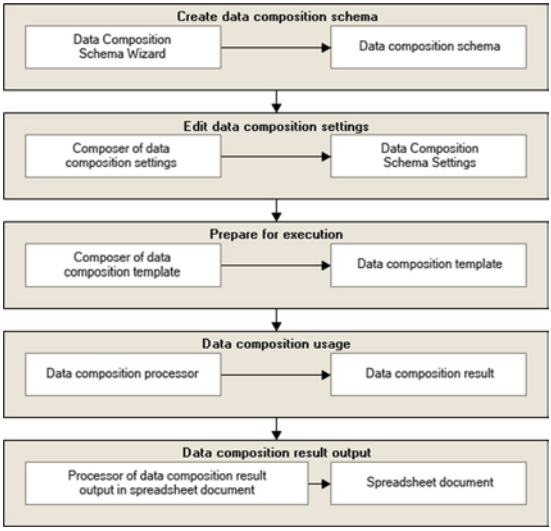


Fig. 199. Data composition system objects

Data composition scheme wizard - can be used to create a data composition scheme.
Data composition settings compiler - can be used to edit the data composition system settings.
Data composition template builder - is used to prepare for execution.
Data composition processor - performs the data composition execution.
Processor for outputting the data composition result to a spreadsheet document - outputs the data composition result elements to the spreadsheet document.
A distinctive feature of the report, obtained using the data composition system, is that it can have a complex structure that includes a different combination of the following elements:

- grouping,
- table,
- chart,
- nested report,

Thus, a report obtained using the data composition system is not a table, but a complex hierarchical structure where the listed elements are included.

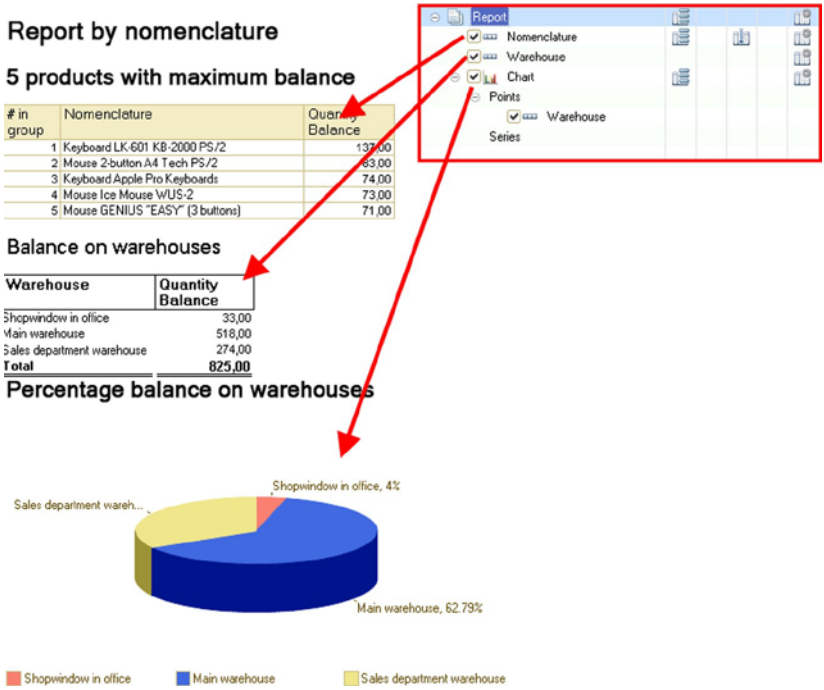


Fig. 200. Report structure

This figure shows an example of a report containing a chart and grouping by product groups at the first level of the hierarchy. And inside the grouping by product groups there is a table containing the turnover of the products belonging to the specified product group.

10.2. General data composition system objects

10.2.1. "Use" property

Many objects, mainly data composition settings subsystems, have a boolean type `Use` property. This property allows to disable some of the functionality

without physically removing it. The default value of this property is - `True`, except for the individualy described cases.

10.2.2. Data composition system field

This is an object representing the field data path. The field is implemented as a separate type in order to eliminate ambiguity in the properties of some objects, which can take both strings and fields as values. It has a constructor with the `String` parameter that describes the data path; it has no properties and methods. If the data path contains an ID with spaces or special characters, then such identifiers must be enclosed in brackets.

10.2.3. Data composition system parameters

The parameter function was implemented for uniform use and editing of collections of some values, the element composition and type of which are predetermined and cannot be changed. The parameter function consists of two parts:

- Available parameters - determine the composition of the collection and the allowed types of its elements. The parameter is similar to the data composition system field.
- Parameter values.

10.3. Data composition schema

The data composition scheme is represented by the `DataCompositionSchema` 1C:Enterprise language object and consists of many other nested objects. The data composition scheme has an XML representation; thus, it can be created by any tools allowing to generate XML, as well as used by any tools that can read XML.
The data composition scheme is used to provide information about the available settings, as well as when creating the data composition template, that is, when the data composition is executed.
For visual editing the data composition scheme, the data composition scheme wizard is used.
Loading the data composition scheme from XML can be performed using standard 1C:Enterprise language tools.

```
ReadXML = New ReadXML;
ReadXML.SetString(Controls.DataCompositionSchemeText.GetText());
DCS = XDTOSerializer.ReadXML(ReadXML,
Type("DataCompositionSchema"));
```

All the expressions described in the data composition scheme are written in the expression language of the data composition system.

10.3.1. Data composition scheme components

Each data composition scheme contains a set of objects describing a particular section. Consider these components.

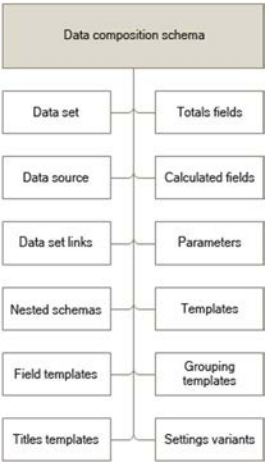


Fig. 201. Data composition scheme components

10.3.1.1. Data sources

A data composition schema can include multiple data sources.

A data source is a source to retrieve data from. The 1C:Enterprise infobase serves as a data source.

Data sources are described in the `DataSources` scheme property, which contains a collection of values consisting of the `DataCompositionSchemaDataSource` elements.

10.3.1.2. Data Sets

Data sets in the data composition scheme contain information about which fields can be retrieved from this set, which fields of the data set can be used in filtering, etc.

Multiple data sets are allowed in the data composition scheme.

Data sets are described in the `DataSets` scheme property. The property contains a collection of values, which may include the following elements:

- **query** (`DataCompositionSchemaDataSetQuery`) - data retrieval is described using query language;
- **object** (`DataCompositionSchemaDataSetObject`) - describes the name of the external data set from which the data will be retrieved;
- **merging** (`DataCompositionSchemaDataSetUnion`) - describes data sets - merging components.

All the three data sets contain a number of common properties:

- Name - name of the data set used to access this data set from other data composition scheme objects. The data set names must be unique within a single data composition scheme.
- Fields - description of fields, available for the data set.

In addition, -the query and object data sets contain the `DataSource` property - name of the data source from which the data will be retrieved. It must contain the name of one data source that is present in the data composition scheme.

Data set - query contains `Query` property - text of the query, using which the data from the data source will be retrieved. It is permitted to use a special query language extension - an extension for the data composition system in the query text.

Data set - merging contains `Elements` property, containing the list of data sets, included in merging.

10.3.1.2.1. Data set - query

It contains the usual query to the data located in the 1C:Enterprise infobase.

Data set - query may contain a batch query. The resulting query will be a batch query. In this case, the composition of the fields that will be placed in the temporary table is automatically determined by the fields used in other queries. If no field is needed from the temporary table, the temporary table will not be placed in the resulting query. The filter used in the data composition settings is applied to all queries of the batch.

The result of the last query in the batch will be considered the contents of the data set.

It is possible to access one data source only (including external data sources) within one data set.

10.3.1.2.2. Data set - object

Data set - object is used to output information from some 1C:Enterprise language object to a report: value table, query result, current document, etc.
This data “source” is described in the composition scheme, then it is filled programmatically, for example, by pressing a button, and passed as an external data set to the composition processor.

10.3.1.2.3. Data set - merging

Data set - merging contains `Elements` property, describing the data sets to be merged.
Note that the field values of the data set - merging will be retrieved from the fields of the nested data sets at their data path. Thus, in the example above, the external data set will have a `ExpenseAmount` field. Data for it will be retrieved from nested data sets, which have the data path - `ExpenseAmount`.

10.3.1.2.4. Data composition scheme data set field

A data set can contain the descriptions of fields available to the data set.
Data set fields are described in the Data set fields property, which contains a collection of values consisting of the `DataCompositionSchemaDataSetField` elements.

10.3.1.3. Data set links

The data sets present in the data composition scheme may be linked to each other.
Data set links are described in the `DataSetLinks` property of the data composition scheme, which contains a collection of values consisting of the `DataCompositionSchemaDataSetLink` elements.

10.3.1.4. Calculated fields

It is possible to describe the fields in the data composition scheme that will be calculated by some expressions using the fields of the data sets. These fields can be used in the settings in the same way as the data set fields, except for the following: use of calculated fields attributes is not supported, hierarchical grouping by calculated fields is not supported.

Calculated fields are described in the `DataCompositionSchemaCalculatedFields` property of the data composition scheme, which contains a collection of values consisting of the `DataCompositionSchemaCalculatedField` elements.

10.3.1.5. Resource fields

It is possible to describe the resource fields in the data composition scheme, the values of which will be calculated for group records. This is performed by describing the total field.

Total fields are described in the `TotalFields` property of the data composition scheme, which contains a collection of values consisting of the `DataCompositionSchemaTotalField` elements. When calculating resources, each record of the data set for which the resource is calculated is involved in the calculation of the resource only once.

10.3.1.6. Parameters

The data composition scheme contains a description of the data parameters.

Data parameters are described in the Data composition scheme parameters property, which contains a collection of values consisting of the `DataCompositionSchemaParameter` elements.

It is possible to specify the parameter mandatory use flag and the prohibition of an empty parameter. For example, using a combination of these flags, it is possible to implement a parameter, without specifying which report generation is not performed.

10.3.1.7. Nested schemas

The data composition scheme may contain descriptions of nested data composition schemes.

Nested data composition schemes are described in the `NestedDataCompositionSchemas` property of the data composition scheme, which contains a collection of values consisting of the `NestedDataCompositionSchema` elements.

10.3.1.8. Templates

It is possible to describe the templates in the data composition scheme that will be used to output a field or grouping. When specifying a template for a field or grouping, the template name described in this property is indicated.

Templates are described in the Data composition scheme templates property. The property contains a collection of values consisting of `DataCompositionSchemaTemplateDescription` elements.

10.3.1.9. Field templates

For each field in the data composition scheme, the template name used to output this field to the composition result can be specified.

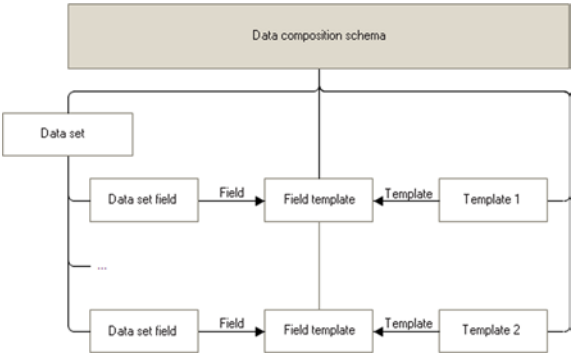


Fig. 202. Field templates

The link between the field and the template is described using the `DataCompositionSchemaFieldTemplate` object. The collection of these objects is contained in the `FieldTemplates` property of the `DataCompositionSchema` object.

10.3.1.10. Grouping templates

For each grouping in the data composition scheme, it is possible to specify the template name that will be used when outputting this grouping.
The link between the grouping and the template is described using the `DataCompositionSchemaGroupTemplate` object. The collection of these objects is contained in the `GroupTemplates` property of the `DataCompositionSchema` object.

10.3.1.11. Grouping header templates

For each grouping, the grouping header templates may be also described.
The link between the grouping header and the template is described using the `DataCompositionSchemaGroupTemplate` object (see previous section). The collection of these objects is contained in the `GroupHeaderTemplates` property of the `DataCompositionSchema` object.

10.3.1.12. Default settings

Each report variant that is specified in the data composition scheme contains default data composition settings that can be specified by the developer. In this case, the default option will be the one that ranks first in the list of data composition system settings options (Chart by period option in fig. 203). The default settings will be applied when the report is first opened, when More - Standard settings command is selected (All actions - Standard settings), and can also be used to programmatically set default settings.

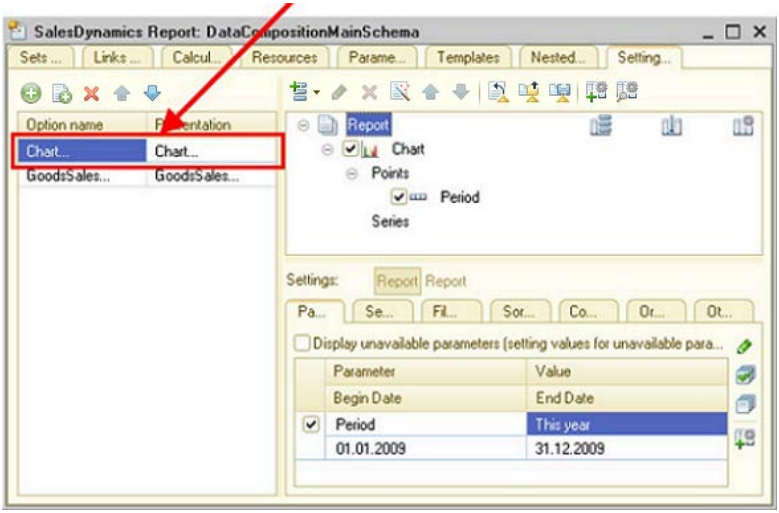


Fig. 203. Default option

10.3.2. Using multiple data sets

Data composition system allows multiple data sets to be used in a single composition.

In order to use several data sets in one composition, it is necessary to add descriptions of the data sets to be used to the scheme and specify the links between the data sets.

Consider the following example.

Write three data sets: `PriceList`, `Balance` and `Sales`.

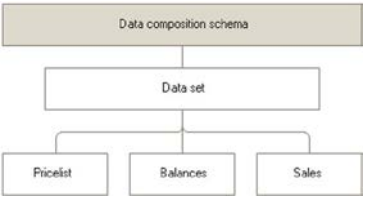


Fig. 204. Example of multiple data sets

Describe the links between data sets. Create a link between `PriceList` and `Balance` data sets and a link between `PriceList` and `Sales` data sets (see fig. 205).

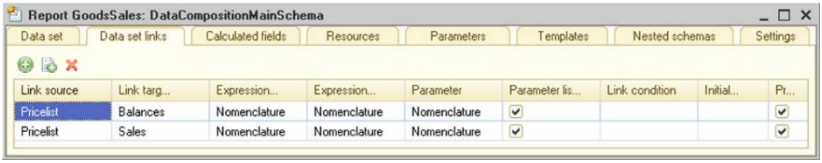


Fig. 205. Data set links

Additionally describe the resources (see fig. 206).

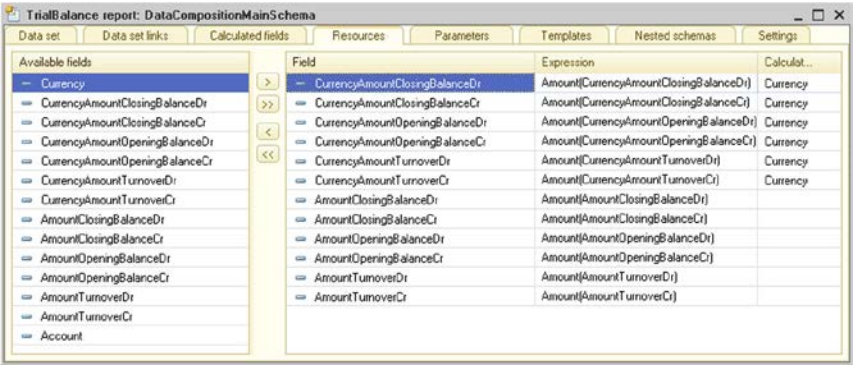


Fig. 206. Data composition scheme resources

If the data composition system describes the links between two data sets, then the data to whick the link is going will be considered **dependent**. The data set from which the link is going will be considered the **parent** one in relation to the data set dependent on it.

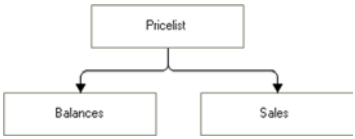


Fig. 207. Dependent data sets

In the examples above, the dependent data sets will be `Balance` and `Sales` data sets. And `Pricelist` -will be the parent data set in relation to both of these sets.

There is no indication of the link type in the data composition scheme. All the connections are considered left external connections. That is, the record of the parent data set will be used in the composition even if no records were found for it in the dependent set.

It is possible to specify the link type in the data composition template. The type of link is generated by the composition compiler depending on the applied global filters. If a global filter is applied to a data set field that is dependent, the links generated in the data composition template from this data set to all its parent sets (up to the beginning of the data set hierarchy) will be of `Internal` type. This means that the records of the parent data set will be included in the composition only if the records are found in the dependent data sets.

For example, if a user applies a global filter on the `Warehouse` field, the `PriceList` and `Balance` data sets will be linked usin internal connection.

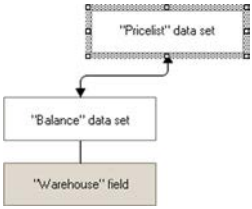


Fig. 208. Internal link

If the data set depends on a certain data set and the link contains a feature of using the parameter list, the data from the dependent data set will be obtained

in batches of 1000 records. If the use of the parameter list is not permitted in the link, the records will be retrieved one by one.
If both dependent and parent data sets contain a field with the same name, this field will be retrieved from the parent data set. In the examples above, the `Products` field will always be retrieved from the `PriceList` data set.

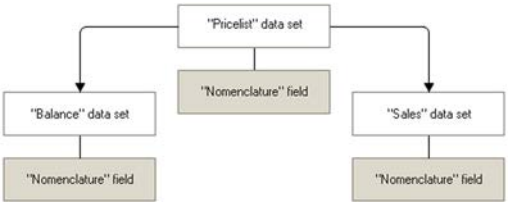


Fig. 209. Common field

Unlinked data sets cannot contain fields with the same name unless they have a common parent, which also contains this field. Thus, the `Balance` and `Turnovers` data sets can contain the `Products` field, but both of them can not contain the `Counterparty` field.
The fields from the source data set (parent set) are used in the source expression of the link description; the receiver data set (dependent data set) fields are used in the receiver link expression. Thus, when describing the link between the `PriceList` and `Balance` data sets the `Products` expression in the `SourceExpression` property uses the the field of `PriceList` data set, and the `Products` expression in `DestinationExpression` property uses the field of `Balance` data set.
Fields from unlinked data sets cannot be used in the same grouping. At the same time data sets that have common parent data sets are not considered to be linked. An exception is made for totals fields that can be used in any grouping. In the above example it is not possible to use `Warehouse` and `Counterparty` in the same grouping. However, we will be able to use the `BalanceQuantity` and `TurnoverAmount` fields, since these are resources.
The data of the dependent data set cannot be retrieved without retrieving the data of the parent data set. That is, upon data retrieval from a dependent set, data from the parent set (and all the parents of the parent) will also be automatically retrieved. In our example when retrieving the `Balance` data set the data from `PriceList` will be also retrieved.
If a grouping uses datasets from multiple data sets, the iteration by the last dependent data set will be performed when the composition is executed.

Thus, if the grouping uses the fields from `PriceList` and `Balance` sets, iteration will be performed by `Balance` set.
If no fields from the linked data set are enabled in the settings, the data set will not be included in the data composition template.

10.3.3. Query language extension for data composition system

The query language extension for the data composition system is implemented using special syntax instructions enclosed in braces and placed directly in the query text.

10.3.3.1. Syntax elements of the data composition system query language extension

SELECT

Description:

This expression describes the fields that the user can select for output. After this keyword, the aliases of the fields from the main query selection list that will be available for customization are set separated by comma.
After the field alias there may be a combination of characters "`.*`", which means the capability of using child fields from this field.
For example, the `Products.*` record denotes the capability of using child fields of the `Products` field (for example, the `Products.Code` field).
`SELECT` element may be present in the first merger query only.

Example:

```
{SELECT Product, Warehouse}
```

WHERE

Description:

Describes the fields which can be used for filtering by the user. This expression uses table fields. Using aliases in the selection list fields is not permitted.
Each part of the merging may contain its own `WHERE` element.
If the parameter values are not specified, the `WHERE` expression is not included in the resulting query.

Example:

```
{WHERE Product.*, Warehouse }
{WHERE Document.Date >= &StartDate, Document.Date <= &EndDate}
```

CHARACTERISTICS

Description:

To ensure operations with characteristics, the syntax for describing characteristics has been introduced in the query language extension for the data composition system.

The example (see below) describes the characteristics for the fields of `Reference` type to the `Products` catalog.

The characteristics describe the following properties:

- `TYPE` - the name of the type for which the characteristics are described.
- `CHARACTERISTICTYPES` - table name or query for a list of characteristics. It is permitted to use the `ALLOWED` keyword in the query. In this case, the list of characteristics will contain only those characteristics that are allowed for the user by the imposed data access restrictions. If the list of characteristics is specified not by a query, but by the table name, then the `ALLOWED` keyword is also used in the query for obtaining the list of characteristics generated by 1C:Enterprise.
- `KEYFIELD` - the name of the field containing the characteristic ID.
- `NAMEFIELD` - the name of the field containing the characteristic name.
- `VALUETYPEFIELD` - the name of the field containing the value type of the characteristic. If the value type is not specified, it is considered that the characteristic is of `Boolean` type.
- `CHARACTERISTICVALUES` - table name or query to retrieve characteristic values.
- `OBJECTFIELD` - the name of the field containing the object ID (for example, the products reference).
- `TYPEFIELD` - the name of the field containing the characteristic ID.
- `VALUEFIELD` - the name of the field containing the value of the characteristic. If not specified, the value will be `True` (if the object has such characteristic), otherwise - `False`.

Example:

```
{CHARACTERISTICS TYPE(Catalog.Products)
  CHARACTERISTICTYPES (SELECT
    AdditionalPropertiesTypes.Reference,
    AdditionalPropertiesTypes.Name,
    AdditionalPropertiesTypes.ValueType
  FROM
    ChartOfCharacteristicTypes.AdditionalPropertiesTypes AS
AdditionalPropertiesTypes)
  KEYFIELD Reference
  NAMEFIELD Name
  VALUETYPEFIELD ValueType
  CHARACTERISTICVALUES InformationRegister.AdditionalProperties
  OBJECTFIELD Products
  TYPEFIELD PropertyType
  VALUEFIELD Property
}
```

Parameters

Description:

In addition to the basic elements, the data composition system accepts elements recorded in the parameters of virtual tables. In such cases, the type of fields depends on the type of parameter where the elements are located. `StartDate`, `EndDate`, `Products` and `Warehouse` fields (see example) will become available in filtering, i.e. the user will be able to apply filters for them.

Example:

```
SELECT
  ProductAccountingTurnover.Products AS Products,
  ProductAccountingTurnover.Warehouse AS Warehouse,
  ProductAccountingTurnover.ReceiptQuantity AS ReceiptQuantity,
  ProductAccountingTurnover.SalesQuantity AS SalesQuantity
FROM
  AccumulationRegister.ProductAccounting.Turnover({&StartDate},
     {&EndDate},
     ,
     {Products.*,
     Warehouse.*}) AS ProductAccountingTurnover
```

10.3.3.2. Automatic population of available fields

When automatically populating the available query fields, the following actions are performed:

- all the fields in the selection list and their child fields become available for selection, ordering, grouping, filtering, etc.;
- parameters of virtual tables become available for filtering.

10.3.4. Data composition system expression language

Data composition system expression language is intended for writing expressions used in various subsystems. Expressions are used in the following subsystems:

- data composition system - to describe calculated fields, total fields, linking expressions, etc.;
- data composition settings - to describe the user field expressions;
- data composition template - to describe the data set linking expressions, template parameters, etc.

NOTE. The data composition expression language is not capable of getting a field separated by dot from the expression.
IMPORTANT! The calculation of the data composition system expression is performed in a safe mode.

10.3.4.1. Literals

The expression may contain type literals described below.

String

Description:

String literal is enclosed in double quotation marks (").

If you need to include a double quotation mark (") in a string literal, use two double quotation marks.

Example:

```
"Literal ""with a quoted text"""
```

Number

Description:

Numbers are written without spaces, in decimal format. Use period (.) as a decimal separator.

Example:

```
10.5
200
```

Date

Description:

`Date` type literal is written using the `DATETIME` keyword. After the keyword, specify the year, month, day, hours, minutes, and seconds in parentheses, separated by commas. Specifying the time part is optional.

Example:

```
// January 6, 1975
DATETIME(1975, 1, 06)

// December 2, 2006, 23 hours 56 minutes 57 seconds
DATETIME(2006, 12, 2, 23, 56, 57)
```

Boolean

Description:

For Boolean values, use `True` and `False` literals.

Value

Description:

For literals of other types (system enumerations or predefined data), use the `Value` keyword followed by the literal name in parentheses.

Example:

```
Value(AccountType.Active)
```

Fields

Description:

Expressions can include fields of data sets. A field is identified by data path. Parts of a data path are separated by dots "`.`" Field names are not case-sensitive. If the data path contains an ID with spaces or special characters, then such identifiers must be enclosed in brackets.

Example:

```
Item.SKU
Sales.AmountTurnover
Sales.[Amount turnover]
```

Parameters

Description:

Expressions can include parameters. To include a parameter in an expression, precede the parameter name with an ampersand "&".

Example:

```
&Contractor
&StartDate
```

Type

Description:

Generates `Type` type value. Type is set using `Type` keyword.

Example:

```
Type("String")
```

10.3.4.2. Operations with numbers

Unary "-"

Description:

This operation negates a number.

Example:

```
 Sales.Quantity
```

Unary "+"

Description:

This operation does not change a number.

Example:

```
+Sales.Quantity
```

Binary "-"

Description:

This operation calculates a difference of two numbers.

Example:

```
BalanceAndTurnover.OpeningBalance -
BalanceAndTurnover.ClosingBalance
BalanceAndTurnover.OpeningBalance - 100
400 - 357
```

Binary "+"

Description:

This operation calculates a sum of two numbers.

Example:

```
BalanceAndTurnover.OpeningBalance + BalanceAndTurnover.Turnover
BalanceAndTurnover.OpeningBalance + 100
400 + 357
```

Multiplication ""*

Description:

This operation calculates a product of two numbers.

Example:

```
Item.Price * 1.2
2 * 3.14
```

Division "/"

Description:

This operation calculates a quotient of two numbers.

Example:

```
Item.Price / 1.2
2 / 3.14
```

Remainder "%"

Description:

This operation calculates a remainder in division of two numbers.

Example:

```
Item.Price % 1.2
2 % 3.14
```

10.3.4.3. Operations with strings

Concatenation (Binary "+")

Description:

This operation concatenates two strings.

Example:

```
Product.SKU + ": ”+ Product.Description
```

LIKE

Description:

This operation checks whether a string matches the specified template.

The value of the `LIKE` operator is `True` if the expression value matches the template, and `False` - otherwise.

The following characters in a template line have a specific meaning different from an ordinary string character:

- "`%`" - percent sign: a sequence containing zero and more arbitrary characters.
- "`_`" - underscore: Any one character.
- «`[...]`» - один или несколько символов в квадратных скобках: any character from a set of characters enclosed in brackets. A character set can contain ranges, such as `a-z`, meaning any character within the range, including the limits of the range.
- "`[^...]`" - a negation sign followed by one or more characters in brackets: any character except those listed after the negation sign.

Any other character represents itself only and does not have any additional meaning. If it is necessary to write one of the listed characters as itself, preceded it by with a special character specified after the `ESCAPE` keyword.

For example, the following template stands for a substring including the following sequence of characters:

- letter `A`,
- letter `B`,
- letter `C`,
- one digit,
- one of the letters `a`, `b`, c or `d`,

- one of the symbols [,] or ^,
- underscore character,
- letter a,
- letter b,
- letter c.

This sequence can begin anywhere in the string.

Example:

```
"%АБВ[0-9][абвг][\[\]\    ]\_abc%" SPECIAL SYMBOL "\"
```

10.3.4.4. Comparison operations

Equals (=)

Description:

This operation compares two operands for equality.

Example:

```
Sales.Contractor = Sales.MainItemSupplier
```

Not equal (<>)

Description:

This operation checks whether the first operand is not equal to the second operand.

Example:

```
Sales.Contractor <> Sales.MainItemSupplier
```

Less than (<)

Description:

This operation checks whether the first operand is less than the second operand.

Example:

```
CurrentSales.Amount < PreviousSales.Amount
```

Greater (>)

Description:

This operation checks whether the first operand is greater than the second operand.

Example:

```
CurrentSales.Amount > PreviousSales.Amount
```

Less or equal (<=)

Description:

This operation checks whether the first operand is less than or equal to the second operand.

Example:

```
CurrentSales.Amount <= PreviousSales.Amount
```

Greater or equal (>=)

Description:

This operation checks whether the first operand is greater than or equal to the second operand.

Example:

```
CurrentSales.Amount >= PreviousSales.Amount
```

The string begins with a substring (BeginsWith)

Description:

This operation checks that the value being checked starts with the string specified in the condition. The condition can be specified by a direct value and template. The following characters in a template line have a specific meaning different from an ordinary string character:

- «/» - the next character must be interpreted as a regular character.
- "%" - percent sign: a sequence containing zero and more arbitrary characters.
- "_" - underscore: Any one character.

The string does not start with a substing (NotBeginsWith)

Description:

This operation checks that the value being checked does not start with the string specified in the condition. The condition can be specified by a direct value and template. The following characters in a template line have a specific meaning different from an ordinary string character:

- «/» - the next character must be interpreted as a regular character.
- "%" - percent sign: a sequence containing zero and more arbitrary characters.

- "_" - underscore: Any one character.

The string matches the template (Like)

Description:

This operation checks that the value being checked matches the specified template. The following characters in a template line have a specific meaning different from an ordinary string character:

- «/» - the next character must be interpreted as a regular character.
- "%" - percent sign: a sequence containing zero and more arbitrary characters.
- "_" - underscore: Any one character.

The string does not match the template (NotLike)

Description:

This operation checks that the value being checked does not match the specified template. The following characters in a template line have a specific meaning different from an ordinary string character:

- «/» - the next character must be interpreted as a regular character.
- "%" - percent sign: a sequence containing zero and more arbitrary characters.
- "_" - underscore: Any one character.

List presence check operation (IN)

Description:

This operation checks whether a value is present in the passed list of values. The operation result is `True` if the value is found, or `False` - otherwise.

Example:

```
Item IN (&Product1, &Product2)
```

The operation of checking the value in the data set (IN)

Description:

This operation checks whether a value is present in the specified data set. The data set must contain a single field.

Example:

```
Sales.Contractor IN Contractors
```

NULL value check operation (IS NULL)

Description:

This operation returns `True` if the specified value is `NULL`.

Example:

```
Sales.Contractor IS NULL
```

The operation of checking the value for NULL inequality (IS Not NULL)

Description:

This operation returns `True` if the specified value is not `NULL`.

Example:

```
Sales.Contractor IS Not NULL
```

10.3.4.5. Logical operations

Logical expressions use `Boolean` type values as operands.

Not operation

Description:

`Not` operation returns `True` if its operand is `False`, and `False` if its operand is `True`.

Example:

```
Not Document.Consignee = Document.Shipper
```

And operation

Description:

`And` operation returns `True` if both operands are `True` and `False` if one of the operands is `False`, for example:

Example:

```
Document.Consignee = Document.Shipper AND Document.Consignee =
&Contractor
```

Or operation

Description:

`Or` operation returns `True` if one of the operands is `True` and `False` if both operands are `False`.

Example:

```
Document.Consignee = Document.Shipper Or Document.Consignee =
&Contractor
```

10.3.4.6. Aggregate functions

Aggregate functions perform certain actions on data sets.

SUM

Syntax:

`Sum(Expression)`

Description:

`Sum` aggregate function calculates a sum of values for expressions passed to it as an argument, for all the detailed records. An `Array` may be passed as a parameter. In this case the function will be applied to an array contents.

Example:

```
SUM(Sales.AmountTurnover)
```

COUNT

Syntax:

`Count(Expression)`

Description:

`Count` function calculates the number of values, that are different from `NULL`. An `Array` may be passed as a parameter. In this case the function will be applied to an array contents.

Example:

```
COUNT(Sales.Contractor)
```

COUNT (DISTINCT)

Syntax:

`Count(Distinct Expression)`

Description:

This function calculates the number of distinct values. To retrieve distinct values specify `Distinct` before `Count` method parameter. An `Array` may be passed as a parameter. In this case the function will be applied to an array contents.

Example:

```
COUNT(Distinct Sales.Contractor)
```

MAX

Syntax:

`Max(Expression)`

Description:

This function gets the maximum value. An `Array` may be passed as a parameter. In this case the function will be applied to an array contents.

Example:

```
MAX(Balances.Quantity)
```

MIN

Syntax:

`Min(Expression)`

Description:

This function gets the minimum value. An `Array` may be passed as a parameter. In this case the function will be applied to an array contents.

Example:

```
MIN(Balances.Quantity)
```

AVG

Syntax:

`Avg(Expression)`

Description:

This function retieves the mean value for values other than `NULL`. An `Array` may be passed as a parameter. In this case the function will be applied to an array contents.

Example:

```
AVG(Balances.Quantity)
```

ARRAY

Syntax:

`Массив([Различные] Выражение)`

Description:

The function generates an array containing the value of the expression for each detailed record.

You can use a value table as a parameter. In this case, the function returns an array that contains values from the first column of the value table, passed as a parameter.

If an expression contains an `Array` function, it is considered an aggregate expression.

If the `Distinct` keyword is specified, the resulting array will not contain duplicate values.

Example:

```
ReportFunctions.StdDeviation(Array(Sum))
```

VALUESTABLE

Syntax:

`ValuesTable([Различные] Expression1 [AS ColumnName1][, Expression2 [AS ColumnName2]], …)`

Description:

This function generates a value table that contains the number of columns that matches the number of function parameters. Detailed records are retrieved from data sets that are required for calculating all the expressions, specified in the function parameters.

If the function parameters are balance fields, the resulting value table contains the values from the records retrieved by unique combinations of dimensions from other periods. At the same time the values are only retrieved for balance fields, dimensions, accounts, period fields and their attributes. Values of the other fields in records from other periods are considered to be `NULL`.

If an expression contains a `ValuesTable` function, it is considered an aggregate expression.
If the `Distinct` keyword is specified, the resulting value table will not contain rows containing the same data.
For each column, it is possible to specify a name using the `AS` keyword, following the expression that generates the column value.

Example:

```
ReportsFunctions.ValueTableToString(ValuesTable(Warehouse AS
Warehouse, BalanceQuantity AS Balance))
```

GROUPBY

Syntax:

`Groupby(<Expression>, <ColumnNumbers>)`

Description:

This function deletes duplicates from an array.

Parameters:

`<Expression>`

Type: `Array` or `ValuesTable`. Removal of duplicates is performed for the value placed in this formal parameter.

`<ColumnNumbers>`

`String`. Used, if `Expression` parameter is of `ValuesTable` type. Value table column numbers or names (comma-separated) to search for duplicates. By default, the search is performed in all the columns.

Example:

```
GroupBy(ValuesTable(PhoneNumber, Address) ,"PhoneNumber");
```

GETPART

Syntax:

`GETPART(<Expression>, <ColumnNumber>)`

Description:

The function retrieves a value table that contains specified columns of the source value table.

Parameters:

<Expression>

Type: `ValuesTable`. Value table to retrieve the columns from.

<ColumnNumbers>

Type: `String`. Value table column names or numbers (comma-separated) to be retrieved.

Returns:

Value table, containing only those columns that are specified in the `CoiumnNumbers` parameter.

Example:

```
GetPart(GroupBy(ValuesTable(PhoneNumber, Address),
"PhoneNumber"),"PhoneNumber");
```

ORDER

Syntax:

`Order(<Expression>, <ColumnNumbers>)`

Description:

Orders array and value table elements.

Parameters:

<Expression>

Type: `Array` or `ValuesTable`. Object to be ordered.

<ColumnNumbers>

Used, if `Expression` parameter is of `ValuesTable` type. Value table column names or numbers (comma-separated) to be ordered. By default, the search is performed in all the columns.
Direction of ordering or an automatic ordering flag may be set after each column.

Returns:

An array or value table that are ordered according to the passed parameters.

Example:

```
Order(ValuesTable(PhoneNumber, Address, CallDate),"CallDate Desc");
```

JOINSTRINGS

Syntax:

```
JOINSTRINGS(<Values>,            <ElementSeparator>,
<ColumnSeparator>)
```

Description:

Joins multiple strings.

Parameters:

<Values>

Values string presentatons of which are to be joined in a string. If parameter value type is `Array`, array elements are joined in a string. If the parameter value type is `ValuesTable`, all the table columns and rows are joined in a string.

<ElementSeparator>

Type: `String`. Contains the text to be used as a separator between array elements and value table rows. The default value is the line feed character.

<ColumnSeparator>

Type: `String`. Contains the text to be used as a separator between value table columns. "; " by default.

Returns:

Joined string.

Example:

```
JoinStrings(ValuesTable(PhoneNumber, Address));
```

GROUPPROCESSING

Syntax:

```
GROUPPROCESSING(<Expressions>,
<HierarchyExpressions>, <GroupingName>)
```

Description:

A value table is generated to contain the parameter values (in columns) for each group record (in rows). If the grouping is hierarchical, each hierarchy level is processed separately. Hierarchical record values are also placed in the

data table. This value table will be placed in the `Data` property of the returned object.
A row of the passed value table for which the function is currently being evaluated is placed to the `CurrentItem` property.
When implementing a function that can take collective processing data as a parameter, it must be considered that the `NULL` value can be passed to the function. For example, this may occur if the `GroupProcessing()` function is evaluated with the name of the grouping that is currently not available.

Parameters:

`<Expressions>`

A string with comma-separated expressions to be evaluated. Each expression can be followed by an optional `AS` keyword and a column name in the resulting value table. Each expression describes a column of the generated value table.

`<HierarchyExpressions>`

Expressions to be calculated for hierarchical records. It is similar to the `Expressions` parameter, but `HierarchyExpressions` is used for hierarchical records, while `Expressions` is used for nonhierarchical records. If this parameter is not specified, expressions specified in the `Expression` parameter are used to calculate hierarchical record values.

`<GroupingName>`

The name of the grouping for group processing calculation. If this parameter is not specified, the calculation is performed in the current grouping. If the calculation is performed in a table and the parameter is a blank string or is not specified, the value is calculated for the row - grouping. When data composition template is generated the template compiler replaces this name with the grouping name in a resulting data composition template. If the grouping is unavailable, the function is replaced with `NULL`.

Returns:

Object `DataCompositionGroupProcessingData`.

Example:

An example of the `ABCClassification()` function implementation, which returns the value 1 if the value is 75% of the total amount, 2 if the percentage is between 75% and 95%, and 3 in other cases.

```
// Evaluate ABC classification.
Function ABCClassification(Data) Export
Var ValuesTable;

  If Data = Null Then
    Return Null;
  EndIf;
  If TypeOf(Data) <> Type("DataCompositionGroupProcessingData")
Then
    Message(TypeOf(Data));
    Raise "Only DataCompositionGroupProcessingData object type can
be passed to ABCClassification() function";
  EndIf;
  If Not
Data.ProcessingTempData.Property("ABCClassificationValueTable",
ValuesTable) Then
    // evaluate the classification once at the first run
    // and then store the performed calculation and will
    // use it
    ValuesTable = Data.Data.Copy();
    ValuesTable.Columns.Add("Number", New
TypeDescription("Number"));
    Number = 0;
    TotalSum = 0;
    For Each ValueTableRow From ValuesTable Do
      ValueTableRow.Number = Number;
      Number = Number + 1;
      If ValueTableRow[0] <> NULL Then
        TotalSum = TotalSum + ValueTableRow[0];
      EndIf;
    EndDo;
    ValuesTable.Sort(ValuesTable.Columns[0].Name + "Desc");
    ValuesTable.Indexes.Add("Number");
    AccumulatedSum = 0;
    ClassAIndex = Undefined;
    ClassBIndex = Undefined;
    For Each ValueTableRow From ValuesTable Do
      If ValueTableRow[0] <> NULL Then
        AccumulatedSum = AccumulatedSum + ValueTableRow[0];
      EndIf;
      If TotalSum = 0 Then
        Percentage = 1;
      Else
        Percentage = AccumulatedSum / TotalSum;
      EndIf;
      If Percentage > 0.75 Then
        If ClassAIndex = Undefined Then
          ClassAIndex = ValuesTable.Index(ValueTableRow);
        ElseIf Percentage > 0.90 Then
          If ClassBIndex = Undefined Then
            ClassBIndex = ValuesTable.Index(ValueTableRow);
          EndIf;
          Break;
```

```
      EndIf;
    EndIf;
  EndDo;
    Data.ProcessingTempData.Insert("ABCClassificationValueTable",
ValuesTable);
    Data.ProcessingTempData.Insert("ABCClassificationClassAIndex",
ClassAIndex);
    Data.ProcessingTempData.Insert("ABCClassificationClassBIndex",
ClassBIndex);
  EndIf;
  If Data.CurrentItem = Undefined Then
    // Totals by grouping.
    Return Null;
  Else
    String = ValuesTable.Find(Data.Data.Index(Data.CurrentItem),
"Number");
    If String = Undefined Then
      Return Null;
    Else
      Index = ValuesTable.Index(String);
      If Index <=
Data.ProcessingTempData.ABCClassificationClassAIndex Then
        Return 1;
      ElseIf Index <=
Data.ProcessingTempData.ABCClassificationClassBIndex Then
        Return 2;
      Else
        Return 3;
      EndIf;
    EndIf;
  EndIf;
EndFunction
```

The following expression can be used in a composition expression to retrieve a class (for example, in a resource or a custom field):

```
ABCClassification(GroupProcessing("Sum(TurnoverSum)"))
```

EVERY

Syntax:

`Every(<X>)`

Description:

Aggregate function `Every` determines the presence of at least one `False` value in a passed set.

Returns:

`True` - if the passed set does not contain `False` value.
`False` - if the passed set contains at least one `False` value.

ANY

Syntax:

`Any(<X>)`

Description:

Aggregate function `Any` determines the presence of at least one `True` value in a passed set.

Returns:

`True` - if the passed set contains at least one `True` value.
`False` - if the passed set does not contain `True` value.

STDDEV_POP

Syntax:

`STDDEV_POP(<X>)`

Description:

Calculates the standard deviation of population for the passed set.
Calculations are performed using the formula: `SQRT(VAR_POP(X))`.

Returns:

Function calculation result.

STDDEV_SAMP

Syntax:

`STDDEV_SAMP(<X>)`

Description:

Calculates the standard deviation of a sample for the passed set.
Calculations are performed using the formula: `SQRT(VAR_SAMP(X))`.

Returns:

Function calculation result.

VAR_SAMP

Syntax:

`VAR_SAMP(<X>)`

Description:

Calculates sample variance for the passed set.

Calculations are performed using the formula: `(SUM(X2)-SUM(X)2/COUNT(X))/(COUNT(X)-1)`. If the number of records in a set is 1 (`COUNT(X)=1`), the returned value is `NULL`.

Returns:

Function calculation result.

VAR_POP

Syntax:

```
VAR_POP(<X>)
```

Description:

Calculates the variance of the population for the passed set. `NULL` values are ignored.

Calculations are performed using the formula: `(SUM(X2)-SUM(X)2/COUNT(X))/(COUNT(X))`.

Returns:

Function calculation result.

COVAR_POP

Syntax:

```
COVAR_POP(<Y>, <X>)
```

Description:

Calculates the covariance of a set of pairs for passed sets.

Calculations are performed using the formula: `(SUM(Y*X)-SUM(X)*SUM(Y)/N)/N`. `N` - number of `X` and `Y` value pairs from the passed sets, where neither `X`, not `Y` values are `NULL`. Pairs with at least one `NULL` value are ignored.

Returns:

Function calculation result.

COVAR_SAMP

Syntax:

```
COVAR_SAMP(<Y>, <X>)
```

Description:

Calculates the covariance sample of a set of pairs for passed sets.

Calculations are performed using the formula: `(SUM(Y*X)-SUM(Y)*SUM(X)/N)/(N-1)`. `N` - number of `X` and `Y` value pairs from the

passed sets, where neither X, not Y values are NULL. Pairs with at least one NULL value are ignored.

Returns:

Function calculation result or NULL - if the function is applied to empty sets.

CORR

Syntax:

```
CORR(<Y>, <X>)
```

Description:

Calculates the correlation factor of a set of pairs for passed sets. Calculations are performed using the formula: COVAR_POP(Y, X)/(STDDEV_POP(Y)* STDDEV_POP(X)). Pairs with at least one NULL value are ignored.

Returns:

Function calculation result or NULL - if the function is applied to empty sets.

REGR_SLOPE

Syntax:

```
REGR_SLOPE(<Y>, <X>)
```

Description:

Calculates the slope of the regression line.

Calculations are performed using the formula: COVAR_POP(Y, X)/VAR_POP(X). Pairs with at least one NULL value are ignored.

Returns:

Function calculation result.

REGR_INTERCEPT

Syntax:

```
REGR_INTERCEPT(<Y>, <X>)
```

Description:

Calculates the Y-intercept of the regression line.

Calculations are performed using the formula: Avg(Y)-REGR_SLOPE(Y, X)*Avg(X). Pairs with at least one NULL value are ignored.

Returns:

Function calculation result.

REGR_COUNT

Syntax:

```
REGR_COUNT(<Y>, <X>)
```

Description:

This function calculates the number of non-`NULL` number pairs.

Returns:

Function calculation result.

REGR_R2

Syntax:

```
REGR_R2(<Y>, <X>)
```

Description:

Calculates the determination factor for the regression. The function is calculated without considering the pairs of values containing `NULL`.

Returns:

- `NULL`, if `VAR_POP(X)` is `0`.
- `1`, if `VAR_POP(Y)` is `0` and `VAR_POP(X)` is not `0`.
- `Pow(CORR(Y, X), 2)`, if `VAR_POP(Y)` is more than `0` and `VAR_POP(X)` is not `0`.

REGR_AVGX

Syntax:

```
REGR_AVGX(<Y>, <X>)
```

Description:

Calculates the average of the independent `<X>` variables for the regression line, after eliminating the pairs with at least one `NULL` value.
Calculations are performed using the formula: `Avg(X)`.

Returns:

Function calculation result.

REGR_AVGY

Syntax:

```
REGR_AVGY(<Y>, <X>)
```

Description:

Calculates the average of the dependent `<Y>` variables for the regression line, after eliminating the pairs with at least one `NULL` value.
Calculations are performed using the formula: `Avg(Y)`.

Returns:

Function calculation result.

REGR_SXX

Syntax:

```
REGR_SXX(<Y>, <X>)
```

Description:

Performs calculation as per formula:
`REGR_COUNT(Y, X) * VAR_POP(X)`. Pairs with at least one `NULL` value are ignored.

Returns:

Function calculation result.

REGR_SYY

Syntax:

```
REGR_SYY(<Y>, <X>)
```

Description:

Performs calculation as per formula:
`REGR_COUNT(Y, X) * VAR_POP(Y)`. Pairs with at least one `NULL` value are ignored.

Returns:

Function calculation result.

REGR_SXY

Syntax:

```
REGR_SXY(<Y>, <X>)
```

Description:

Performs calculation as per formula:
`REGR_COUNT(Y, X) * COVAR_POP(Y, X)`. Pairs with at least one `NULL` value are ignored.

Returns:

Function calculation result.

RANK

Syntax:

```
RANK(<Order>, <HierarchyOrder>, <GroupName>)
```

Description:

The function calculates the place of the current record among the records of the current grouping, if ordered in the order specified by the parameters of the function. Numbering starts with 1.

Parameters:

<Order>

Type: `String`. Contains comma-separated expressions for ordering group records. You can set the order direction by using the words `Asc` and `Desc`. It is possible to add `AutoOrder` string after the field, which means that the ordering fields defined for the referred object are used for ordering references. If the sequence is not specified, the value is calculated based on the grouping sequence.

<HierarchyOrder>

Type: `String`. String containing expressions for ordering hierarchical records.

<GroupingName>

Type: `String`. The name of the grouping where the function is calculated. If this parameter is not specified, the calculation is performed in the current grouping. If the calculation is performed in a table and the parameter is a blank string or is not specified, the value is calculated for the row grouping. When the data composition template is generated, the template compiler replaces this name with the grouping name in the resulting template. If the grouping is unavailable, the function is replaced with `NULL`.

Returns:

Serial number. If a sequence includes multiple records with identical order field values, the function returns identical values for these records.

CLASSIFICATIONABC

Syntax:

```
ClassificationABC(<Value>,              <GroupCount>,
<PercentageForGroups,> <GroupingName>)
```

Description:

Performs an ABC classification of each record in the specified grouping.

Parameters:

<Value>

Type: String. Specifies an expression for calculating the classification.

<GroupCount>

Type: Number. Specifies how many groups the set of values must be broken into.

<PercentageForGroups>

Type: String. The volume (in %) of each splitting group, except the last group. Listed in a string, separated by commas.

<GroupingName>

Type: String. The name of the grouping where the function is calculated. If this parameter is not specified, the calculation is performed in the current grouping. If the calculation is performed in a table and the parameter is a blank string or is not specified, the value is calculated for the row grouping. When the data composition template is generated, the template compiler replaces this name with the grouping name in the resulting template. If the grouping is unavailable, the function is replaced with NULL.

Returns:

Class number. 1 corresponds class A, 2 - class B, 3 - class C, etc.

Example:

```
ClassificationABC("AmountTurnover, 3, "15, 25")
```

10.3.4.7. Other operations

CASE operation

Description:

CASE operation selects one of several values based on the specified conditions.

Example:

```
CASE When Amount > 1000 Then Amount Else 0 End
```

10.3.4.8. Rules for comparing two values

If types of the compared values are not identical, the relationship between the values is defined based on the type priority:

- `NULL` (the lowest priority);
- `Boolean`;
- `Number`;
- `Date`;
- `String`;
- reference types;
- other types.

The relationship between different reference types is defined based on table reference numbers corresponding to one type or another. The relations between other types are defined by the comparison of the CID of these types.

If the data types are identical, their values are compared according to the following rules:

- For `Boolean` type, `True` is greater than `False`.
- For `Number` type, common number comparison rules are applied.
- For `Date` type, earlier dates are less than later dates.
- For `String` type, string comparison is based on the regional database settings.
- Reference types are compared by their values (for example, record numbers, etc.).
- The result of consecutive comparison of the items of the array will be the result of the comparison of arrays.
- The result of consecutive comparison of the cells of tables will be the result of the comparison of the tables.
- The following rule is applied for the comparison of objects which comparison is not described above:
  - The comparison by the string representation of compared objects is used for ranking.
  - When comparing in expressions, in `=` and `<>` operators, the algorithm output is the result of the check whether an object is being compared to itself.
  - The comparison operations `<`, `<=`, `>`, `>=` are not supported with the generation of exclusion.

10.3.4.9. Operations with NULL value

Any operation where one of the operands is `NULL` value returns `NULL`.
The exceptions are:

- `And` operation returns `NULL` only if none of the operands is `False`;
- `Or` operation returns `NULL` only if none of the operands is `True`.

10.3.4.10. Operation priorities

Operations have the following priorities (the first operation in the list has the lowest priority):

- `Or`;
- `And`;
- `Not`;
- `IN`, `IS NULL`, `IS Not NULL`;
- =, <>, <=, <, >=, >;
- Binary +, Binary -;
- *, /, %;
- Unary +, Unary -.

10.3.4.11. Functions

EVAL

Syntax:

```
Eval(<Expression>, <Grouping>, <CalculationType>)
```

Description:

`EVAL` function calculates an expression within a certain grouping context. The function is used for compatibility with previous 1C:Enterprise versions. We recommend to use the `EvalExpression` function instead.

Parameters:

`<Expression>`

A string containing an expression to be calculated.

`<Grouping>`

A string containing a name of a grouping in the context of which an expression is calculated. If the grouping name is a blank string, the calculation is performed in the context of the current grouping. If a `Overall` string is used as a grouping name, the calculation is performed in the context of the grand total. In other cases the calculation is performed in the context of the parent grouping with this name.

<CalculationType>

A string containing the calculation type. If the parameter value is `Overall`, the expression is calculated for all the records of the grouping. If the parameter value is - `Grouping`, the values are calculated for the current group record of the grouping.

Example:

```
Sum(Sales.AmountTurnover)/EVAL("Sum(Sales.AmountTurnover)",
"Overall")
```

In this example the result is the ratio of the sum of the grouping record by the `Sales.AmountTurnover` field to the sum of all composition records by this field.

EVALEXPRESSION

Syntax:

```
EvalExpression(<Expression>,            <Grouping>,
<CalculationArea>,   <Start>,   <End>,   <Sorting>,
<HierarchicalSorting>,
<IdenticalOrderValuesProcessing>)
```

Description:

This function calculates an expression within a certain grouping context. When the interval is specified, the calculation is possible only for the aggregate functions of the data composition system; the calculation is not performed for the other functions.

The function performs the calculation after applying the grouping filter. However, it ignores hierarchical filters. The function cannot be applied to a grouping in a group filter of that grouping. For example, in `Products` grouping filter using `EvalExpression("Sum(AmountTurnoevr)",  , "Overall") > 1000` expression is not possible. But this can be used in a hierarchical filter. If the last record precedes the first one, it is considered that there are no records for calculating detailed data and aggregate functions. When interval expressions are calculated for the grand total (the `Grouping` parameter has the `Overall` value), it is considered that there are no records for calculating detailed data and aggregate functions.

If the ordering expression contains fields that cannot be used in the grouping, then the template compiler, when generating the `EvalExpression` function expression, replaces the `EvalExpression` function with `NULL`.

Parameters:

<Expression>

Type: `String`. Expression to be calculated.

<Grouping>

Type: `String`. Contains name of the grouping where the expression is calculated. If a blank string is used as a grouping name, the calculation is performed within the current grouping context. If a `Overall` string is used as a grouping name, the calculation is performed in the context of the grand total. In other cases the calculation is performed in the context of the parent grouping with this name.

Example:

```
Sum(Sales.AmountTurnover) / Eval("Sum(Sales.AmountTurnover)",
"Overall")
```

In this example the result is the ratio of the sum of the grouping record by the `Sales.AmountTurnover` field to the sum of all composition records by this field.

<CalculationArea>

Type: `String`. The parameter can take values:

- `Overall` - expression will be evaluated for all the grouping records.
- `Hierary` - expression will be calculated for the parent hierarchical record, if any, or for the entire grouping if there is no parent hierarchical record.
- `Grouping` - expression will be calculated for the current group record of the grouping.
- `NonResourceGrouping` - when calculating the function for group record by resources, the expression will be calculated for the first group record of the original grouping.
  When calculating the `EvalExpression()` function with the `NonResourceGrouping` value for group records that are not groupings by resources, the function is calculated in the same way as if the parameter value were `Grouping`.
  The data composition template compiler when generating a data composition template at outputting a resource field, by which the grouping is performed, to a template group, places an expression that is calculated using the `EvalExpression()` function to the template, specifying the `NonResourceGrouping` parameter. For other grouping resources, standard resource expressions are placed.

<Start>

Type: `String`. Specifies the first record of a fragment where the aggregate functions of the expression are calculated and the record for retrieving field values outside aggregate functions. Value may be one of the following:

- `First`. Get the first grouping record. It is possible to specify an expression that defines the offset from the beginning of the grouping after the word in parentheses. The retrieved value must be a positive integer. For example, `First(3)` - returns the third record from the beginning of the grouping.
  If the first record is beyond the grouping, it is considered that there are no records. For example, if there are 3 records and `First(4)` is required to be retrieved, it is considered that there are no records.
- `Last`. Get the last grouping record. After the word in parentheses, it is possible to specify an expression, the result of which will be used as an offset from the end of the grouping. The retrieved value must be a positive integer. For example, with `Last(3)` - the third record from the end of the grouping is retrieved.
  If the last record is beyond the grouping, it is considered that there are no records. For example, if there are 3 records and `Last(4)` is required to be retrieved, it is considered that there are no records.
- `Previous`. Get the previous grouping record. After the word in brackets, it is possible to specify an expression, the result of which will be used as an offset backwards from the current grouping record. For example, `Previous(2)` - the previous from the previous record is retrieved.
  If the previous record is beyond the grouping (for example, it is required to retrieved `Previous(3)` for the second record of the grouping), the result is the first grouping record.
  When the previous record is retrieved for a grouping total, the result is the first record.
- `Next`. Get the next grouping record. After the word in parentheses, it is possible to specify an expression, the result of which will be used as an offset forward from the current grouping record. For example, `Next(2)` - the next from the next record is retrieved.
  If the next record is beyond the grouping, it is considered that there are no records. For example, if there are 3 records and `Next()` is specified for the third record, it is considered that there are no records.
  If the next record is retrieved for a grouping total, it is considered that there are no records.
- `Current`. Get the current record.

If a record is retrieved for a grouping total, the result is the first record.

- `BoundaryValue`. It is required to retrieve a record by the specified value. The `BoundaryValue` word must be followed by an expression of the first order field that begins the fragment, in parentheses.
  The first record with an order field value greater than or equal to the specified value is retrieved as a record. For example, if the `Period` field is used as an order field, this field has the values 01.01.2010, 02.01.2010 and 03.01.2010, and it is required to retrieve `BoundaryValue(DateTime(2010, 1, 15))`, a record with the date 02.01.2010 is retrieved.

<End>

Type: `String`. Specifies the last record of a fragment where the aggregate functions are calculated. Value may be one of the following:

- `First`. Get the first grouping record. It is possible to specify an expression that defines the offset from the beginning of the grouping after the word in parentheses. The retrieved value must be a positive integer. For example, `First(3)` - returns the third record from the beginning of the grouping.
  If the first record is beyond the grouping, it is considered that there are no records. For example, if there are 3 records and `First(4)` is required to be retrieved, it is considered that there are no records.
- `Last`. Get the last grouping record. After the word in parentheses, it is possible to specify an expression, the result of which will be used as an offset from the end of the grouping. The retrieved value must be a positive integer. For example, with `Last(3)` - the third record from the end of the grouping is retrieved.
  If the last record is beyond the grouping, it is considered that there are no records. For example, if there are 3 records and `Last(4)` is required to be retrieved, it is considered that there are no records.
- `Previous`. Get the previous grouping record. After the word in brackets, it is possible to specify an expression, the result of which will be used as an offset backwards from the current grouping record. For example, `Previous(2)` - the previous from the previous record is retrieved.
  If the previous record is beyond the grouping (for example, it is required to retrieve `Previous(3)` for the second record of the grouping), the result is the first grouping record.
  When the previous record is retrieved for a grouping total, the result is the first record.

- `Next`. Get the next grouping record. After the word in parentheses, it is possible to specify an expression, the result of which will be used as an offset forward from the current grouping record. For example, `Next(2)` - the next from the next record is retrieved.
  If the next record is beyond the grouping, it is considered that there are no records. For example, if there are 3 records and `Next()` is specified for the third record, it is considered that there are no records.
  If the next record is retrieved for a grouping total, it is considered that there are no records.
- `Current`. Get the current record.
  If a record is retrieved for a grouping total, the result is the first record.
- `BoundaryValue`. It is required to retrieve a record by the specified value. The `BoundaryValue` word must be followed by an expression of the first order field that ends the fragment, in parentheses.
  The first record with an order field value greater than or equal to the specified value is retrieved as a record. For example, if the `Period` field is used as an order field, this field has the values 01.01.2010, 02.01.2010 and 03.01.2010, and it is required to retrieve `BoundaryValue(DateTime(2010, 1, 15))`, a record with the date 02.01.2010 is retrieved.

<Sorting>

Type: `String`. Lists comma-separated expressions that describe the ordering rules. If no string is specified, the values are ordered similarly to the grouping for which an expression is calculated. Each expression may be followed by the `Ascending` (for ascending ordering) or the `Desc` keyword (for descending ordering), or `Autoorder` (to order reference fields by fields to order the referenced object by). `Autoorder` word may be used with both `Ascending` and `Desc` words.

<HierarchicalSorting>

Type: `String`. Similar to `Sorting` parameter. It is used for ordering hierarchical records. If this parameter is not specified, the template compiler orders the fields based on the `Sorting` parameter.

<IdenticalOrderValuesProcessing>

Type: `String`. Specifies the rule to determine the next or the previous record if there are multiple records with the same order field value:

- `Separately` means that the sequence of ordered records is used in order to determine the previous and next records. Default value.

- `Together` means, that the previous and next records are determined based on the ordering expression values.
  For example, a retrieved sequence is ordered by date:

No.	Date	Full name	Value
1	01 January 2001	Ivanov M.	10
2	02 January 2001	Petrov S.	20
3	02 January 2001	Sidorov R.	30
4	03 January 2001	Petrov S.	40

  If parameter value is `Separately`:
  - The previous record for record 3 will be record 2.
  - If the calculated fragment is defined as `Current, Current` (the `Start` and `End` parameters, respectively), then this fragment for record 2 will consist of the one record 2. Expression `EvalExpression("Sum(Value)", , , Current, Current)` will equal 20.

  If the parameter is `Together`, then:
  - The previous record for record 3 will be record 1.
  - If the calculated fragment is defined as `Current, Current` (the `Start` and `End` parameters, respectively), then this fragment for record 2 will consist of records 2 and 3. Expression `EvalExpression("Sum(Value)", , , Current, Current)` will equal 50.

  If `Together` is specified, offsets for the `First`, `Last`, `Previous`, and `Next` positions cannot be specified in the `Start` and `End` parameters.

Example:

To calculate an accumulated sum, use the following expression:

```
EvaluateExpression("Sum(AmountTurnover)", , , "First", "Current")
```

To retrieve a grouping value in the previous string, use the following expression:

```
EvaluateExpression("Rate", , , "Previous")
```

EVALEXPRESSIONWITHGROUPARRAY

Syntax:

```
EvalExpressionWithGroupArray(<Expression>,
<GroupingFieldExpression>,          <RecordFilter>,
<GroupingFilter>)
```

Description:

The function returns an array with every item containing the result of calculating the expression for the grouping by the specified field.

When the template compiler generates a template, it converts the function parameters to data composition template field terms. For example, the `Counterparty` field will be converted to `DataSet.Counterparty`.

The template compiler, when generating expressions for outputting a custom field, which expression contains just the `EvalExpressionWithGroupArray()` function, generates the output expression so that the output information is ordered. For example, for a custom field with the following expression:

```
EvalExpressionWithGroupArray("Sum(AmountTurnover)", "Contractor")
```

The template composer generates the following output expression:

```
JoinStrings(Array(Order(EvalExpressionWithGroupValueTable
("Presentation(Sum(DataSet.AmountTurnover)),
Sum(DataSet.AmountTurnover)", "DataSet.Counterparty"),"2")))
```

Parameters:

<Expression>

Type: `String`. Expression to be calculated. String, for example, `Sum(AmountTurnover)`.

<GroupingFieldExpression>

Type: `String`. Grouping field expressions - comma-separated expressions of the grouping fields. For example, `Counterparty, Batch`.

<RecordsFilter>

Type: `String`. An expression describing the filter applied to the detailed records. The expression does not support the use of aggregate functions. For example, `DeletionMark = False`.

<GroupingFilter>

Type: `String`. An expression describing the filter applied to the group records. For example, `Sum(AmountTurnover) > &Parameter1`.

Example:

```
Maximum(EvalExpressionWithGroupArray ("Sum(AmountTurnover)",
"Contractor")).
```

EVALEXPRESSIONWITHGROUPVALUETABLE

Syntax:

EvalExpressionWithGroupValueTable(<Expression>, <GroupingFieldExpression>, <RecordFilter>, <GroupingFilter>)

Description:

The function returns a value table, each line of which contains the result of calculating expressions for grouping by the specified field.

When the template compiler generates a template, it converts the function parameters to data composition template field terms. For example, the Counterparty field will be converted to DataSet.Counterparty.

The template compiler, when generating expressions for outputting a custom field, which expression contains just the EvalExpressionWithGroupValueTable() function, generates the output expression so that the output information is ordered. For example, for a custom field with the following expression:

```
EvalExpressionWithGroupValueTable ("Contractor,
Sum(AmountTurnover)", "Contractor")
```

The template composer generates the following output expression:

```
JoinStrings(GetPart(Sort(EvalExpressionWithGroupValueTable
("DataSet.Contractor, DataSet.ContractorPresentation,
Sum(DataSet.AmountTurnover), Presentation(DataSet.AmountTurnover),
DataSet.SortingField", "DataSet.Contractor"), "5, 1, 3"), "2, 4"))
```

Parameters:

<Expressions>

Type: String. Expressions to be calculated. This parameter can contain multiple comma-separated expressions. Each expression can be followed by an optional AS keyword and a column name in the value table. Example: Counterparty, Sum(AmountTurnover) As SalesVolume.

<GroupingFieldExpression>

Type: String. Grouping field expressions - comma-separated expressions of the grouping fields. For example, Counterparty, Batch.

<RecordsFilter>

Type: `String`. An expression describing the filter applied to the detailed records. The expression does not support the use of aggregate functions. For example, `DeletionMark = False`.

<GroupingFilter>

Type: `String`. An expression describing the filter applied to the group records. For example, `Sum(AmountTurnover) > &Parameter1`.

Example:

```
EvalExpressionWithGroupValueTable ("Contractor AS Contractor,
Sum(AmountTurnover) AS SalesVolume", "Contractor")
```

The result of this function is a value table with `Counterparty` and `SalesVolume` columns to contain the counterparties with their respective sales volumes.

LEVEL

Description:

This function returns the current record level.

Example:

```
LEVEL()
```

LEVELINGROUP

Description:

This function returns the record level relative to the root of the grouping.

Example:

```
LEVELINGROUP()
```

VALUEISFILLED

Description:

This function returns `True` if the value is not equal to the default value for this type, is not `NULL`, is not a blank reference, and is not `Undefined`. Logical values are checked for `NULL`. Strings are checked for absence of non-space characters.

SERIALNUMBER

Description:

This function returns the next sequential number.

Example:

```
SERIALNUMBER()
```

GROUPSERIALNUMBER

Description:

This function returns the next sequential number in the current grouping.

Example:

```
SEQUENTIALNUMBERINGROUPING()
```

FORMAT

Description:

This function returns a formatted string of the passed value. The format string is set according to the formate string of 1C:Enterprise.

Parameters:

- Value,
- Format string.

Example:

```
FORMAT(Invoices.AmountDoc, "NFD=2")
```

BEGINOFPERIOD

Description:

This function extracts a certain date from a given date.

Parameters:

- `Date` type expression.
- Period type - string, containg one of the values:`Minute`, `Hour`, `Day`, `Week`, `Month`, `Quarter`, `Year`, `TenDays`, `HalfYear`. If the period type is set to `Week`, then the value of the First weekday property of the infobase regional settings is considered.

Example:

```
BEGINOFPERIOD(DateTime(2002, 10, 12, 10, 15, 34), "Month")
```

Result:

```
01.10.2002 0:00:00
```

ENDOFPERIOD

Description:

This function extracts a certain date from a given date.

Parameters:

- `Date` type expression.
- Period type - string, containg one of the values:`Minute`, `Hour`, `Day`, `Week`, `Month`, `Quarter`, `Year`, `TenDays`, `HalfYear`. If the period type is set to `Week`, then the value of the First weekday property of the infobase regional settings is considered.

Example:

```
ENDOFPERIOD(DateTime(2002, 10, 12, 10, 15, 34), "Week")
```

Result:

```
13.10.2002 23:59:59
```

DATEADD

Description:

This function adds a value to a date.

Parameters:

- `Date` type expression.
- Increment type - string, containing one of the values: `Second`, `Minute`, `Hour`, `Day`, `Week`, `Month`, `Quarter`, `Year`, `TenDays`, `HalfYear`. Note that the `Decade` term means exactly 10 days, regardless of the date specified in the first parameter of the function.
- Size - date increment size. `Number`. Fractions are ignored.

Example:

```
DATEADD(DateTime(2002, 10, 12, 10, 15, 34), "Month", 1)
```

Result:

```
12.11.2002 10:15:34
```

DATEDIFF

Description:

This function returns an interval between two dates.

Parameters:

- `Date` type expression.
- `Date` type expression.
- Difference type - one of the following values: `Second`, `Minute`, `Hour`, `Day`, `Month`, `Quarter`, `Year`.

Example:

```
DATEDIFF(DATETIME(2002, 10, 12, 10, 15, 34),
     DATETIME(2002, 10, 14, 9, 18, 06),
     "DAY")
```

Result:

```
2
```

CURRENTDATE

Description:

This function returns the system date. During the composition of the composition template all the expressions available in the composition have the `CurrentDate()` function replaced with the current date value.

Example:

```
CURRENTDATE()
```

SUBSTRING

Description:

This function extracts a substring from a string.

Parameters:

- An expression of String type.
- position of the first character of the substring to be extracted;
- The substring length.

Example:

```
SUBSTRING(Contractors.Address, 1, 4)
```

STRINGLENGTH

Description:

This function returns a string length. Parameter - string type expression.

Example:

```
STRINGLENGTH(Contractors.Address)
```

YEAR

Description:

This function extracts a year from `Date` type value. The only paremeter - is the expression of `Date` type.

Example:

```
YEAR(Invoice.Date)
```

QUARTER

Description:

This function extracts a quarter number from `Date` type value. The quarter number is in the range from 1 to 4. The only function parameter - is the expression of `Date` type.

Example:

```
QUARTER(Invoice.Date)
```

MONTH

Description:

This function extracts a month number from `Date` type value. The month number is in the range from 1 to 12. The only function parameter - is the expression of `Date` type.

Example:

```
MONTH(Invoice.Date)
```

DAYOFYEAR

Description:

This function extracts a day of the year from `Date` type value. The day number is in the range from 1 to 365 (366). The only function parameter - is the expression of `Date` type.

Example:

```
DAYOFYEAR(Invoice.Date)
```

DAY

Description:

This function extracts a day of the month from `Date` type value. The day number is in the range from 1 to 31. The only function parameter - is the expression of `Date` type.

Example:

```
DAY(Invoice.Date)
```

WEEK

Description:

This function extracts a week number from `Date` type value. The weeks of the year are numbered starting with 1. The only function parameter - is the expression of `Date` type. The calculation considers the value of the First weekday property of the infobase regional settings.

Example:

```
WEEK(Invoice.Date)
```

WEEKDAY

Description:

This function returns a day of the week from a `Date` type value. The day of the week is in the range from 1 (Monday) to 7 (Sunday). The only function parameter - is the expression of `Date` type.

Example:

```
WEEKDAY(Invoice.Date)
```

HOUR

Description:

This function returns an hour of the day from a `Date` type value. The number is in the range from 0 to 23. The only function parameter - is the expression of `Date` type.

Example:

```
HOUR(Invoice.Date)
```

MINUTE

Description:

This function returns a minute of an hour from a `Date` type value. The minute of an hour is in the range from 0 to 59. The only function parameter - is the expression of `Date` type.

Example:

```
MINUTE(Invoice.Date)
```

SECOND

Description:

This function returns the number of seconds from a `Date` type value. The second of a minute is in the range from 0 to 59. The only function parameter - is the expression of `Date` type.

Example:

```
SECOND(Invoice.Date)
```

CAST

Description:

This function extracts a type from an expression that can contain a composite type. If the expression contains a type other than the required one, it returns `NULL`.

Parameters:

- Converted expression.
- `Type` - a string, containing a type string. For example, `Number`, `String`, and so on. In addition to primitive types, the string can also

contain a table name. In this case the function attempts to cast to a reference to that table.

Example:

```
CAST(Data.Attribute1, "Number(10,3)")
```

ISNULL

Description:

This function returns the second parameter value if the first parameter is `NULL`. Otherwise it returns the first parameter value.

Example:

```
ISNULL(Sum(Sales.AmountTurnover), 0)
```

PRESENTATION

Syntax:

`Presentation(<Expression>)`

Description:

This function returns the string presentation of a passed value that has non-primitive type. For primitive type values, the value itself is returned.

If an array is passed as a parameter, the function returns a string containing string presentations of all the array elements, separated by ";". If a value table is passed as a parameter, the function returns a string containing the string presentations of all the value table rows, where the presentations of the cells of each row are separated by ";", and the rows - are separated by the line break character. If any element has an empty string presentation then the `<Empty value>` string is output instead of its presentation.

Example:

```
Presentation(Counterparty)
```

STRING

Syntax:

`String(<Presentation>)`

Description:

If an array is passed as a parameter, the function returns a string containing string presentations of all the array elements, separated by ";". If a value table is passed as a parameter, the function returns a string containing the string presentations of all the value table rows, where the presentations of the cells

of each row are separated by ";", and the rows - are separated by the line break character. If any element has an empty string presentation then the <Empty value> string is output instead of its presentation.

Example:

```
String(SaleDate)
```

ACOS

Syntax:

```
ACos(<X>)
```

Description:

Calculates the arc cosine value.

ASIN

Syntax:

```
ASin(<X>)
```

Description:

Calculates the arc sine value.

ATAN

Syntax:

```
ATan(<X>)
```

Description:

Calculates the arc tangent value.

COS

Syntax:

```
Cos(<X>)
```

Description:

Calculates the cosine value.

SIN

Syntax:

```
Sin(<X>)
```

Description:

Calculates the sine value.

TAN

Syntax:

```
Tan(<X>)
```

Description:

Calculates the tangent value.

EXP

Syntax:

```
Exp(<X>)
```

Description:

Calculates `e` to the power of `X`.

LOG

Syntax:

```
Log(<X>)
```

Description:

Calculates the base logarithm value.

LOG10

Syntax:

```
Log10(<X>)
```

Description:

Calculates the decimal logarithm value.

POW

Syntax:

```
Pow(<X>, <Y>)
```

Description:

Calculates the value of `X` to the power of `Y`.

SQRT

Syntax:

```
Sqrt(<X>)
```

Description:

Calculates the square root value.

ROUND

Syntax:

`Round(<Expression>, <DecimalsNumber>)`

Description:

Rounds the `Expression` value to `DecimalsNumber`.

INT

Syntax:

`Int(<Expression>)`

Description:

Calculates the integer part of the passed number, completely cutting off the fractional part.

VALUETYPE

Syntax:

`ValueType(<Expression>)`

Description:

Calculates the type of expression passed as a parameter.

Returns:

`Type` type value.

10.3.4.12. Common module functions

An expression of the data composition engine can contain calls to the functions of global common configuration modules and non-global common modules with the Client (ordinary application) (when using an ordinary application) or Server (when using a managed application) property set. No additional syntax is required to call such functions.

```
BriefDescription(Doc.Reference, Doc.Date, Doc.Number)
```

In this example the `BriefDescription()` function will be called from the common configuration module.

Note that you can only use common module functions if you specify the corresponding data composition processor parameter.

Common module functions cannot be used in custom field expressions.

10.3.5. Data composition schema wizard

Data composition schema wizard is the 1C:Enterprise language object `DataCompositionSchemaWizard`, designed to visually create the data composition schema. In addition, the data composition schema wizard is used in the configurator when editing the data composition schema.
For query - data sets the wizard automatically retrieves nested data sets from the query text and handles them as data set fields - nested data set.
For object - data sets the wizard allows to add fileds - nested data sets.

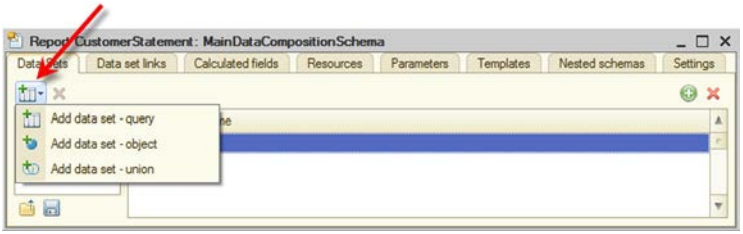


Fig. 210. Adding new data set

Below is an example of opening the data composition schema wizard window and subsequent serialization of the resulting composition schema in XML.

```
Procedure ReportEditorCommandBar (Button)

  Wizard = New DataCompositionSchemaWizard;

  Wizard .SetSchema(GetDataComposiotionSchema());
  Wizard.Edit(ThisObject);

EndProcedure

Procedure ChoiceProcessing(SelectionValue, Source)

  If ValueType(Source) = TypeOf("DataCompositionSchemaWizard") Then

    DataCompositionSchema = Source.GetSchema();
    WriteXML = New WriteXML;
    WriteXML.SetString();
    XDTOSerializer.WriteXML(WriteXML, DataCompositionSchema,
"dataCompositionScheme", "http://v8.1c.ru/8/data composition
system/scheme");
    Controls.DataCompositionSchemaText. SetText(WriteXML.Close());
  EndIf;

EndProcedure
```

Operation with the data composition schema is divided into the following steps:

- data sets editing,
- data sets fields editing,
- links of data sets editing,
- calculated fields editing,
- resources editing,
- parameters editing,
- templates editing,
- nested settings editing,
- data composition system editing.

10.3.5.1. Edit data sets

The data composition system enables editing of the following objects:

- data set - request;
- data set - object;
- data set - merge;
- query fields.

When you add a data set, the application automatically generates a name and data source (if the data source does not exist).

The data composition schema entry field - nested data set can be described in the query, as well as in the data set - object.

В тексте запроса набора данных - объекта в предложениях `{ВЫБРАТЬ}` и `{ГДЕ}` можно использовать вложенные таблицы.

```
SELECT
 GoodsReceipt.Date,
 GoodsReceipt.Number,
 GoodsReceipt.Vendor,
 GoodsReceipt.Warehouse,
 GoodsReceipt.Goods.(
 LineNumber,
 Goods,
 Price,
 Quantity,
 Sum
 )
{SELECT
 Date
 Number,
 Vendor.*,
 Warehouse.*,
 Goods.(
  LineNumber,
  Article.*,
  Price,
  Quantity,
  Sum
  )}

FROM
 Document.GoodsReceipt AS GoodsReceipt
{WHERE
 GoodsReceipt.Date,
 GoodsReceipt.Number,
 GoodsReceipt.Vendor.*,
 GoodsReceipt.Warehouse.*,
 GoodsReceipt.Goods.(
  LineNumber,
  Article.*,
  Price,
  Quantity,
  Sum
  )}
```

In the example above, query fragments describing the fields of nested tables available for settings are shown in bold.

10.3.5.1.1. Edit Dataset - Query

Edit data set - query consists in creating a query to the data. To do this, you can use the query builder or edit the query text directly in the Data Composition Schema Wizard window.

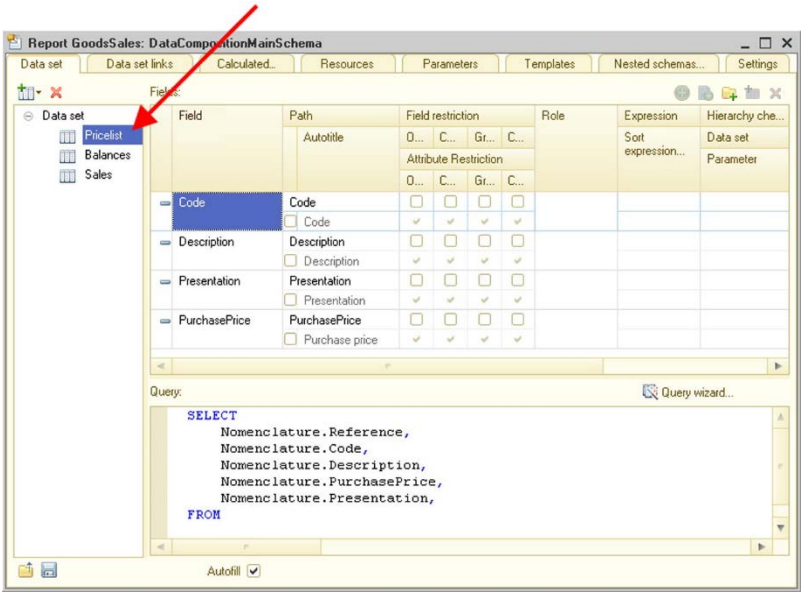


Fig. 211. Data set - query

If the AutoFill check box is selected, the data composition system automatically fills in the data composition schema fields based on the query.

10.3.5.1.2. Edit Data set - Object

Edit data set - object consists of adding/editing the name of the object containing the data; adding/editing fields and groupings.

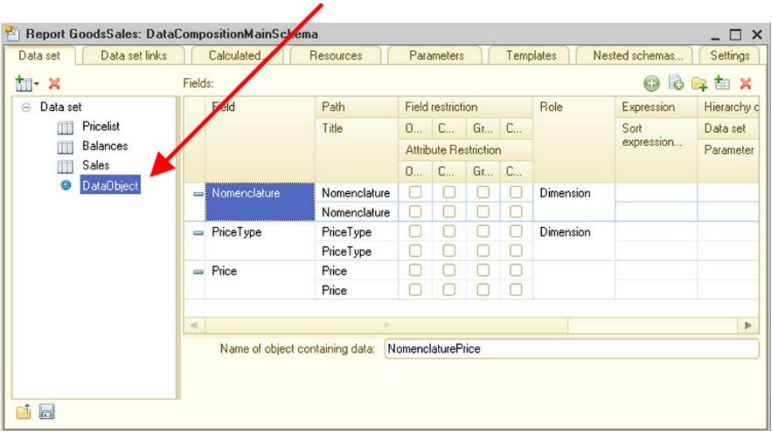


Fig. 212. Data set - object

10.3.5.1.3. Edit Dataset - Merge

Edit data set - merge consists of editing the merged fields from the fields that are subordinate to the data set. The created data sets - queries and objects can be added to the data set - merge by moving them with the mouse pointer.

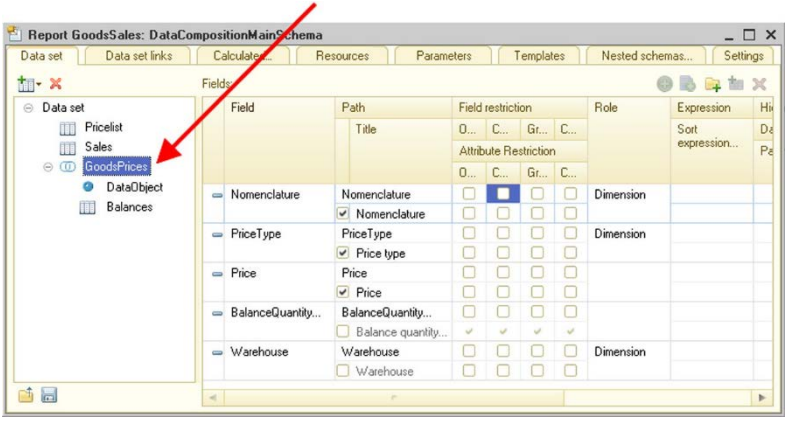


Fig. 213. Data set - merging

10.3.5.2. Edit Data sets fields

When editing fields, you can specify:

- field header;
- field availability restriction;
- attribute fields availability restriction;
- role for the field;
- field view;
- order expressions;
- hierarchy validation method (data set and parameter);
- field value type;
- field appearance.

Note that the field role, view, order expression, parameter, value type, and field appearance are edited only at the top level of the dataset hierarchy.

If the data set field does not have a header set, the data composition system attempts to generate a field header based on the synonym of the query field. If there is no synonym for the field, the field data path is used as the field header, supplemented with spaces using the algorithm to convert names to synonyms. Synonyms for query fields are obtained only for fields that either have no alias set or are not different from the default alias for this field.

Synonyms for the query field are obtained as follows.

- If the query does not contain combinations:
  - If the query field expression consists of a single field, the synonym is obtained from that field.
  - If a field expression is an aggregate function over a single field, the synonym for the field is derived from this field. For other expressions, it is considered that a synonym cannot be obtained.
- If the query does not contain combinations:
  - In all combinations, a field is searched for, which can be synonymous with the field used as a synonym. If no such field is found, the synonym cannot be obtained.

In the Data Composition Schema Wizard, there is an additional column with a check box indicating that the field header is set manually. The check box is automatically cleared if the field does not have a header set.

When filling fields based on a query, the Wizard automatically fills in the headers for those fields that cannot be synonymous and for those fields that have a different alias in the query than alias of this field by default. Thus, the check box will be automatically selected for such fields.

If this check box is cleared, the Title column displays, which will be shown to the user. At the same time, the header in the field is not filled. The Header

column for fields that have the check box cleared is not edited and is displayed in the disabled text color.
If the developer needs to change the header, he/she sets a check box, and the field header text is filled in based on the text that was previously displayed in the Header column. If the check box is cleared, the Wizard clears the header text and starts displaying the automatically generated header in the header.
The field role is initially defined in the query, but it is possible to change it in a separate dialog box.

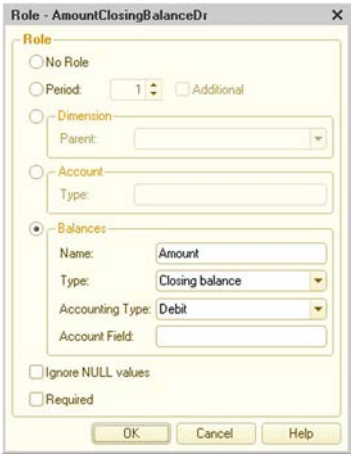


Fig. 214. Field role

For the count field, explicitly specify a reference to the count type in the View parameter.
For the dimension field, specify the path to the parent dimension data in the Dimension parameter.
The Ignore NULL option indicates that group records for this field will not be included in the result if it contains NULL.
In the resulting dataset, fields with the Required check box will always be present if at least one of its data sets is used in the settings. For example, it is necessary to obtain the unfolded balances for the extra dimensions. In this case, if the Extra dimension field is not used in the query, the balances will be minimized.
Field ordering expressions, value type, and appearance can also be edited in a separate dialog.

10.3.5.3. Edit Dataset Links

In the case of multiple top-level datasets, it is possible to set up relationship between them by one or more fields.

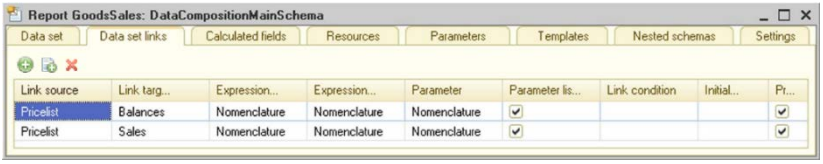


Fig. 215. Data set links

Data sets are the source and receiver of the link. The source and receiver expressions - dataset fields.

10.3.5.4. Edit Calculated Fields

The Calculated Fields tab allows to create and edit the following properties/characteristics of calculated fields:

- data path;
- expression;
- header;
- availability restriction;
- presentation expression;
- order expressions;
- value type;
- appearance;
- available values.

Order expressions are edited in a separate dialog box.

10.3.5.5. Edit Resources

Resources can be calculated across all fields of all datasets and calculated fields. The left tabular field shows a list of available and unused fields. The right tabular field displays the fields, by which the results and expressions of their calculation will be formed.

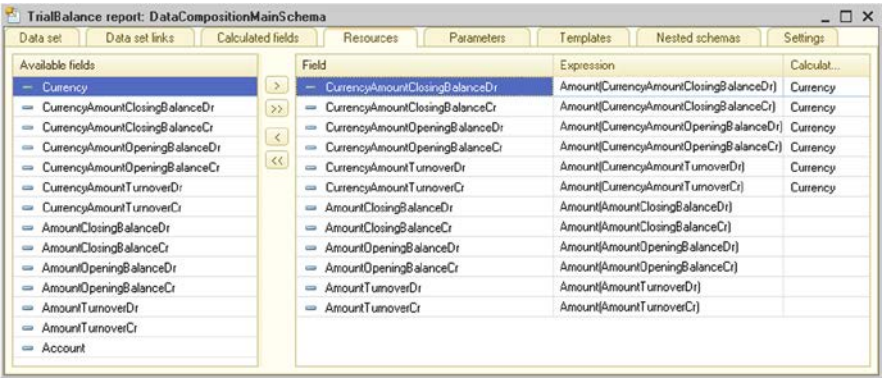


Fig. 216. Edit Resources

By default, the `Sum` function is set for numeric fields, and for non-numeric fields - `Quantity` function. Use the `>>` button to add all fields of the `Number` type to the resources. Note that you can add multiple lines for a single resource. When the template composer gets an expression for a resource, it checks the target grouping and generates the expression for that grouping.

In the Edit dialog box of the Calculate by column, strings with the field name and the `Hierarchy` keyword are added for fields that can be hierarchically grouped. The expression will be used for hierarchical grouping records by the field specified before the word. When you select a grouping broken don by a resource can be calculated, you can select both a normal and a hierarchical field at the same time.

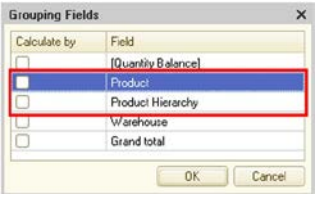


Fig. 217. Calculation of resources by grouping fields

If you have set the resource to be calculated only for a certain grouping (i.e. you have selected at least one grouping field in the Calculate by column), this resource is output to the results only for this grouping and its nested groupings.

10.3.5.6. Edit parameters

Edit Parameters includes:

- parameter name editing;
- header editing;
- editing the parameter available types and values;
- defining the value and availability of the parameter values list;
- defining the expression;
- defining the parameter as the available data composition setting field;
- availability restriction;
- defining whether the parameter is required;
- flag of parameter use;
- specifying the editing options.

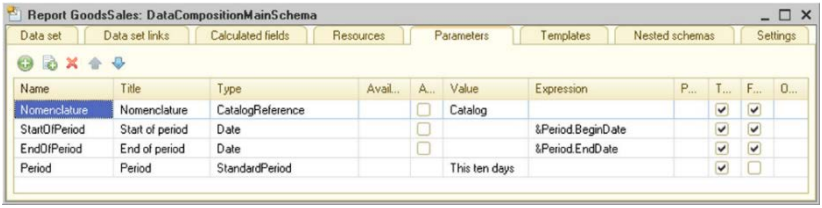


Fig. 218. Edit parameters

If the parameter value is undefined, it is assumed that the value - is zero reference to the specified type. Parameters can use predefined data and enumerations (Designer mode).

In the Available Values column, the values that can be selected by the user as parameters of the data composition schema are edited. If the check box in the Values List Available column is selected, it means that several values of the parameter can be used.

NOTE. When using the `StandardPeriod` parameter type, note that the start and end dates of the standard period also contain time. And the starting date has the time 00:00:00, and the end date – 23:59:59. Thus, it is not necessary to use the `BEGINOFPERIOD` and `ENDOFPERIOD` functions in the query.

Here is an example of using the mandatory parameter of the data composition schema: for example, ypu need to specify the Organization parameter in the report. In the data composition schema, for the Organization parameter the Usage property is set to Always and the Forbid blank values property - to True.

While operation with the report, the Usage property is unavailable to the user (it is absent), and the empty value is displayed in the usual way (wavy red line). If the user does not fill in the parameter value and runs the report for execution, he will receive a message that the parameter value should be entered.

10.3.5.7. Edit Templates

Adding templates is performed by clicking the Add Template button on the command bar.

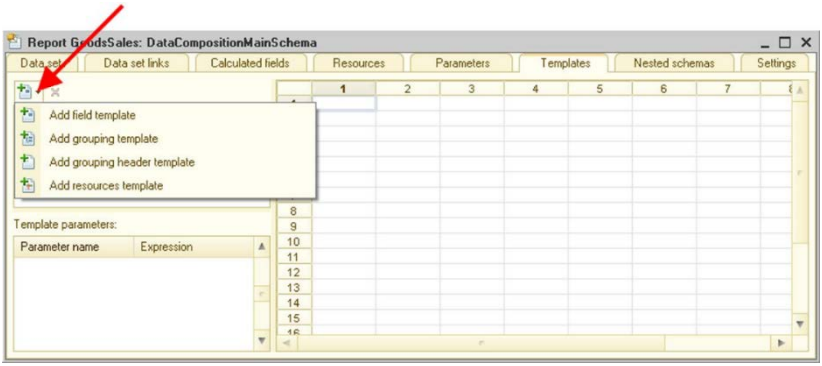


Fig. 219. Template Editor

It is suggested that you choose one of the template options:

- Field Template,
- Grouping Template,
- Grouping Header Template,
- Table Resources Template.

For the grouping template and grouping header, you can specify the grouping name or grouping fields, template type.

For the table resource template, you can specify templates for two groupings, at the intersection of which it is located.

Column of the tabular field Area indicates the coordinates of the template area in the spreadsheet document.

Editing areas of a spreadsheet document can be done using the properties panel called by Alt + Enter. Both the appearance and contents of the cell and the decryption parameter for it are edited.
The template can be edited in all languages supported by the application. When you specify a parameter or template to a spreadsheet document cell, the parameters are added to the template and are displayed in the Parameter Name column of the Teamplate Parameters table field. You can edit the template parameter expression. When loading, template areas are separated by empty lines.

10.3.5.8. Nested schemas

The Nested Schemas tab allows to create and edit nested data composition schemas. The schema can be nested schemas edited by the Data Composition Schema Wizard.

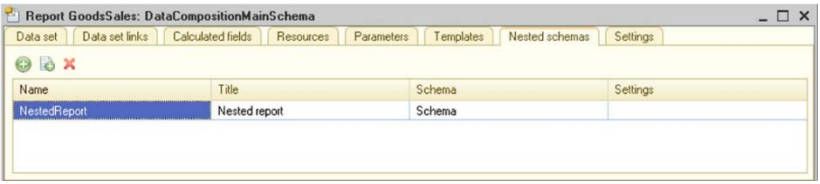


Fig. 220. Nested schemas

10.3.5.9. Settings

The data composition schema contains the default data composition settings that can be specified by the developer.

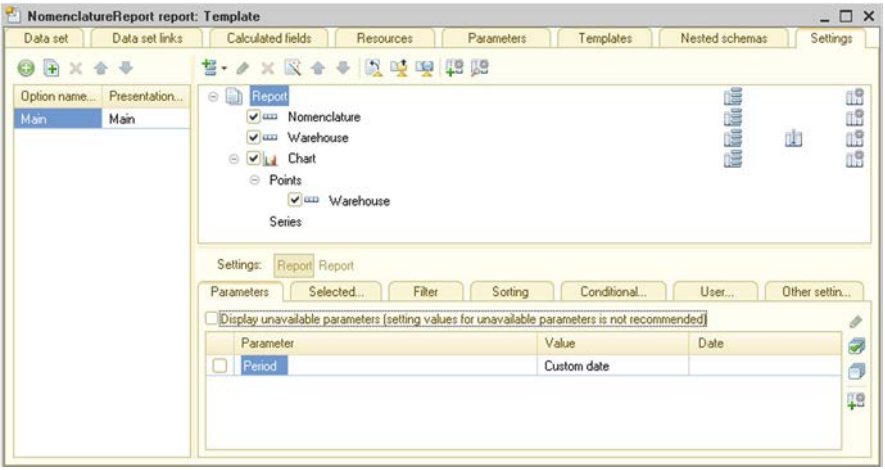


Fig. 221. Data Composition Schema Settings Editor

Setting up the Auto Resource Position report controls how resource fields will be displayed:

- Do not use - in this case, the resources are displayed in the order, in which the fields on the Selected Fields tab are listed.
- After all fields - in this case, the resources are displayed after all fields.

For grouping (or grouping a table), there is a Grouping Usage Option parameter that controls the display of additional information after resource fields. If this parameter is not set or is set to Auto, the grouping is responsible for displaying the detailed records.

If the parameter is set to Additional Information and is specified in the column grouping, one column will be displayed in the resulting table in this column, the fields specified on the Selected fields of this grouping and available for display in the grouping-line. If the group-column grouping fields are specified, an error will be generated when the tepmplate is assembled. For grouping-line - similar, only one line is displayed, not a column.

For example, an Item is displayed in the table in lines. The columns of the table show the warehouses. A report on the balances of goods in warehouses is generated. It is necessary to display the item number after the columns with the balance. To do this, add grouping without group fields to the columns of

the table and specify the Grouping Usage Option to the Additional Information value in the grouping settings. On the Selected Fields tab specify the Item.Article field.
When this grouping is displayed outside the table, all fields that are available in the grouping are displayed.
For example, if a report contains a grouping. in which multiple groups are nested. Using grouping with the Grouping Usage Option parameter set to Additional Information, the parent grouping data between groups can be displayed.
For groups with the option to use Additional Information, the following parameters are ignored

- Location of grouping fields - the headers of the fields displayed in the grouping are always displayed at the beginning of the row or column.
- Number of records.
- Percentage of records.
- Results Layout.
- Grouping Fields Layout.
- Groupings Layout.
- Overall Results Layout.

When a template composition is performed, if in the grouping with the use option Additional Information there is a field that is not available in a grouping or table, an exception is issued.

10.3.5.9.1. Nested fields

Numeric fields

The platform automatically generates nested fields for numeric resources. These fields are calculated automatically and facilitate data retrieval. For example, you can use them to get the percentage ratio of a resource value to the resource value total in all the report data. Let's enumerate these fields.

% in hierarchy group

ID:

`PercentInHierarchy`

Description:

This field contains the percentage of a resource value in the current hierarchy group. It is 100% in a nontable output.

% in column or point hierarchy group

ID:

`PercentInColumnOrPointHierarchy`

Description:

This field contains the percentage of the resource value in the current cell to the value of the resource total at the current hierarchy level of the current grouping for a column or for a point. In a non-table output the field value is equal to the % in hierarchy group field value.

% in row or series grouping

ID:

`PercentInRowOrSeriesHierarchy`

Description:

This field contains the percentage of the resource value in the current cell to the value of the resource total at the current hierarchy level of the current grouping for a row or for a series. It is 100% in a nontable output.

% in grouping

ID:

`GroupPercent`

Description:

This field contains the percentage of a resource value in the current cell to the value of the resource total in the current grouping. It is 100% in a nontable output.

% in column or point grouping

ID:

`ColumnOrPointGroupPercent`

Description:

This field contains the percentage of a resource value in the current cell to the value of the resource total in the current group for a column or a point. It is 100% in a nontable output.

% in row or series grouping

ID:

`RowOrSeriesGroupPercent`

Description:

This field contains the percentage of a resource value in the current cell to the value of the resource total in the current group for a row or a series. It is % in the grouping field in a nontable output.

% in column or point

ID:

`ColumnOrPointPercent`

Description:

The percentage ratio of a resource value in the current cell to the value of the resource total for a column or a point. It is % total in a nontable output.

% in row or series

ID:

`RowOrSeriesPercent`

Description:

The percentage ratio of a resource value in the current cell to the value of the resource total for a row or a series. It is 100% in a nontable output.

% total

ID:

`OverallPercent`

Description:

This field contains the percentage of a resource value in the current cell to the value of the overall resource total in the table. It is 100% in a nontable output.

For fields that contain the type Date as part of their types

The platform automatically generates nested fields for fields that contain `Date` type inside their types. These fields are calculated by the platform,

which simplifies getting date- and time-related data. For convenience, these fields are grouped into three groups: Даты начала (ДатыНачала, BeginDates), Даты конца (ДатыКонца, EndDates), Части дат (ЧастиДат, DateParts). Let's enumerate these fields.

"Beginning dates" group

Beginning of day

ID:

BegOfDay
BegOfDay

Description:

Date and time of the beginning of the day.

Beginning of week

ID:

BegOfWeek
BegOfWeek

Description:

The start date and time of the week, considering the First day of the week property of the regional settings of the infobase.

Beginning of ten days

ID:

BegOfTenDays

Description:

Date and time of the beginning of ten days.

Beginning of month

ID:

BegOfMonth
BegOfMonth

Description:

Date and time of the beginning of the month.

Beginning of quarter

ID:

`BegOfQuarter`
`BegOfQuarter`

Description:

Date and time of the beginning of the quarter.

Beginning of half year

ID:

`BegOfHalfYear`

Description:

Date and time of the beginning of the half year.

Beginning of the Year

ID:

`BegOfYear`
`BegOfYear`

Description:

Date and time of the beginning of the half year.

Beginning of minute

ID:

`BegOfMinute`
`BegOfMinute`

Description:

The date and time of the beginning of the minute (if the date includes time).

Beginning of hour

ID:

`BegOfHour`
`BegOfHour`

Description:

The date and time of the beginning of the hour (if the date includes time).

"End dates" group

End Of Day

ID:

EndOfDay
EndOfDay

Description:

Date and time of the end of the day.

End Of Week

ID:

EndOfWeek
EndOfWeek

Description:

The start date and time of the week, considering the First day of the week property of the regional settings of the infobase.

End Of Ten Days

ID:

EndOfTenDays

Description:

Date and time of the end of ten days.

End Of Month

ID:

EndOfMonth
EndOfMonth

Description:

Date and time of the end of the month.

End Of Quarter

ID:

EndOfQuarter
EndOfQuarter

Description:

Date and time of the end of the quarter.

End Of Half Year

ID:

`EndOfHalfYear`

Description:

Date and time of the end of the half year.

End Of Year

ID:

`EndOfYear`
`EndOfYear`

Description:

Date and time of the end of the year.

End Of Minute

ID:

`EndOfMinute`
`EndOfMinute`

Description:

The date and time of the end of the minute (if the date includes time).

End Of Hour

ID:

`EndOfHour`
`EndOfHour`

Description:

The date and time of the end of the hour (if the date includes time).

"Date parts" group

Day

ID:

`Day`

Description:

Day of the month (number).

Day of the week

ID:

`WeekDay`
`WeekDay`

Description:

The number of the day of the week (as the `WeekDay()` function returns, the week always starts on Monday).

Day of the week name

ID:

`WeekDayName`

Description:

Presentation of the day of the week (1 - Monday, 2 - Tuesday, etc.). The day number is obtained by using the `WeekDay()` function.

Day of the year

ID:

`DayOfYear`
`DayOfYear`

Description:

Number for the day of the year.

Week of the year

ID:

`WeekOfYear`
`WeekOfYear`

Description:

The day of the week in the year, considering the First day of the week property of the regional settings of the infobase.

Month

ID:

`Month`
`Month`

Description:

Number of the month of the year.

Month name

ID:

MonthName

Description:

Presentation of the name of the month (1 - January, 2 - February, etc.).

Quarter

ID:

Quarter

Description:

Number of the quarter.

Year

ID:

Year

Description:

Year value in 4-digit view (number).

Minute

ID:

Minute
Minute

Description:

Number of the minute.

Hour

ID:

Hour
Hour

Description:

Number of the hour.

10.3.5.9.2. System fields

In the list of selected fields, the platform generates a dedicated group of system fields (`SystemFileds`) that are intended to define a sequential number of a record both in the report as a whole and within the groupings. Remember that system fields are not included in the list of fields that are obtained by expanding an auto field, and therefore require manual adding.

`Ser. #`

ID:

`SystemFields.SerialNumber`

Description:

Contains the sequential number of the row in the report. Numbering starts with 1.

`# in group`

ID:

`SystemFields.GroupSerialNumber`

Description:

The sequential row number in the current grouping. Numbering starts with 1.

`Level`

ID:

`SystemFields.Level`

Description:

The current record level. Numbering starts with 1.

`Level in group`

ID:

`SystemFields.LevelInGroup`

Description:

The current record level in relation to the grouping. Numbering starts with 1.

10.3.5.9.3. Parameter fields

The selected fields include a dedicated group named Parameters, which is intended to include data composition schemas with the Include in Available Fields check box selected in the report.

10.3.6. Data composition schema option settings

In the data composition schema, you can define multiple settings. **Setting option** - is a set of report settings that the developer considered necessary to select separately. These settings are stored in the data composition schema.
For example, for the report Dynamics of sales one report option can be a diagram showing sales of goods by periods, and another - tabular report showing sales of goods broken down by buyers. Each report option has its own set of user settings.
When you use a data composition schema for a report, the settings described in the schema are provided to the user as standard report options.
You can create a custom report option in 1C:Enterprise mode. This action is recommended for experienced users. In this case, the new report option is stored in the Report Option repository, from which other users can download the desired option.
Use the Configuration Comparison and Merge mechanism to compare and merge parts of the settings.
Loading the data composition scheme from XML can be performed using standard 1C:Enterprise language tools.

10.3.6.1. Data Composition Setting Option Structure

Structure - is a skeleton of settings. It determines the mutual arrangement of their main elements.
The settings structure is available through the `Structure` property of the `DataCompositionSettings` object. The elements of the setting structure can be:

- groupings;
- tables (`DataCompositionTable`);
- charts (`DataCompositionChart`);
- nested setting objects (`SettingsOfNestedObjectOfDataCompositionSystem`).

Grouping

There are three different types of data in the settings structure to implement the grouping:

- groupings (`DataCompositionGroup`);
- tables groupings (`DataCompositionTableGroup`);
- charts groupings (`DataCompositionChartGroup`).

Presence of three types is connected with necessity to implement the restrictions imposed on the mutual arrangement of elements in the structure tree: tables and charts cannot include anything but groupings. Accordingly, all grouping objects have the same object model, they differ in the type of nested collection of values and the composition of output parameters.

Grouping fields

The set of fields, by which the grouping is performed, is described with the help of the `GroupingFieldsOfDataCompositionSystem` object. The `Element`s property of this object contains a collection of grouping fields consisting of the `GroupingFieldOfDataCompositionSystem` objects.

NOTE. When performing the grouping by field-period, the parent field-period is automatically added to the grouping, which is not an additional period, if the parent groups have not grouped by this parent field-period.

For example, if grouping is performed in the `Recorder` field, the `PeriodSecond` field is automatically added to the grouping.

At the same time, the creation of groups according to the attributes of field-periods is prohibited.

Grouped auto fields

Before using the autofield will be converted to a set of grouping fields.

The set is formed as follows. The selected fields with the following conditions are taken:

- they are available for use in grouping fields;
- they are not resources;
- they are not attributes of other selected fields;
- they do not depend on already existing grouping fields.

If a field is already included in the data of the grouping field, it is not added again.

Tables

Description of the table in the structure of settings is performed with the help of the `TableOfDataCompositionSystem`.

Chart

Description of the chart in the structure of settings is performed with the help of the `ChartOfDataCompositionSystem`.

Nested Object

Description of the nested object in the structure of settings is performed with the help of the `SettingsOfNestedObjectOfOfDataCompositionSystem`.
The object has the `Name` property used to identify the nested report in the generated data composition template.

10.3.6.2. Data composition settings properties

Case

A set of fields that are output to composition results. It is described with the help of `DataCompositionSelectedFields` object. The `Elements` property of this object contains a collection of selected fields consisting of the `DataCompositionSelectedField` objects.

Selected fields group

Used for fields grouping. It is described with the help of `DataCompositionSelectedFieldGroup` object.

Autoselected field

Before using the autofield will be converted to a set of selected fields. The assortment of fields in a set depends on the structure item where the auto field belongs and on the structure part where the item is located. For each item, 1C:Enterprise iterates through all the parent items of the report structure and selects the resources and fields from the selected fields of these items using the following rules:

- For grouping and grouping tables the autofields are substituted by:
  - all used fields of this grouping, which are available for use in the selected fields;
  - the fields that are the attributes of its grouping fields;
  - resources of parent items.

IMPORTANT! The application consideres only those groupings that are of No Hierarchy or Hierarchy type when iterating.

- For a chart grouping, resources are not selected. Instead, 1C:Enterprise iterates through all the parent items of the report structure and selects the grouping fields from the selected fields of these items, provided that this grouping has Hierarchy Only type.
- For groupings of Detailed Records (grouping, table grouping, or chart grouping) type, all the used fields except for the fields that are included in higher-level groupings and attributes of those fields are selected from the main selected fields of the setting where the grouping belongs. If the grouping has the Hierarchy Only type, its fields and attributes are used in the generation of a selection field set. For a chart grouping, resources are not selected.
- For the Detailed information type groupings, the selected fields contain the fields - nonresources that are available in the grouping or parent grouping, but also in the table - in a grouping at the opposite axis.
- For the chart, the autoselected field is replaced by all the parent resources.
- For a table, an autoselected field is converted to a set of resources used by the parent items.

NOTE. If a field is already included in the selected fields, it is not added again.

The fields are added to a set as follows: first, the fields of the grouping's fields (for groupings); next, fields of the global settings (for Detailed Records groupings); and finally - the resources and fields of the parent items.

Filters

Used to filter the records that fall into the result of the composition. It can also be used to select records, to which some formatting (conditional appearance) is applied, and to create user selection fields.

It is described with the help of `DataCompositionFilter` object. The `Elements` property of this object contains a collection of grouping fields consisting of the `DataCompositionFilterItem` objects.

Selection Elements Group

Used for the selection elements grouping, which arranges the result data. It is described with the help of `DataCompositionFilterItemGroup` object.

Order

Describes how to organize the records displayed to the result. It is a `DataCompositionOrder` object. The `Elements` property of this object contains a collection of order items consisting of the `DataCompositionOrderItem` objects.

Order Auto Element

Before use, the order automatic element will be converted to a set of order elements.

The following rules apply to the formation of the set: resources are added unconditionally, and fields that are the attributes of the grouping field are added in order from the non-resource fields, and the grouping field itself (all fields will be entered for detailed records). The grouping fields that are not specified in the global sorting are added to the end of the list. If a field is already included in theis order, it is not added again.

Using conditional appearance in reports

Describes how to arrange different result fields. It is a `DataCompositionConditionalAppearance` object. The `Elements` property of this object contains a collection of order items consisting of the `DataCompositionConditionalAppearanceItem` objects.

In the data composition appearance of the Data Composition Wizard, the `Format`, `Text` options are edited as multilingual.

For a conditional appearance element, scope of application can be defined. Conditional appearance can be used as follows:

- In grouping;
- In hierarchical grouping;
- In overall;
- In fields header;
- In resource fields header;
- In overall header;
- In overall resource fields header;
- In header;
- In parameters;
- In filter.

Formatted fields

Fields, to which the formatting is applied. They are described with the help of `DataCompositionAppearanceFields` object. The `Elements`

property of this object contains a collection of grouping fields consisting of the DataCompositionAppearanceField objects. If no fields are specified, the appearance will be applied to the entire area.

Output Parameters

The values of the output parameters determine the appearance of the corresponding objects. Inheritance is supported for some parameters. Therefore, the collection of output parameters for an element can contain parameters that are not related to it itself, but are used in elements that can be inserted into the subordinate part of the settings structure.

Data Parameters

Data parameter values are usually used in queries to filter the selection.

User-defined fields

User-defined fields allow to expand the set of available fields by defining user own expressions or sets of options with conditions of using a particular option.

User-defined fields are described using the DataCompositionUserFields object. The Elements property of this object contains a collection of user-defined fields consisting of two types of objects:

- Field-Expression (DataCompositionUserFieldExpression object);
- Field-Selection (DataCompositionUserFieldCase object).

The field type is determined automatically by the application based on its properties.

User-Defined Field Options

Description of a set of alternatives defining the value of the selection field. They are described with the help of DataCompositionUserFieldsCaseVariants object. The Elements property of this object contains a collection of user-defined selection field consisting of the DataCompositionUserFieldsVariant objects.

Managing autofields

If one element of the settings structure contains AutoFieldOfGroupingOfDataCompositionSystem, AutoSelectedFieldOfDataCompositionSystem, and OrderAutoElementOfDataCompositionSystem, they are converted in the following order:

- AutoFieldOfGroupingOfDataCompositionSystem,
- AutoSelectedFieldOfDataCompositionSystem,
- AutoOrderElementOfDataCompositionSystem.

10.3.6.3. Available Objects

Available objects - are a set that defines objects that can be used as nested in the composition. For example, nested reports.

10.3.6.4. Available fields

Available fields - are many fields that can be used in setting the data composition and will be recognized and properly processed in subsequent stages of the composition. The available fields vary in use. There are the following field collections:

- selection fields (`SelectionAvailableFields` property);
- grouping fields (`GroupAvailableFields` property);
- order fields (`OrderAvailableFields` property);
- data parameter fields (`DataParametersAvailableFields` property);
- Selection fields (`FilterAvailableFields` property);
- selection fields of structure elements - are used in all structure elements except the top (`StructureItemsFilterAvailableFields` property).
- additional selection fields - are used in the conditional appearance (`AdditionalFilterAvailableFields` property).

All listed properties contain collections of values, whose elements are the `DataCompositionAvailableField` objects.

In the Data Composition Wizard settings and in the Report Data Composition Wizard settings the available fields are arranged in the following order: first, non-resource fields in alphabetical order by heading, followed by resource fields in alphabetical order by heading. System folders are the last in the list.

Available Selection Field

A special type of available fields is implemented for use in selections. It has all the properties of a regular available field, as well as provides a set of available comparison views and available field values required to correctly construct the selection elements.

10.3.6.5. Data Composition Settings Composer

The settings Composer is represented as 1C:Enterprise language object `DataCompositionSettingsComposer`. The object is used to link

data composition settings and data composition schema. Based on the data composition schema, the source of available settings for the settings Composer is built.
If it is necessary to use the source of available settings of one Composer in another, there is an option to get the source of available settings using the `GetAvailableSettingsSource()` method of the `DataCompositionSettingsComposer` object.

Example of transferring the source of available data composition settings:

```
Variable FormParameters;
FormParameters = New Structure;
FormParameters.Insert("SourceOfAccessibleSettings",
List.SettingsComposer.GetAvailableSettingsSource());
FormParameters.Insert("Settings", List.SettingsComposer.Settings);
FormParameters.Insert("FixedSettings",
List.SettingsComposer.FixedSettings);
FormParameters.Insert("UserSettings",
List.SettingsComposer.UserSettings);
OpenForm("SharedForm.SettingsComposerForm", FormParameters);
```

10.3.7. User Settings of Data Composition System

It is possible to select some settings so that the user can edit them in a separate form. This mechanism is called **User Settings**.
Both the userand full settings are used for the composition. At the same time, user settings are “superimposed” on full ones, generating the actual executable settings.

10.3.7.1. User Settings Object Model

The following objects are edited using the user settings:

- DataCompositionFilter,
- DataCompositionFilterItem,
- DataCompositionFilterItemGroup,
- DataCompositionOrder,
- DataCompositionSelectedFields,
- DataCompositionConditionalAppearance,
- DataCompositionConditionalAppearanceItem,
- DataCompositionSettingsParameterValue,
- DataCompositionGroup,
- DataCompositionTableGroup,

- DataCompositionChartGroup,
- DataCompositionTable,
- DataCompositionChart,
- NestedDataCompositionSchema,
- ElementsCollectionOfDataCompositionStructure,
- DataCompositionTableStructureItemCollection,
- DataCompositionChartStructureItemCollection.

These objects have the following properties:

- UserSettingID - is used to identify the user settings object. If this property is set, the object is considered user and is subject to editing in user settings.
  When an interactive setting is checked as a user one the application automatically generates a UUID and fills this property with a string representation of that ID.
- UserSettingPresentation - string used to display the view in user settings. In the Schema Wizard, you can enter a view in multiple languages.
- ViewMode - is used to define quick settings. More information about this feature will be provided below.

User settings in the object model are represented as a separate object DataCompositionUserSettings. This object has the Elements property. This collection contains elements of user settings. The following types are allowed:

- DataCompositionFilter,
- DataCompositionFilterItem,
- DataCompositionFilterItemGroup,
- DataCompositionOrder,
- DataCompositionSelectedFields,
- DataCompositionConditionalAppearance,
- DataCompositionConditionalAppearanceItem,
- DataCompositionParameterValue,
- DataCompositionGroup,
- DataCompositionTableGroup,
- DataCompositionChartGroup,
- DataCompositionTable,
- DataCompositionChart,
- DataCompositionNestedObjectSettings,
- DataCompositionSettingStructure.

10.3.7.2. Configuring User Settings Elements

You can set the user attribute of a customization element in the user element configuration form, which is called by the Element Properties of User Settings command.

Fig. 222. Open User Setting

In the user element setup form, you can also specify the view that will be used for the element, and how it is edited.

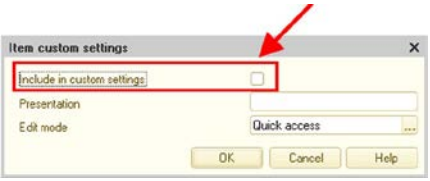


Fig. 223. User Setting Property

In the settings structure list, you can use the Element Properties of User Settings command to set up the user settings for the current structure item. Each structure item has its own assortment of customizable items.

Object	Customizable items
Report	• selected fields, • order, • filter, • conditional appearance, • components of groupings
Grouping/ table grouping/ chart grouping	• grouping, • selected fields, • filter, • order,

Object	Customizable items
	• conditional appearance, • components of nested groupings
Chart	• chart, • selected fields, • conditional appearance, • components of series groupings, • components of points groupings
Tables	• table, • selected fields, • conditional appearance, • components of rows groupings, • components of columns groupings
Nested schema	• nested report, • selected fields, • filter, • order, • conditional appearance, • components of groupings

Use the User-defined settings item `Properties` command to change the following settings:

- In a filter list - user settings for the current filter item/filter item groups;
- in a list of output parameters and data parameters - user settings for the current parameter;
- in a conditional appearance list - user settings for the current conditional appearance item.

Besides, the User Settings command available on the table structure opens a form that displays the user settings along with their default values.

10.3.7.3. Edit User Settings

The `UserSettings` object is edited in the table (see fig. 224).

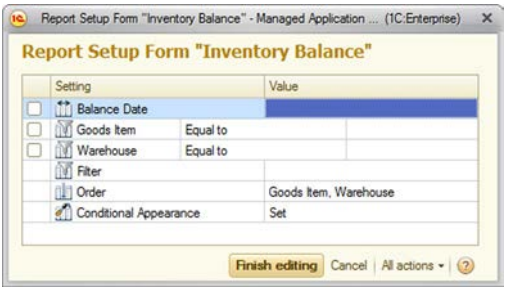


Fig. 224. User Settings Editor

TIP. It makes sense to put in the user settings (from all the variety offered by the data composition system) that are designed to manage the report by the end user. It is assumed that the user will only operate these settings.

For example, in a query that is used as a dataset for a report, there are many fields that can be selected by. However, the report developer believes that there are a number of selection elements that are used most often and it is logical to put them into user settings. Then the user will be able to edit both the "special" elements of selection (row `Goods` on fig. 224) and the whole selection (row `Selection` on fig. 224).

It is also logical to put in user settings the settings such as:

- **Period or date** - for almost all reports;
- Grouping and conditional appearance **- for tabular reports;**
- **Value of accounts** - for accounting reports, etc.

At the same time, for each report the developer will decide which settings will become user ones in the report (or report option).

10.3.7.4. Quick User Settings

The developer can select some frequently used user settings (for example, filter by product in a Balances report or filter by company in an accounting

report) to edit them directly in the report form. You can then specify that the user setting is quick (the `ViewMode` property is set to Quick Access).

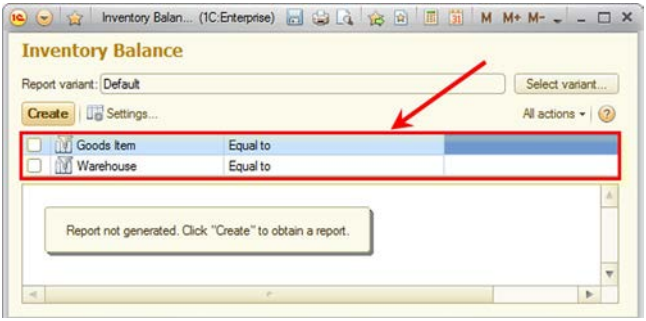


Fig. 225. Quick User Settings

The table used to edit user settings also has the `ViewMode` property, which determines whether to show all user settings or only quick ones.
If the user is not satisfied with the current composition of quick user settings, he/she can change their composition, for example, exclude from the composition of quick user settings those that he/she does not plan to change frequently.
To do this, the user needs to select the command More - Edit settings components (All actions - Change settings components) in the report settings edit window.

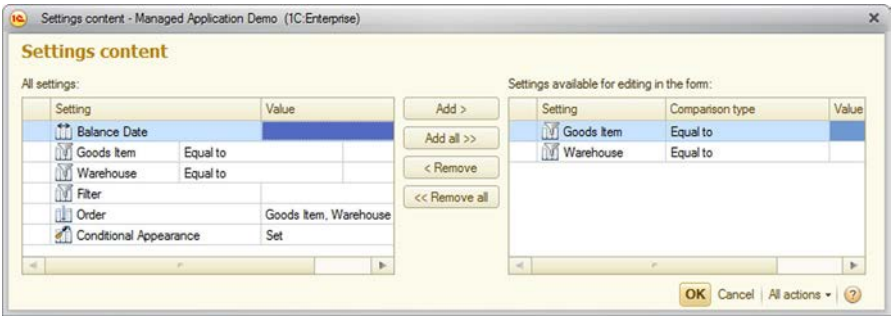


Fig. 226. Edit Quick User Settings Components

The left part of the form shows all user settings that can be selected quickly, and the right - settings that are currently edited in the report form.
In the filter or conditional appearance editing forms, there are commands to edit the properties of the user settings element. Thus, the user can transfer

those elements of selection or conditional design, which he or she constantly changes in his or her work, to the structure of quick user settings.
You can also manage the components of quick user settings from the form of editing the settings using the Edit as report column. This column is not displayed in the form by default and must be enabled using the More - Edit Form(All Actions - Change Form) command in the settings edit window.
In this case, you cannot add new settings, but you can quickly change the components of existing settings.

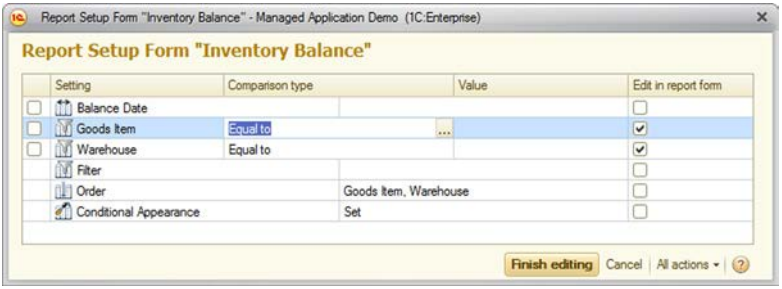


Fig. 227. Edit in Report Form column

10.3.7.5. Settings Composer

The Settings Composer has the `UserSettings` property. This property contains values of edited user settings. This property unavailable for writing with 1C:Enterprise language. In addition, the Settings Composer has the `LoadUserSettings()` method that loads the user settings values passed as the method parameter.
`GetSettings()` method allows to get a copy of the current settings (considering the user settings).
`LoadSettings()` method loads the passed settings into the Settings Composer (user settings are also overwritten based on the transferred data).

10.3.7.6. Filling User Settings Values

When filling in the user settings for different setting items, the user settings are added to the correspondingly filled items.
`For the types` DataCompositionFilterItem`,` DataCompositionFilterItemGroup`,` DataCompositionConditionalAppearanceItem`,` DataCompositionParameterValue`,` DataCompositionGroup`,` DataCompositionTable-

Group, DataCompositionChartGroup, DataCompositionTable, DataCompositionChart, NestedDataCompositionSchema in the corresponding user elements the UserSettingID property is filled as well as the properties that are in fact edited in user settings.
For types DataCompositionFilter, DataCompositionConditionalAppearance, DataCompositionOrder, DataCompositionSelectedFields an object of the corresponding type is created. The elements, the property ViewMode is different from Unavailable, are added to its collection.
There are exceptions:

- No items will be added that are marked as user ones. For example, a filter item that is marked as a user filter will not be placed in a user filter.
- Items containing user items will not be added. For example, a condition group will not be added if the group contains elements marked as user elements.
- For nested elements, the ViewMode property is not analyzed. They are added or not along with the parent elements.

For types ElementsCollectionOfDataCompositionStructure, DataCompositionTableStructureItemCollection, DataCompositionChartStructureItemCollection the DataCompositionSettingStructure object is created, into which the groupings are placed that already exist in the structure. Only groupings with set grouping fields are placed (detailed records are not placed). Placement of groupings is performed before detailed records, branching, table, chart, nested schema, unused grouping, grouping with user structure are met.

10.3.7.7. User Settings Usage

The application of user settings to the basic settings is performed in the GetSettings() method of the settings Composer. The following actions are performed at this:

- For types DataCompositionFilterItem, DataCompositionConditionalAppearanceItem, DataCompositionParameterValue the content of elements is copied to the corresponding user elements of settings.
- For types DataCompositionFilter, DataCompositionConditionalAppearance, DataCompositionOrder,

DataCompositionSelectedFields elements, which are located in the main settings and which are marked as Unavailable, remain unchanged. Items from user settings are transferred to primary settings. For Filter, SelectedFields and ConditionalAppearance they are added to the end of collection, for Order - to the beginning.

- For types DataCompositionFilterItemGroup, DataCompositionGroup, DataCompositionTableGroup, DataCompositionChartGroup, DataCompositionTable, DataCompositionChart, DataCompositionNestedObjectSettings the Usage property is set in the corresponding element of primary settings (based on the Usage of user settings element).
- For type DataCompositionSettingStructure the corresponding groupings are serached for in the structure element of the primary settings and are placed in the correct order. Missing groupings will be created. Groupings that are not found in the user settings or groupings that are disabled by the user are not deleted, but are marked in a special way. This allows to save them for possible future use. User groupings with an empty set of fields (detailed records) are ignored when applied.

10.4. Data Composition Template

The data composition template is represented by an object of the 1C:Enterprise language DataCompositionTemplate and consists of many other nested objects. The data composition template is a guide for performing data composition for a data composition system. Data composition template already contains description of area templates, text of executable queries, grouping arrangement, etc.

10.4.1. Data composition template components

Each data composition tempalte contains a set of objects describing a particular section. Consider these components.

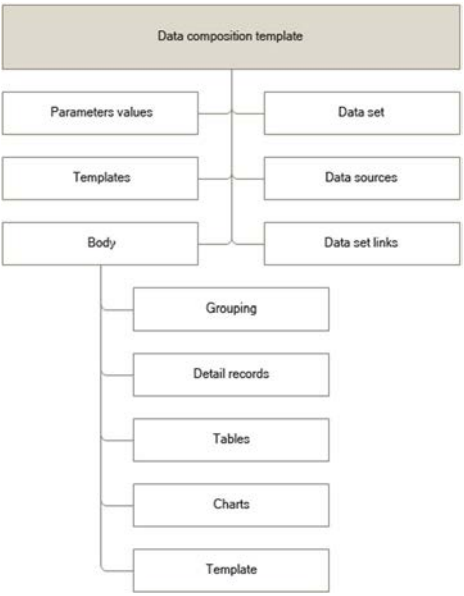


Fig. 228. Data composition template components

Data sources

A data composition template can include multiple descriptions of data sources.

A data source is a source to retrieve data from. The 1C:Enterprise infobase serves as a data source.

Data sources are described in the `DataSources` template property, which contains a collection of values consisting of the `DataSource` of `DataCompositionTemplate` elements.

You can create multiple data sources that use the connection string to one infobase.

Data Sets

Data composition template data sets contain a description of what data should be retrieved in the data composition.

They are described in the `DataSets` property of data composition template. Multiple data sets are allowed.

The nested data set is considered as a regular data set. The nested data set is always associated with its parent dataset. If a filter condition is specified for the nested data set, then the nested data set relationship to the parent dataset is considered internal.

Data Composition Template Data Set Field

A data set can contain the descriptions of fields available to the data set. Fields that are not described in the data set are unavailable. If some field is present in the data set query, but not in the description of data set fields, the field will be unavailable for use.

Data set fields are described in the data set Fields property, which contains a collection of values consisting of the `DataCompositionTemplateDataSetField` elements.

Data Composition Template Parameter Values

Data composition template can contain parameters in any expressions present in it.

Parameters values are described in the data composition template `ParameterValues` property, which contains a collection of values consisting of the `DataCompositionTemplateParameterValue` elements.

Data Composition Template Data Set Links

The data sets present in the data composition template may be linked to each other. The links between the data sets are described in the data composition template.

Data set links are described in the data composition template `DataSetLinks` property, which contains a collection of values consisting of the `DataCompositionTemplateDataSetLink` elements.

Data Composition Template Area Template

When the composition is executed, the composition result displays area templates. These area templates are also found in the data composition template.

Parameters values are described in the data composition template `Templates` property, which contains a collection of values consisting of

the `DataCompositionTemplateAreaTemplateDefinition` elements.

Data Composition Template Body

Previous components of the data composition template contained information about where to retrieve information from. The instruction how to compose the data is in the data composition template body. Data composition template body consists of the elements.

The following types of elements can be used:

- grouping - describes the grouping to be displayed in the result;
- detailed records - describes the detailed records of the data set that are displayed in the result;
- table - describes the table displayed in the result;
- chart - describes the chart displayed in the result;
- template - describes the templates used in output.

Grouping of table

Describes the grouping of a table and is represented by the 1C:Enterprise language object `DataCompositionTemplateTableGroup`.

A table grouping contains the same properties as a regular grouping, with the following differences:

- The table grouping body and the table grouping hierarchical body can contain only those elements that are contained in the table body, i.e. only table grouping, detailed table records, table grouping templates. The hierarchical body can additionally contain a hierarchical grouping of the table indicating the place where the hierarchical records of the grouping will be displayed.
- Table grouping templates are used as headings and footer templates.
- The following properties can be used:
  - `OverallsTemplate` - specifies the template used to output overall results by grouping, of`DataCompositionTemplateTableGroupTemplate` type;
  - `OverallResultsLocation` - contains the location of the overall results for the grouping, of `DataCompositionTotalPlacement` type.

Detailed table entries

Describe the detailed table entries, represent themselves as the object of the 1C:Enterprise language `DataCompositionTemplateTableRecords`.

Table Grouping Template

Table grouping template describes the templates used to output table grouping, and is represented by the 1C:Enterprise `DataCompositionTemplateTableGroupTemplate` object.

Chart Grouping

Describes the grouping of a chart and is represented by the 1C:Enterprise language object `DataCompositionTemplateChartGroupBody`.

A chart grouping contains the same properties as a regular grouping, with the following differences:

- The chart grouping body and the chart grouping hierarchical body can contain only chart grouping templates and chart groupings. The hierarchical body can additionally contain a hierarchical grouping of the chart indicating the place where the hierarchical records of the grouping will be displayed.
- There are no header and footer templates.

Chart Grouping Template

Chart grouping template describes the charts used to output table grouping, and is represented by the 1C:Enterprise `DataCompositionTemplateChartGroupTemplate` object.

10.4.2. Areas Templates

Area template is a declarative description of the location of the output data, as well as their visual design, necessary for the output of data in documents of different formats.

There are several fundamentally different areas templates:

- the actual template of the area,
- templates of chart areas.

Templates structure is shown on fig. 229.

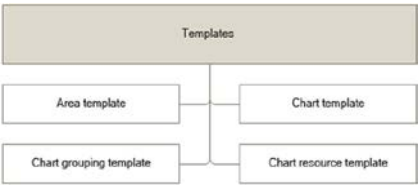


Fig. 229. Template Structure

In 1C:Enterprise language the area template is represented as the object of `DataCompositionAreaTemplate` type. This object is a collection of `DataCompositionAreaTemplateTableRow` objects.

10.4.2.1. Area Template Structure

As mentioned above, the area template is a collection of objects of `DataCompositionAreaTemplateTableRow` type. Table `Row` is a collection of cells located horizontally from left to right. Thus, several rows one after another of the table of data composition area form a square table.

Collection of Table Cells

This object is a collection of table row cells. It is described with the 1C:Enterprise language object `DataCompositionAreaTemplateTableCells`. The elements of collection are table cells - objects of `DataCompositionAreaTemplateTableCell`.

Table cell is a square area that is used for data output to documents of different formats. Cell can contain displayed fields, text or appearances.

Template Elements Collection

A collection of template elements (`DataCompositionAreaTemplateItems` object) is a collection of fields (objects of the `DataCompositionAreaTemplateField` type). These objects can be contained in the collection in any order. The data contained in these objects is used in various document formats.

Field

This element is a field displayed in a table cell or in a list item. Field can contain its arbitrary value and appearance. Field is described with the 1C:Enterprise language object `DataCompositionAreaTemplateField`.

Table Cell Appearance

Table cell formatting is a collection of objects describing table cell formatting. It is described with the 1C:Enterprise language object `DataCompositionAreaTemplateTableCellAppearance`.

Field Appearance

Field appearance is a collection that contain only one object - appearance element `Format`. It is described with the 1C:Enterprise language object `DataCompositionAreaTemplateFieldAppearance`.

10.4.2.2. Templates Structure of Chart Areas

There are three types of chart areas templates:

- chart template - object of DataCompositionAreaTemplateChartTemplate;
- template of chart resource - object of DataCompositionAreaTemplateChartResourceTemplate type;
- template of chart grouping - object of DataCompositionAreaTemplateChartGroupTemplate type.

When you create chart templates, one chart template, one chart resource template, and several chart grouping templates are formed. The number of grouping templates corresponds to the number of points and series in the chart.

Principle of operation

Area templates are used to output reports to documents of various formats. The area template is an integral part of the data composition template definition - of an object of the DataCompositionSchemaTemplateDescription type.

In order to display values of report fields in cells or elements of the area template list, it is necessary to assign to the Value property of the data composition area field (object of FieldOfDataCompositionArea type) the value of DataCompositionAreaTemplateField type, containing the parameter name. The parameter itself to be added to the list of parameters of the template definition, the parameter name should be assigned as the name, and the name of the output field or expression in terms of the data composition system expression terms - as an expression.

Chart Template

Chart template is used to describe the chart type. It is described with the 1C:Enterprise language object DataCompositionAreaTemplateChartTemplate.

Chart Resource Template

Template of chart resource is used to form the chart values and is described with the 1C:Enterprise language object DataCompositionAreaTemplateChartResourceTemplate.

Chart Grouping Template

Chart grouping template is used to form points and series of chart. It is described with the 1C:Enterprise language object DataCompositionAreaTemplateChartGroupTemplate.

10.4.3. Appearance Template

Appearance template is a declarative description of the particular report areas. These descriptions are used by the template areas generator when generating area templates based on settings elements of data composition.
To create a template interactively, when you create a template using the template wizard, specify its type Appearance template of data composition and click Finish. This opens the appearance template dialog box.

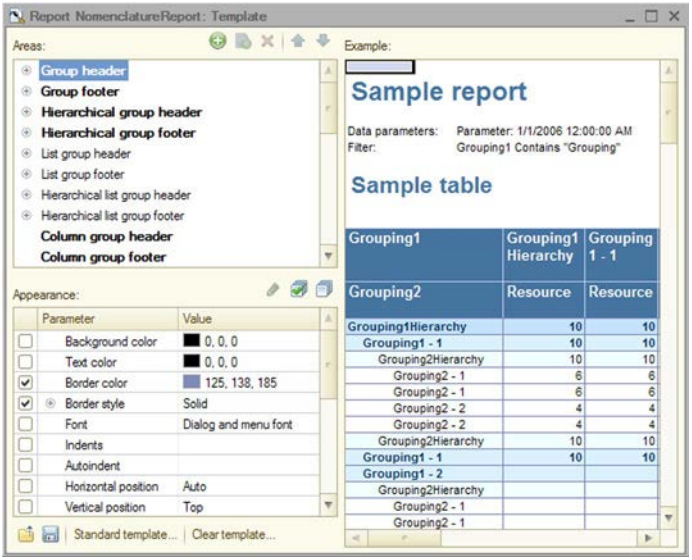


Fig. 230. Appearance template dialog box

The wizard dialog box consists of a list of appearance areas list, a table of appearance settings and a spreadsheet document field `Example` to display the result of the selected settings.
Non-empty template areas are displayed in bold.
The drag and drop context menu allows to copy the appearance parameters from one area to another or replace them.
With the Standard Template you can load already prepared template from the standard templates list, which also includes the common templates of configuration.
Remove Template button removes the appearance in the template.
Sequence of actions for appearance setting is as follows:

- select the appearance area;

- in the settings table specify (put the ticks) the parameters, appearance of which to be changed, and then specify the appearance values;
- results of the changes made can be viewed in the Example field.

If it is supposed that several levels of grouping will be displayed in the report, then it is necessary to create subordinate areas by the number of levels of grouping in the list of areas for each area, in which there will be levels. To create an area level, specify the area and click Add button of the command bar. A row is added to the list of areas, which name is LevelN, where N - the grouping level number. The grouping level formatting is selected as described above.

NOTE. If multiple levels are specified, deleting a grouping level always removes the lowest level of the current area, regardless of which level is selected in the list.

Appearance Template Structure

Each template consists of a number of template areas. The template area has a name - row from the list of template areas, indicating to which area to apply the template, and a collection of appearance elements.

Appearance element has two properties:

- Level - positive number, and if the level is 0, it is assumed that the formatting is applied by default to all templates of the area of a certain type;
- Appearance - collection containing the names of appearance properties and their values.

In 1C:Enterprise language the appearance template is represented as the object of `DataCompositionAppearanceTemplate` type.

Data Composition Appearance Template

This object is a collection of areas of appearance template. This collection is implemented in the form of parameter values. The name of the parameter is the name of the area. The name of the area is a predefined line from the list of areas. The value of the parameter is the object of `DataCompositionAppearanceTemplateArea`.

Data Composition Appearance Template Area

This object is a description of a predefined area and is a collection of the appearance template area elements (objects of `DataCompositionAppearanceTemplateAreaItem`).

Element of Appearance Template Area of Data Composition

This object is a description of the template area for a specific hierarchy level. If the level number is 0, the template is considered to apply to all areas of that type.

10.4.4. Templates Areas Generator

Templates Areas Generator allows to dynamically generate templates of data composition areas used to output data composition results to documents of different formats. The template area means a declarative description of the location of the output data and its appearance.

When the templates areas generator is running, the following steps can be highlighted:

- generation of report common templates,
- generation of groupings templates,
- groupings layout,
- output grouping fields layout,
- application of the appearance template,
- application of the conditional appearance,
- merging cells.

Let's consider each operation stage of the areas templates generator.

Generation of report common templates

At this stage, the templates areas generator creates common templates based on the data composition setting. The type and number of templates depend on the data composition element type.

Grouping Template

A special template is formed for the grouping - grouping header. This template contains the names of the output fields on the left side of the template and the names of the output resource fields on the right side of the template, for example:

Counter-party	Counterparty.Code		Count	Sum
Products	Products.Code	Products.Name		

Table Template

The following template group is generated for the table:

- **Table Header Template**. This template contains names of output fields of table rows, for example:

Counterparty	Counterparty.Code	
Products	Products.Code	Products. Name

- **Results Template by Rows**. This template contains a special word `Total` and names of resource fields, if they are displayed horizontally, for example:

Total	
Count	Sum

- **Results Template by Columns**. This template contains a special word `Total` and names of resource fields, if they are displayed horizontally, for example:

Total	Count
	Sum

- **Overall Results Template**. This template contains resource field, output in the table and necessary for display in the overall results table, for example:

Presentation(Sum(Sales.QuantityTurnover))	Presentation(Sum(Sales.AmountTurnover)

Layout of these templates inside the table is shown below.

Table Header	Columns Area	Results Template by rows
Rows Area	Resources Area	Results Area by rows
Results Template by columns	Results Area by columns	Template of overall results

Layout of results templates by rows and by columns is managed by the properties `OverallResultsLocationHoriznotally` and `OverallResultsLocationVertically`, respectively. The following options of the overall results location are possible:

- `No` - overall results are not displayed.

Table Header	Columns Area
Rows Area	Resources Area

- `Start` - overall results are displayed in the first column or in the first row of the table, respectively.

Table Header	Results Template by rows	Columns Area
Results Template by columns	Template of overall results	Results Area by columns
Rows Area	Results Area by Rows	Resources Area

- `End` - overall results are displayed in the last column or the last row of the table, respectively.

Table Header	Columns Area	Results Template by rows
Rows Area	Resources Area	Results Area by rows
Results Template by columns	Results Area by columns	Template of overall results

- `BeginAndEnd` - overall results are displayed in the first column/row and in the last column/row of the table.

Table Header	Results Template by rows	Columns Area	Results Template by rows
Results Template by columns	Template of overall results	Results Area by columns	Template of overall results
Rows Area	Results Area by rows	Resources Area	Results Area by rows
Results Template by columns	Overall Results Template	Results Area by columns	Template of overall results

- `Auto` - results on the columns are located in the last row, and results on the rows - in the last column.

Chart Template

The template contains the appearance properties of the chart.

Generation of Groupings Templates

There are three types of groupings in the data composition setting:

- grouping,
- table grouping,
- chart grouping.

A separate set of areas templates are generated for each type of groupings. But the layout of groupings relative each other, location of the output fields inside the groupings areas location of the resource fields are performed universally. During generating any template of the grouping area the following stages can highlighted.

Defining the grouping template type. Grouping template type is retrieved from the parameter `DataCompositionGroupTemplateType` of the `DataCompositionOutputParameterValues` object. This property makes sense only for simple groupings, i. e. for groupings that are not included into the table or chart. The following options of selected fields location are possible:

- `Auto` - automatic detection of selected fields location: if the grouping has the nested table, chart, nested report or grouping with the `Vertical` type of grouping template, then the selected fields are arranged vertically, otherwise - horizontally.
- `Horizontal` - horizontal layout of the selected fields, one after another from left to right.

Counterparty	Counterparty.Code	
Products	Products.Code	Products.Name
Aleks-2002	00009	
1C:Accounting 8	00013	1C:Accounting 8

- `Vertical` - vertical layout of the selected fields, one under another.

Counterparty	Aleks-2002
Counterparty.Code	00009
Products	1C:Accounting 8
Products.Code	00013
Products.Name	1C:Accounting 8

Layout of groupings relatively each other. Layout of groupings relatively each other is controlled by the parameter `GroupingFieldsLocation` of the `DataCompositionOutputParameterValues` object. The following location options are available:

- `Together` - groupings are located one under another. For example, for grouping by counterparty or products.

Counterparty	Counterparty.Code	
Products	Products.Code	Products.Name

Aleks-2002	00009	
1C:Accounting 8	00013	1C:Accounting 8

- `Separately` - each grouping is located in a separate area. Groupings are located one after another from left to right. Output grouping fileds are also displayed in the nested groupings.

Counterparty	Counterparty.Code	Products	Products.Code	Products.Name
Aleks-2002	00009			
Aleks-2002	00009	1C:Aspect 7.7	00015	1C:Aspect 7.7. Compact Trading System

- `SeparatelyAndInTotalsOnly` - each grouping is located in a separate area. Groupings are located one after another from left to right. Output fields are displayed only in this grouping.

Counterparty	Counterparty.Code	Products	Products.Code	Products.Name
Aleks-2002	00009			
		1C:Aspect 7.7	00015	1C:Aspect 7.7. Compact Trading System

- **Output Fields Layout**. There are two types of fields: actual output fields (field-owners and/or their attributes) and resource fields. The output of these fields has significant differences.
- **Fields Output**. As mentioned above, there are fields-owners and fields-attributes. Fields-owners are displayed in the area template according to their order in the data composition setting. Layout of the fields-attributes is controlled by a special parameter `AttributesLocation` of the `DataCompositionOutputParameterValues` object. The following location options are available:
  - `Together` - attributes fields are located together in a separate column and at the output are separated by a comma.

Counterparty	Counterparty.Code
Products	Products.Code, Products.Name

  - `Saparately` - each attribute field is located in a separate column.

Counterparty	Counterparty.Code	
Products	Products.Code	Products.Name

- `WithOwnerField` - attributes fields are located in one column with their field-owner and are separated by a comma and at the output are separated by a comma.

Counterparty, Counterparty.Code
Products, Products.Code, Products.Name

- `SpecialPosition` - fields-attributes are located in a special column, which in its turn is located to the right of other grouping columns.

Counterparty	Counterparty.Code
Products	Products.Code, Products.Name

- **Output of fields located in folders**. Layout of these fields is controlled by the Folder Location property. The following location options are available:
  - `Auto` - fields are displayed depending of the grouping type. If the grouping is a simple one, the fields are displayed horizontally; if the grouping is of table type – vertically.

Counter-party			
Products	Products Attributes		
	Prod-ucts.Code	Products.Name	Products.Main-Supply

Counterparty	
Products	Products Attributes
	Products.Code
	Products.Name
	Products.MainSupply

  - `Horizontally` - fields are displayed horizontally from left to right.

Counter-party			
Products	Products Attributes		
	Prod-ucts.Code	Products.Name	Products.Main-Supply

  - `Vertically` - fields are displayed vertically one under another.

Counterparty	
Products	Products Attributes

Products.Code
Products.Name
Products.MainSupply

- `InSeparateColumn` - fields are displayed in a separate column located to the right of other columns.

Counterparty	Counterparty.Code	
Products	Products Attributes	
	Products.Code	Products.Name	Products.MainSupply

- `Together` - fields are displayed together and at the output are separated by a comma.

Counterparty	
Products	Products Attributes
	Products.Code, Products.Name, Products.MainSupplier

- **Output of Resource Fields**. Specific property `ResourcesLocation` of the `OutputParameters` object colntrols the resource fields location. The following location options are available:
  - `Horizontally` - resource fields are displayed horizontally from left to right.
  - `Vertically` - resource fields are displayed vertically one under another.

If the header is not defined for the folder, the place for header is not provided in the template. In this case the fields are displayed according to the value of Folder Layout property.

Application of the Appearance Template

After the grouping and fields are located, the appearance template is applied to the generated area. The purpose of the appearance template application is that all elements of the area template have the pre-defined appearance properties added to them, for example `BackColor` or `TextColor`. Note that there are a large number of appearance templates of areas.

Application of the Conditional Appearance

After the appearance template is applied, the conditional appearances are applied to the area template. Applying conditional appearance is that appearance properties containing logical expressions are added to the area template.

As a result of the logical expression calculation, it is decided what value of the appearance property should be applied.

Merging cells

Once the area template is generated, it may contain empty cells. These cells need to be merged to better represent the area template. The following cells location options are available.

- The empty cells are located to the right of the filled cells. The cells located on the right are merged with the cells on the left.

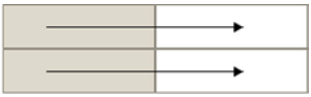


Fig. 231. Empty cells on the right

- Empty cells are located below the filled cells. The cells located below are merged with the cells located above.

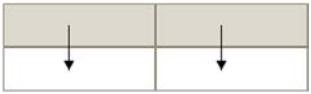


Fig. 232. Empty cells below

- Empty cells are located on the right and under the filled cells. The cells located on the right are merged with the cells on the left. Then cells located below are merged with the cells located above.

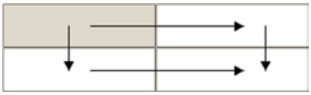


Fig. 233. Empty cells below and on the right

10.4.5. Generate Data Composition Template

Generating of data composition template is performed based on the data composition schema and settings. The data composition schema is created by the report developer, the settings are entered by the user.

Generating of composition template is performed using the 1C:Enterprise language object `DataCompositionTemplateComposer`.
This object does not have properties and has one method `Execute()`, to which the data composition schema is transferred, as well as the data composition settings, the variable (into which the `DetailsData` is placed), and appearance template. This method returns the created data composition template.
During the operation the template composer:

- Creates the composition template.
- Modifies the queries for data sets to obtain the necessary for user information, and places them in the composition template.
- Generates the filters in the query text or in the data set description.
- In necessary, generates the data sets to retrieve and check the hierarchy.
- Data composition template generates the necessary parameters with values set by the user.
- Fills the elements of data composition template body, into which it places the groupings, templates, etc., fills their parameters.
- Uses the generator of template areas. If the appearance template is indicated (the fourth parameter), this template will be used for the appearance of the generated template.
- If the third parameter is indicated, it creates the `DetailsData` object and places it into the transferred variable. Object created in such a way should be used for operation of composition processor and during decryption debugging.

Template composer during the data composition generation analyzes the value of the `Grouping` parameter of the `ComputeExpression()` function. If the parameter contains the name of the grouping, which is present in the settings, the template composer leaves this name unchaged. Otherwise, the template composer among the parent groupings of the current grouping searches for the grouping that contain in the grouping fields all the fields that are indicated in the `Grouping` parameter of the `ComputeExpression()` function, and places into this parameter the name of the found grouping.
For example, if the function `ComputeExpression()` parameter is the row `Products,ProductsDescription`, then during the template generation the template composer will find the grouping by fields `Products` and `ProductsDescription`, and will place the name of the found grouping to the generated template. At this the grouping is being search for among the groupings that are available in the place of grouping computing. So, if the template composer generates the expression for the table cell, the search of

grouping will be performed in the columns groupings, for which the cell is displayed, in the rows groupings, for which the cell is displayed, in the grouping, into which the table is nested. This, the groupings are searched for, for which the function `ComputeExpression()` can be applied for. If several of them are found, the closest grouping is selected. Firstly, in the table the grouping is searches for in the row, then the grouping in the column, then the grouping above the table.
If the grouping is not found, the expression is replaced with `NULL`.
Actually, this enables to use `ComputeExpression()` in the user settings, indicating the grouping not by the grouping names, but by the names of the groupings fields.

Example:

```
DataCompositionSchema = RetrieveDataCompositionSchema();
ExecutableSettings = RetrieveExecutableSettings();
TemplateComposer = New DataCompositionTemplateComposer;
CompositionTemplate =
TemplateComposer.Execute(DataCompositionSchema,
ExecutableSettings);
```

The generated data composition template can be used for execution, i. e. for obtaining the result.

10.5. Data Composition Processor

Execution of data composition is performed using the 1C:Enterprise application `DataCompositionProcessor` object.
The data composition template is passed to the enter of the data compositino processor.
Operation with the data composition processor is easy: after the installation the data composition processor of the data composition template, at this object it is possible to successively obtain the data composition results elements that can be further used, for example for output in the spreadsheet document, or be saved for further application.
Below is the example of operation with the data composition processor.

```
Controls.TDResultSpreadsheetDocument.Remove();

DCTemplate = RetrieveCompositionTemplate();

DCProcessor = New DataCompositionProcessor;
DCProcessor.Initialize(DCTemplate);

OutputProcessor =
New DataCompositionResultSpreadsheetDocumentOutputProcessor;
OutputProcessor.SetDocument(Controls.TDResultSpreadsheetDocument);
OutputProcessor.BeginOutput();

While True Do

  DCResultElement = DCProcessor.Next();
  If DCResultElement = Undefined Then

    Break;

  EndIf;

  OutputProcessor.OutputItem(DCResultElement);

EndDo;

OutputProcessor.EndOutput();
```

When initializing the data composition processor you may additionally specify:

- object - external data sets - structure, which contain the external data set as a key, and data set - as a value.
- decryption data - object, which will contain the information about the decryption;
- option to use external functions - flag that shows whether the functions of the configuration common modules can be used in the expressions.

10.6. Functional options and rights to view the field in the report

When receiving the default report settings, the data composition system automatically performs the following actions with the settings:

- If the user does not have the right for interactive view of some field or this field is connected with the disabled functional options, this field

becomes unavailable for user setting. I. e. it will not be displayed in the available fields list.

- If the field is connected with the attribute, the type of which is disabled by the fucntional option, this field is removed from the list of available fields.
- The query table and the attributes of this table become unavailable if the feature option disables the configuration object that forms the table.
- If a field that the user does not have permission to view or a field that is associated with a disabled feature is used in a user field, such user field is removed and selection is not set for it.
- A data composition system field is considered inaccessible if all fields used in the field expression are associated with disabled functionality or unavailable for viewing fields. All fields included in the field expression in the query are considered, except for fields participating in the conditions of queries and fields, participating in the section `WHEN` the selection operation (`SELECTION WHEN ELSE END`), as well as expressions from associations, if any, are considered as well.
- All fields that the user does not have permission to view and fields that are associated with disabled functionality are removed from the grouping fields. If after deleting a grouping field, the grouping does not have a single grouping field with a set flag of use, then the whole grouping is removed, and its contents (if, for example, the includes a table) is placed in the grouping location.
- If a grouping-row or grouping-column is removed from the table, and the table is without the groupings, then this table is deleted.
- If a grouping-series or grouping-point is removed from the chart, and the chart is without the groupings, then this chart is deleted.
- All fields that the user does not have permission to view and fields that are associated with disabled functionality are removed from the regulation.
- All fields that the user does not have permission to view and fields that are associated with disabled functionality are removed from the formatted fields of the conditional appearance element. If after deleting a formatted field from the conditional appearance element, the element does not have a single formatted field with an enabled flag of use, then this conditional appearance element is also deleted.
- If the selection of a conditional appearance element used a field, for which the user does not have permission to view, or a field that is associated with disabled functionality, such conditional appearance element is deleted.

If the field was previously available and the user saved the setting in any way, and later (when the field became unavailable) loads the setting, the fields will not be automatically deleted from the setting. This is done to allow the user to replace unavailable fields with other fields or to remove their use from the settings.

If during the performing of the `Execute()` method of the `TemplateComposerOfDataComposition` object the `CheckFieldsAvailability` parameter is set to `True` value, then the check of fields availability will be performed for the current user, as well as the check of field availability in the enabled functional option. If the unavailable field is used in the settings, an exception will be displayed. If the parameter value is set to `False`, the check will not be performed.

NOTE. If the report is executed fro, the automatically generated form, the fields availability for the current user will be checked, as well as the field availability in the enabled functional option.

10.7. Data Composition Result

Data composition result is represented by a set of data composition result elements. As the object of 1C:Enterprise language, the data composition result does not exist, only a set of data composition elements exist that form the result.

If necessary, the data composition result elements can be placed in some universal collection of values, for example `Array`, in order to manipulate the result a single whole.

The result elements can be obtained using the `DataCompositionProcessor` object, they also can be created and filled using the 1C:Enterprise language tools.

Data composition result elements can be output to the spreadsheet document using the output processor.

Let's see the example of the data elements.

Item 1

Property	Value
Item Type	Start
Templates	TableHeader, ColumnTitle, RowHeader, Resources
Nested Items Layout	Vertically

Item 2

Property	Value
Item Type	Start
Nested Items Layout	Horizontally

Item 3

Property	Value
Item Type	BeginEnd
Template Name	TableHeader

Item 4

Property	Value
Item Type	BeginEnd
Template Name	ColumnTitle

Item 5

Property	Value
Item Type	End

Item 6

Property	Value
Item Type	Start
Nested Items Layout	Horizontally

Item 7

Property	Value
Item Type	BeginEnd
Template Name	RowHeader

Item 8

Property	Value
Item Type	BeginEnd
Template Name	Resources

Item 9

Property	Value
Item Type	End

Item 10

Property	Value
Item Type	End

Result of output of such items should look as follows:

TableHeader	ColumnTitle
RowHeader	Resources

If item 2 contained the templates `TableHeader` and `RowHeader`, item 3 would use a template from that item, but item 7 would use a template from item 1, as item 2 ends with item 5.

Result elements can be stored in XML by standard means, for example:

```
WriteXML = New WriteXML;
WriteXML.SetString();
WriteXML.WriteStartElement("result");

DataCompositionTemplate = RetrieveCompositionTemplate();

DataCompositionProcessor = New DataCompositionProcessor;
DataCompositionProcessor.Initialize(DataCompositionTemplate);

While True Do

  DataCompositionResultItem = DataCompositionProcessor.Next();
  If DataCompositionResultItem = Undefined Then

    Break;

  EndIf;

  XDTOSerializer.WriteXML(WriteXML, DataCompositionResultItem,
"item", "http://v8.1c.ru/8/data composition system/result");

EndDo;

WriteXML.WriteEndElement();

Controls.DataCompositionResult.SetText(WriteXML.Close());
```

10.7.1. Outputting composition results to a spreadsheet document

Outputting report to a spreadsheet document is performed using the `DataCompositionResultSpreadsheetDocumentOutputProcessor` object.

Data composition result elements can be obtained using the data composition processor or generated by other tools.

Example of outputting the data composition result to a spreadsheet document:

```
Controls.TDResultSpreadsheetDocument.Remove();

DataCompositionTemplate = RetrieveCompositionTemplate();

DataCompositionProcessor = New DataCompositionProcessor;
DataCompositionProcessor.Initialize(DataCompositionTemplate);

OutputProcessor = New
DataCompositionResultSpreadsheetDocumentOutputProcessor;
OutputProcessor.SetDocument(Controls.TDResultSpreadsheetDocument);
OutputProcessor.BeginOutput();

While True Do

  DataCompositionResultItem = DataCompositionProcessor.Next();
  If DataCompositionResultItem = Undefined Then

    Break;

  EndIf;

  OutputProcessor.OutputItem(    DataCompositionResultItem);

EndDo;

OutputProcessor.EndOutput();
```

There is also an option to use `Output()` method of the `OutputProcessor` object. You can specify `DataCompositionProcessor` as the method parameter. In this case the composition result will look as follows:

```
OutputProcessor.Output(DataCompositionProcessor);
```

Data composition processor fixes the table header if one table or one grouping is output to the report (possibly with the nested groupings). You can manage the fixation options (top and on the left) using the output parameters settings Fixation on the left and Fixation at the top.

10.7.2. Data composition result output to a table or value tree

Report output to a table or value tree is performed using the `DataCompositionResultValueCollectionOutputProcessor` object. `SetObject()` method is the analogue of `SetDocument()`

method. If the `SetObject()` method was not called, the result will not be output in the value table.

Data composition result elements can be obtained using the data composition processor or generated by other tools.

Example of outputting the data composition result to the value tree:

```
TemplateComposer = New DataCompositionTemplateComposer;
DataCompositionTemplate =
TemplateComposer.Execute(DataCompositionSchema,
SettingsComposer.Settings, , ,
Type("DataCompositionValueCollectionTemplateGenerator"));

DataCompositionProcessor = New DataCompositionProcessor;
DataCompositionProcessor.Initialize(DataCompositionTemplate);

OutputProcessor = New
DataCompositionResultValueCollectionOutputProcessor;
OutputProcessor.SetObject(ResultTree);
OutputProcessor.BeginOutput();

While True Do

  DataCompositionResultItem = DataCompositionProcessor.Next();
  If DataCompositionResultItem = Undefined Then

    Break;

  EndIf;

  OutputProcessor.OutputItem(    DataCompositionResultItem);

EndDo;

OutputProcessor.EndOutput();
```

There are the following restrictions when outputting the composition result to the table or value tree:

- Only groupings or detailed records can be present in the settings. Use of tables or charts is not allowed.
- All folders indicated in the selected fields are ignored.
- The conditional appearance is not used, as well as the appearance for the field that is indicated in the data composition schema.
- Only the following is used from the output parameters:
  - overall result vertical layout,
  - fields header type,
  - number of records,
  - percentage of records.
- Pre-defined templates are not used.

There are two types of templates for result output to the table or value tree: header template (`DataCompositionAreaTemplateValueCollectionHeader`) and content template (`DataCompositionAreaTemplateValueCollection`).

10.8. Calculation of totals by balance fields in the data composition system

The balance field from the point of view of the data composition template is the field that has the `Balance` flag as a role.

10.8.1. Calculation of totals by balance fields

If the data composition template in some data set has the field of the opening balance, the data set should also have the corresponding field of final balance, and vice versa.
All fields-periods described in the data set should have the continuous numbering starting with 1.
For the correct calculation of results by the fields of the data source the data should correspond to the following rule: the data must be unique in the values of fields-periods and fields-dimensions, i.e. the data must not contain rows with the same values of fields-periods and fields-dimensions.
The following algorithm is used to calculate the totals for the fields-balances:

- if it is necessary to calculate the total of the field-balance for grouping by field-period:
  - if grouping has already been carried out for all fields-periods:
    - for each combination of fields-dimension, by which the grouping was performed:
      - the record nearest to the current period is obtained;
      - if the received record was for the current period, then opening and final balances will be obtained from this record;
      - else, if the received record has a previous period, the final balance of the record will be used as the opening and final balance;
      - else, the opening balance of the received record will be used as the opening and final balance;

- else (grouping has not yet been performed for all fields-periods):
  - for each combination of fields-dimension, by which the grouping was performed:
    - the first and last records, with fields of used periods equal to the current period, are obtained.
    - if records are found, the first record will be used as the opening balance, the last - as the final one.
    - if no records are found, the nearest record is obtained and its balances are used as opening balances and final balances, depending on whether the found record precedes the current period;
- else (not grouping by field-period):
  - the first chronology records for unused fields-dimensions will be used as the initial opening records, the latter - as the final one.

10.8.2. Calculation of totals by book balances fields

The calculation of totals by fields of book balances is performed in the same way as the calculation of totals for regular fields-balances. In addition, field-account information is used to calculate totals for such fields. If totals are calculated for a field-account grouping, or if a field-account grouping has been performed before the grouping, for which totals are calculated, the account type is used when calculating the totals. Otherwise, the account type is considered active and passive.
Depending on the type of account, the final balance will be calculated according to the following formulas:

- If the received balance is greater than 0:

	Dt	Cr
Active	Balance	0
Passive	0	-Balance
AP	Balance	0

- If the received balance is less than 0:

	Dt	Cr
Active	Balance	0
Passive	0	-Balance
AP	0	-Balance

10.8.3. Template Composition

To ensure that the correct totals are calculated, the template composer performs additional actions when generating the template:

- if the opening balance is used, automatically adds the final balance to the query and field, even if it is not used, and vice versa;
- if the report uses a dimension field-attribute, automatically adds the dimension field itself to the query, even if it is not used.

10.9. Operation with a hierarchy in the data composition system

The data composition system has the following aspects of using hierarchy:

- hierarchical groupings,
- `condition` IN HIERARCHY.

Let's consider these aspects in more detail.

10.9.1. Hierarchical groupings

When you create a grouping, you can specify that a grouping field needs to perform a hierarchical grouping.

For the application to perform a hierarchical grouping, the data composition processor needs to know where to get the data to build the hierarchy. This is implemented by creating a data set that specifies the relation of the data set to itself.

Consider an example. Suppose you want to build a hierarchy for a field of the `Catalog.Products` type.

The data set to build the hierarchy will look as follows:

```
SELECT
  Products.Reference AS Reference.
  Products.Parent AS Parent
FROM
  Catalog.Products AS Products
WHERE
  Products.Reference IN(&References)
```

For this data set, a link must be defined from the `Parent` field to the `Products` field with the `Reference` parameter. Thus, the data set will allow to consistently retrieve all the parents of the element.

The data set to build the hierarchy can be either explicitly described in the data composition schema or automatically generated by the data composition template composer.
If you specify a hierarchy data set in the data composition schema, you must add, for example, a query of the following kind:

```
SELECT
  Products.Reference AS Reference.
  Products.Parent AS Parent
{SELECT
  Reference.*,
  Parent}
FROM
  Catalog.Products AS Products
WHERE
  Products.Reference IN(&References)
```

This data set must be linked to itself, as described above. In addition, you need to create a link to the data set from the field, for which you want to create a hierarchy.

10.9.2. Hierarchical detailed records

You can report detailed records with a hierarchy using the data set link settings.
For example, let's define a data set `Products`:

```
SELECT
  Products.Reference,
  Products.Parent,
  Products.Code,
  Products.Name,
  Products.IsFolder,
FROM
  Catalog.Products AS Products
WHERE Products.Parent IN(&Parents)
```

Let's define the link if the `Products` data set with itself.
Set the `Reference` field as the expression-source field, as the expression-receiver - the `Parent` field. The link parameter - is the `Parents` field with the option to get a list of parameters. Therefore, from each row in the data set, the application will select the value of the `Reference` field, and it will retrieve records that contain that value in the `Parent` field. If you want the child elements to be selected only in groups, set the relationship condition expression to the `IsFolder` field. For the initial value of the hierarchical

relationship, we will use the expression `Value(Reference.Products.EmptyRef)`. This means that the first level of the hierarchical data set will have records that have a `Parent` field that is a null reference to the `Products` catalog.
To display such a set of data in the report in the data composition schema settings, it is enough to display detailed records (add the `Code` and `Name` fields to the list of selected fields).
The result is given in fig. 234.

0000027	Mouse Ice Mouse MUS-2
0000028	Mouse LOGITECH M-S48 PS/2
0000029	Mouse GENIUS "EASY" (3 buttons),
0000005	Keyboards
0000018	Keyboard Apple Pro Keyboards
0000020	Keyboard LK-601 KB-2000 PS/2
0000006	Software
0000045	Windows
0000009	Windows XP Home Edition Russian CD
0000010	Windows XP Home Edition Russian UPG CD
0000011	Windows XP Professional Russian CD
0000012	1C:Bookkeeping 8. Basic version
0000013	1C:Bookkeeping 8.
0000014	1C:Bookkeeping of a Government Institution 8
0000015	1C:Payroll and Human Resources 8
0000016	1C:Enterprise 8. Trade Management
0000007	Services
0000041	Delivery
0000042	SW installation
0000043	OS Windows setup consulting
0000044	1C setup consulting
0000033	Notebooks
0000034	Notebook Rover Computers Navigator KT7
0000035	Notebook Rover Computers Explorer
0000046	Telephone set
0000047	Phone Vega 700
0000048	Phone Vega 300
0000051	Phone Siemens SL45
0000052	Phone LG W7200

Fig. 234. Hierarchical detailed records

10.9.3. Display one item in multiple parent records

The data composiiton system allows to display a single item in multiple parent records when you build the hierarchy.
Let's give an example.
In the configuration, create two catalogs: `Children` and `Employees`, as well as the information register `EmployeesChildren` with dimensions `Child` and `ChildParent`.
In the `Employees` catalog, create records: Ivanova Maria, Ivanov Ivan.
In the `Children` catalog, create a record: Ivanov Stepan.

Suppose, there are records in the information register `EmployeesChildren`:

Child	Parent
Ivanov Stepan	Ivanov Ivan
Ivanov Stepan	Ivanova Maria

Create a data set query `Children`, which will receive a list of children:

```
SELECT
  Children.Reference
FROM
  Reference.Children AS Children
```

Let's also create a data set query `Hierarchy` that will receive hierarchical records:

```
SELECT
EmployeesChildren.Child AS Reference,
EmployeesChildren.ChildParent AS Parent
FROM
InformationRegister.EmployeesChildren AS EmployeesChildren
WHERE
EmployeesChildren.Child IN(&Reference)

UNITE ALL

SELECT
Employees.Reference,
NULL
FROM
Reference.Employees AS Employees
WHERE
Employees.Reference IN(&Reference)
```

The first union of the request chooses children's parents. The second part of the query selects employees because the hierarchical set must contain the hierarchical records themselves.

Describe the links between data sets.

As the source of the link, specify the `Children` data set, the receiver - data set `Hierarchy`. The source- expression is the `Reference` field, the expression-receiver - field `reference`. The link parameter will indicate the `Reference` field. Let's specify that a list of parameters is possible.

Let's describe the hierarchical relationship. The data set `Hierarchy` will link to itself, the field-source - is the `Parent`, and the field-receiver - is the `Reference`. The link parameter will specify the `Reference` field and specify the option of the parameter list.

IMPORTANT! The hierarchical data set field, to which the dataset is linked, must be named the same as the data set in the original data set. Otherwise, the system will not be able to retrieve the field requisites for the hierarchical set, including not being able to get a view for hierarchical values.

In the report settings, create a hierarchical grouping by the `Reference` field, whose view is changed to `Parent`, and run the report:

Child
Jone Jones
Samuel Jones
Mary Jones
Samuel Jones

Fig. 235. Resultant table

As a result, the record of Ivanov Stepan is displayed in both groupings.

10.9.4. Condition IN HIERARCHY

User can specify a condition `IN HIERARCHY` for the field. In this case, the user must be given records that are in the hierarchy of the specified reference. If this condition is imposed in the global filter, it will be in the query body as a condition `IN HIERARCHY`. If the condition is not used in the global filter, then to process the condition, the data composition processor must have a data set that will contain references that meet the criteria.
The data set to build the hierarchy can be either explicitly described in the data composition schema or automatically generated by the template composer.

10.10. Using the data composition system to develop application solutions

In the process of application development, the data composition system can be used by the 1C:Enterprise language in accordance with the object model described above.
In addition, the data composition system can be involved in the visual design of reports. For example, after you create a configuration object Report, you

can create a template for that report that contains a data composition schema. To do this, click the open button in the Main Data Composition Schema input field.

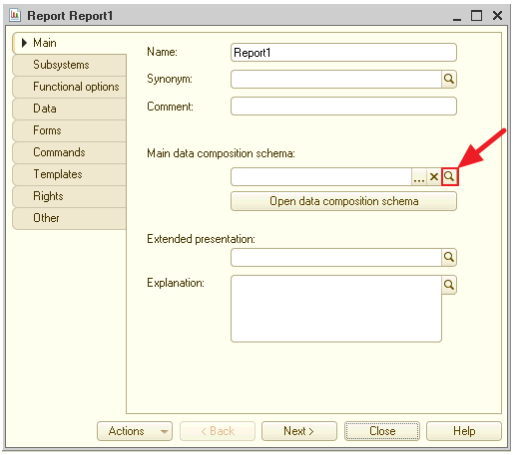


Fig. 236. Data Composition Schema Creation

Clicking Open Data Composition Schema will open the main data composition schema. If there is no schema, a new one is created and assigned as a main one.

This will open the Template Wizard, which will create a template that contains the data composition schema.

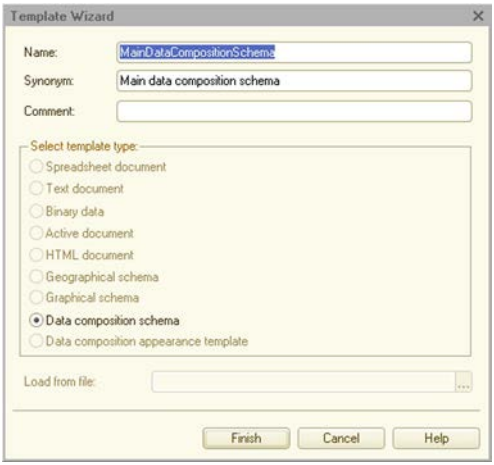


Fig. 237. Template Wizard

When you click Finish, you will open the Data Composition Schema Wizard, which you can use to create a data composition schema or load an existing composition schema from an XML document.
After the data composition schema is created, the report is ready for use and can be started in 1C:Enterprise mode. When you run the report, the application automatically generates the report form and the settings form.
On the Forms tab in addition to the main report form, you can specify:

- the form of the report settings that will appear when you run the Setting command in 1C:Enterprise mode, which is used to set user settings;
- the form of editing the report option (which will open when you select the menu item More- Change the option (All actions - Change the option) of the automatic report form), by which an advanced user can edit the current version of the report.

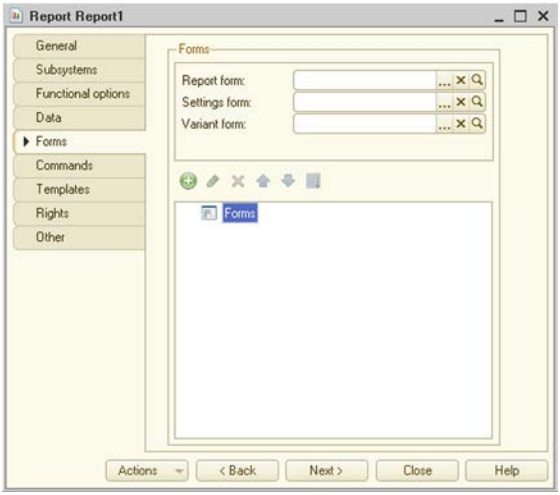


Fig. 238. Customizing report forms

If the forms that the application automatically generates when using a data composition system-based report do not suit the end users of the application, then the application developer can create their own forms. To do this, use the features of the Report Form Wizard.

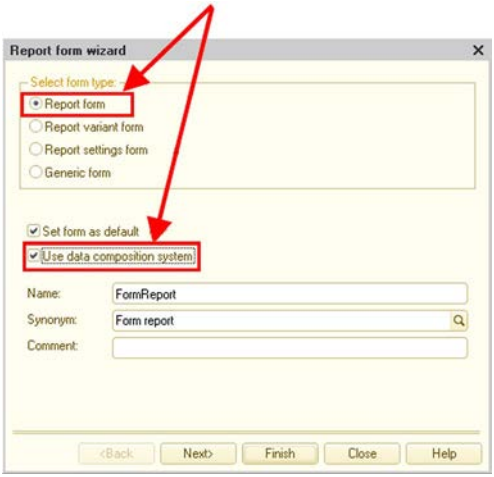


Fig. 239. Report Form Wizard

10.10.1. Input parameters

In the data composition schema for data set fields, calculated fields, and parameters, you can specify input parameters. These parameters describe how to enter values for fields in the selection and parameters. For example, in a Warehouse Inventory report, you can specify that the Quick Choice will be available for the `Products` field. Then, by setting the item selection value, we will be able to use this feature. You can also specify a special selection form for the `Warehouse` field of the same report.
More information about input options can be found in the online Help.

10.10.2. Using metadata object properties in reports

If the data composition schema uses the data set - query, the query fields from the metadata objects get information about the input parameters and some of the appearance options. Therefore, if you do not specify an input or output parameter in the data composition schema, the value will be automatically derived from the corresponding metadata object.
The following input parameters are obtained from the metadata object:

- Mask,
- Choice Parameter Links,
- Link By Type,
- Type Link Item,
- Choice Form,
- Editing Format,
- Quick Choice.

In addition, the following output parameters are obtained from the metadata object:

- Format,
- Mark Negatives,
- Multi Line Mode.

10.10.3. Background Report Execution

When operating the application, there may be situations when reports are generated for a long time. It is desirable that the user be able to perform other

operations with the application during such a long report. This feature is provided by the mechanism of background execution of reports (using the mechanism of background tasks).

The automatically generated report form uses the background execution of the report in the client/server mode of the application, and in the file mode - directly generates the report. If the client/server operation mode does not require the background execution of the report, you can use the report form extension method `ComposeResult()`, indicating that the report is directly generated as a parameter.

Example:

```
ComposeResult(ResultCompositionMode.Directly);
```

If you interactively change user settings, interactively change or select a report option while the report is running in the background, the report will stop running and the spreadsheet document will be displayed in lighter colors.

If the report form is closed during background execution, the report execution will be automatically terminated even if the `StandardProcessing` parameter in the event handler of the report form `BeforeClose` is set to `False` value. The execution of the report (and the corresponding background job) is stopped after the `OnClose` report form of the event handler is called.

When generating a report, various errors can occur, for example, a report can take up more RAM than the server limits. In this case, an exception is displayed. It is recommended that exceptions that can be generated during report generation be displayed to the user instead of being ignored.

The background report execution can be accompanied by displaying the status window. The status window will be generated if it is allowed (the `AutoShowStateMode` Report Form Extension property), and if the report runs longer than 2 seconds (4 seconds in low-speed connection mode). To display the status of the report, the data composition system uses the `StatePresentation` property of the spreadsheet document field. If the report is generated within a specified time, the status is not displayed and the result is displayed immediately.

The client application will check the completion of report generation at a certain interval. In this case, the interval between each next check is 1.4 times longer than the previous one (but not longer than 20 seconds).

To control whether the report state can be displayed, the Report Form extension property is set to `AutoShowStateMode`. The property determines how a spreadsheet document that displays the result of the report will display the status of the report. By default (for example, for an auto-generated form),

the value of this property is Auto. This means that the status window will display information about the report being irrelevant and that the report is being formed.

10.10.4. Using a single color for a series

Data composition system allows it to set up a chart in a report, so that reports displaying similar objects use same colors for similar objects in a chart. For that purpose, the following two chart settings are used in data composition system: Chart Kind. Save Colors and Chart Kind. Color Save Key.

All primary operations are performed using he first property (Chart Kind.Save Colors). It defines the way the platform ensures consistency of objects and chart colors:

- For user - in this scenario data as to consistency of the series object and chart color is stored in the system settings storage with Common/ChartColors ID (ValueAppearance object is used).
- For current result - in this scenario data as to consistency of object and chart color is stored as part of a single report generation only without saving consistency data with respect to the object and color in between report generation. This can be useful, if a report displays several charts using a single object set for the chart series. If this value is selected in settings, all series related to one and the same object will be displayed in different charts in one color. However, during further generation this color can be different.
- For user key - in this scenario object and color consistency data is stored in the system settings storage with a key generated using a value specified in Chart Kind.Color Save Key. The storage will store ValueAppearance object. Using this parameter value, you can procure that several reports use a single object and color match data set in all charts. A full system storage key is generated through concatenation of Common/ChartColors/ and Chart Kind.Color Save Key. If any value is saved with respect to this key, data composition system behavior will be similar to that, when Chart Kind. Save Colors parameter is set to Do not use.
- Do not use - means that data composition system defines series color each time in accordance with internal rules.
- Auto - Auto value interpretation depends on the following parameters:
  - If consistency mode is Version 8.3.15 (or earlier), and there are two or more series groupings, Auto is interpreted as Do not use.

- If no Chart Kind. Color Save Key value is set, and consistency mode is Version 8.3.15 (or earlier), Auto is interpreted as Do not use.
- If Chart Kind. Color Save Key is set, Auto is interpreted as For user key.
- Otherwise, Auto is interpreted as For the user.

So far as reports are concerned, where a great number of various series is expected to be displayed in a chart per a single configuration user, it is recommended to set Chart Kind. Color Save Key to For current result. So far as reports are concerned, where in various scenarios a report displays data having regard to small series groups, it is recommended to set a color save key.

If when you generate a chart data as to consistency of objects and colors is retrieved from the system settings storage, as soon as a chart is generated data composition system re-writes `ValueAppearance`. Therefore, object - color consistency always remains up-to-date.

Client application form extension supports Clear Saved Chart Colors command which may be used to manage reports. It enables a user to delete any object - color consistency which already exists. Only objects used in the current report are cleared in Chart Kind. Save Colors and Chart Kind. Colors Save Key. However, series colors defined when a report is being developed remain valid. The said command is available, only if the current report has charts.

10.11. Features of using the data composition system

- The data composition system is not allowed to use names equal to keywords as the data path: `CHOICE`, `WHEN`, `THEN`, `OTHERWISE`, `END`, `IS`, `NOT`, `LIKE`, `SPECIAL CHARACTER`, `DIFFERENT`.
- If the parameter has period - standard period with an empty start date or end date – it is assumed that the nested `StartDate` and `EndDate` parameters are not set. That is, if the `Period` parameter of the start date contains a blank date, the `Period.StartDate` parameter is considered to be unspecified. Similar to `Period.EndDate` parameter and for period end date. Correspondingly, the parameters in which the `Period.StartDate` and `Period.EndDate` parameters are used are considered unspecified.

- The query is considered incorrect if it contains the DIFFERENT keyword, and the ARRANGE BY clause specifies an expression that is not in the selection list.
- Grouping on fields-periods by fields-attributes is prohibited.
- If the user does not have the right to interactively view the metadata object, the composition system considers all fields in the object table unavailable.
  For example, suppose you use the following query as a data set:

```
SELECT
  Doc.Reference.Date,
  Doc.Reference.Number,
  Doc.Products,
  Balances.Balance
  FROM Document.SalesInvoice.Content AS Doc
LEFT JOIN AccumulationRegister.ProductAccounting.Balances Balances
  BY Doc.Products = Balance.Products
```

  If the user does not have any interactive rights to the SalesInvoice table or Sales.Invoice.Table, then the Date, Number, Products fields will not be available. If you do not have rights to the AccumulationRegister.ProductAccounting table, the Balance field will not be available. If there is no right to any table, no field will be available.
- The child fields of a composite type are available in user preferences if you have the View right at least for one table from a composite type and the field itself. Child fields become available for selection, can be included in filter, sorting, a conditional appearance and in other places of user settings where available fields are.
- Generating a data composition template removes decryption expressions from predefined templates, which contain fields that are not available by function options.
- The field of the hierarchical data set, with which the communication is performed, must have the same name as the field in the data set - of the source. Otherwise, no detail fields will be obtained from the hierarchical data set.
- If a dataset query in the selection list produces an expression that is the application of the PRESENTATION() function to an expression that is obtained in another field, then at the automatic filling of the available fields, the field, in which the presentation is obtained, will not be available for use in the settings.

- When compiling data, an error will be thrown if grouping is performed by a calculated field and the grouping specifies the need to add in the period.
- In data composition system grouping templates, the autoindent is performed only for the upper-left template cell.
- When a batch query is used in a data composition system, fields that are not used in subsequent batch queries are removed from the selection list and grouping fields. If a query that generates a temporary table does not contain any field- this query is removed from the batch query. Also, data sets that do not use any of the fields are excluded from the resulting template. To prevent such fields from being deleted, either use this field or select the Required check box as a data composition schema role.
- If the temporary table fields that are created in the query of the data composition system are not used by subsequent requests from the batch query, and if the temporary table that you are creating is not in the required joins, the query that creates a temporary table is removed from the batch query of the data composition template.
  For example, suppose you use the following query as a data set:

```
SELECT
  Product
INTO
  TempTable
FROM AccumulationRegister.Sales
Group by Product
;
CHOOSE Warehouse, Product
FROM AccumulationRegister.Balances LEFT JOIN TempTable BY True
```

  If the `Product` field is not in the data composition settings, the first batch query will not be deleted, because the `TempTable` is in a mandatory connection.
- When displaying the data composition results in the collection of values, the fields of different groups referring to the same data composition field fall into one column of the collection.
- When a data composition system data set is used in a query with automatic filling of the available structure settings `MERGE` or `MERGE ALL`, as well as the selection overlay on the field attribute, the selection is applied only to those parts of the join, in which this attribute exists. When you apply a selection to a field, the selection is applied only in those parts where the field expression is not `NULL`.
- When the list of available fields is generated, privileged mode of operation is considered in case of use from 1C:Enterprise language. If you

want the contents of the available fields to be based on user access rights, but the composition result has all the data, it is recommended that you perform the composition of the data template in normal (non-privileged) mode, and the data composition itself perform in privileged mode.

- During the calculation of the data composition system expressions, the transition to Safe mode is performed.
- When using data from tabular part fields derived from other fields in the data composition system query, it is recommended that you explicitly specify aliases for table parts and its nested fields. Otherwise, you may have problems with the availability of fields when composing data composition template.
- If you want to perfrom the selection by the `DataVersion` or `PredefinedDataName` attribute of some field, with a comparison type other than equal/not equal, filled/unfilled, greater than (or equal to)/less than (or equal to), in the list/not in the list, you need to ensure availability of this field for selection, not just availability for use in selection.
- If it is necessary to set the value of the parameter, which must be set by the user for correct operation of the data composition schema query, then when enabling availability of this parameter for the user, it is recommended to set the parameter to Always.
  Otherwise, if the option is disabled and the option is disabled due to unavailability by functional options or, directly, by the user of the report, an error will occur about an unknown value of the parameter when executing the report.
  For example, the report uses the query:

```
SELECT
  SalesInvoice.Date,
  SalesInvoice.Number,
  SalesInvoice.DocSum
FROM
  Document.SalesInvoice AS SalesInvoice
WHERE
  SalesInvoice.Warehouse = &Warehouse
```

  If it is necessary to set the `Warehouse` parameter by the user, you should remove the user's accessibility restriction, but also enable the use of the parameter in Always.
  Если же возможно, то лучшим решением будет использование параметра в секции запроса компоновки данных `{ГДЕ}`.

In this case, the application will automatically disable conditions that are not enabled.
Example of a modified query:

```
SELECT
  SalesInvoice.Date,
  SalesInvoice.Number,
  SalesInvoice.DocSum
FROM
  Document.SalesInvoice AS SalesInvoice
{WHERE
  SalesInvoice.Warehouse = &Warehouse
```

- The source of the available settings of the data composition system does not automatically make it possible to use the fields that have the password mode specified in the properties of the metadata object. Thus, the field cannot be seen in the report or dynamic list.
  When performing template composing from the 1C:Enterprise language, the fields with password can be used if the value `False` in the `CheckFieldsAvailability` parameter is transferred to the `DataCompositionTemplateComposer` method.

Chapter 11. Accounting

Accounting in 1C:Enterprise is provided by the configuration objects Charts of Accounts and Accounting Registers. Since these objects are closely related, this chapter provides a general description of the capabilities provided by these objects.
The following sections will cover the configuration of the listed objects.

11.1. Overview

Designer tools allow to organize the accounting for multiple charts of accounts, and for each chart of accounts an arbitrary hierarchy of sub-accounts of large nesting can be built. The accounting results are stored by 1C:Enterprise separately for each chart of accounts.
The chart of accounts can have flexible numbering of invoices using numbers and letters.
Analytical records can be maintained for any account or sub-account and include various sections of the analytics. Analytical accounting objects can be the elements of 1C:Enterprise catalogs, documents, or arbitrary numbers, dates and rows.
The configuration tools allow to maintain an unlimited number of accounting types for each chart of accounts. A typical example of accounting types is quantitative and currency accounting.
The accounting type is enabled when you edit the chart of accounts. For each account, you can specify whether the account will have a different type of accounting.
Designer provides the means to set up accounting for several enterprises in one infobase. The summary information can be received both on the separate enterprise, and on the whole, on all enterprises. This feature will be useful for enterprises with consolidated balance.

11.2. Charts of accounts

A chart of accounts is one of the basic concepts in bookkeeping. A chart of accounts is a set of synthetic accounts intended for grouping data related to the

business activities of an enterprise. Data stored in such synthetic accounts provides the user with a complete picture of the state of the enterprise funds in monetary terms.

1C:Enterprise application provides flexible options for keeping the charts of accounts. Actually the required accounting type is organized by setting up the chart of accounts.

Several charts of accounts. 1C:Enterprise can contain several charts of accounts and accounting for all chart of accounts can be performed simultaneously. 1C:Enterprise can contain several charts of accounts and accounting for all chart of accounts can be performed simultaneously.

For example, such "multi-track" accounting will obviously be needed for joint ventures that need to keep records at the same time on two or more accounting standards.

Subaccounts. Charts of accounts in the 1C:Enterprise application support a multilevel hierarchy of account - subaccounts. Each chart of accounts can include an unlimited number of top-level accounts. Each account can also have an unlimited number of subaccounts. Each subaccount, in turn, can have its own subaccounts, and so forth. Number of subaccounts levels in 1C:Enterprise is unlimited.

You can set up an account code structure when you create a chart of accounts as a template that consists of an arbitrary sequence of characters. Technically, the account code structure does not affect the account hierarchy, but it is recommended to adhere to the code structure when creating the account structure.

Split account editing. It is known that mistakes in accounting are often caused by improper application of the approved chart of accounts. The most common situation - is the lack of analytical accounting on those accounts, for which the expanded balance is to be obtained. Then the accounting, being correct in form (all "coincides"), in fact gives wrong results. Therefore, it is often useful to limit the creativity of end users in setting up charts of accounts.

To this end, the 1C:Enterprise application allows to divide the process of editing the chart of accounts.

First, the editing of the charts of accounts can be performed in Designer by a specialist who performs the configuration of the 1C:Enterprise application.

New charts of accounts can be created in Designer. When you configure the charts of accounts, you specify their main characteristics: length of account code and account name; the maximum amount of extra dimension, which can be in one account (sub-accounts), etc. You can also enter the required accounts and sub-accounts (predefined accounts) in the chart of accounts, as well as set up accounting types in accounts.

Second, when operating with the 1C:Enterprise application, the end user can add their own accounts and sub-accounts to the chart of accounts, but they will not be able to delete the predefined accounts and sub-accounts created in Designer.

Storage of accounting totals. Accounting records are stored in the accounting registers according to the structure of the chart of accounts. The summary information can be viewed in the register forms and also extracted by means of the 1C:Enterprise language. For this purpose in the language there are methods of reception of balances and turnovers on accounts on any kind of accounting, as in the whole account or sub-account, and by objects of Analytical Accounting.

11.3. Analytical Accounting

Information about the assets of the enterprise, which accumulates in the accounts of accounting, often has a generalizing character. For example, in the "standard" chart of accounts there is a score of 10 "Materials", which is intended "... to summarize the information about the presence and movement of raw materials, materials, fuel, spare parts, packaging, etc. values belonging to the company". Sub-accounts can be opened to account 10 to account for different types of materials. But with this organization of accounting the information on the total cost of materials of the same type on the sub-accounts will accumulate, and generally in the account 10 - the cost of all materials.

To obtain detailed information on the availability of specific materials, the organization of Analytical Accounting for materials is necessary. In this case, the total amounts on the sub-accounts are broken down into smaller - costs of the specific materials.

In the 1C:Enterprise Analytical Accounting is organized with the help of a special mechanism of extra dimension.

Extra Dimension term. Extra dimension in the system 1C:Enterprise application is the object of Analytical Accounting. The term "Extra Dimension" can be designated for any objects of Analytical Accounting: fixed assets, intangibles, materials, organizations, accountable persons, contracts, budgets. The type of extra dimension, in turn, is a set of the same-type objects of Analytical Accounting. For example, the accounting of enterprise indebtedness to buyers and customers is necessarily conducted according to normative documents, separately for each buyer and customer. In 1C:Enterprise application such list of buyers and customers (suppose that it is only the organizations) will be called "type of extra dimension" "Organization", and any organization from this list will be named "extra dimension".

1C:Enterprise Designer allows to create any quantity of extra dimension types in accordance with the requirements of completeness of Analytical Accounting at the enterprise.

As a generic means of describing the properties of Analytical Accounting objects, Chart of characteristic types is used.

IMPORTANT! It is not recommended to use primitive types for the chart of characteristic types that is used as extra dimension types of the chart of accounts. This can significantly affect performance when recording the movements of the accounting register. We recommend that you use only reference types.

Organizing Analytical Accounting. 1C:Enterprise Analytical Accounting can be conducted on any account or sub-account. To do this, when you edit the chart of accounts, the necessary extra dimension type "attaches" to the required account or sub-account. It is possible to attach several types of different extra dimensions to the account or sub-account (their maximum number is defined in the Extra Dimension Max. Quantity property) and thus to organize conducting of Analytical Accounting in any necessary sections.

Entering information about business transactions in the 1C:Enterprise application is performed as records of accounting registers. For each of the corresponding account records (if the accounting is specified for these accounts), specify the Analytical Accounting objects.

For example, in manufacturing plants, you can attach extra dimension types accounts to the production cost account: Cost types for accounting by cost types, Products for accounting by types of products (works, services) and Subdivisions- for accounting by business units. Analytical information on costs can be obtained by any of these types of extra dimension.

In addition, the ability to conduct Analytical Accounting in the 1C:Enterprise application allows to keep a record of the same dimension object in different sections.

Thus, the same catalog can be assigned to different types of extra dimension. For example, an organization buys components from suppliers and sells finished products to customers. In this case, in 1C:Enterprise application the catalog of the Organization can be assigned as a type of extra dimension Suppliers and as a type of extra dimension Customers.

Extra Dimension and Sub-accounts. The properties of the chart of accounts in the 1C:Enterprise application (in particular, a large number of nesting levels of sub-accounts or a large length of the sub-account code), nevertheless, allow to organize the conduct of Analytical Accounting with the use of sub-accounts, but not extra dimension. However, the opportunities offered by the conduct of Analytical Accounting in sub-accounts and with the use of extra dimension are fundamentally different.

Sub-accounts are a hierarchical structure subordinate to a specific synthetic account. For example, to account for the debt to the enterprise from customers on the synthetic account of accounts receivable (in the standard chart of account it is the account 62) for each customer it is possible to open a separate sub-account and to keep the account of debts on this sub-account. This scheme will provide

information on both the amount of the enterprise's debt to a particular customer, and the total amount owed to all customers - it would be by summing up the information on the amounts owed from all sub-accounts.
However, if any of the customers-organizations becomes a supplier as well, it will also be necessary to arrange accounts receivable with the organization as a vendor. If you are conducting Analytical Accounting on sub-accounts for a new vendor, you will obviously need to open a new sub-account in the Accounts Payable account (account 60). Now, if you keep a record of this sub-account, you can get information about the netting with the organization as a vendor.
But to get data about the general state of mutual settlements with the organization, it is necessary to combine the information on calculations from these organizations from two accounts. To do this, you need to remember, which sub-account is in the account of accounts receivable and which sub-account is in the account accounts payable to a particular organization, to obtain information on the status of calculations from these sub-accounts and process it.
When you use the same extra dimension, the same extra dimension list, such as the `Organization`, is attached to all accounts (sub-accounts), on which you want to maintain an analytical accounting for organizations. Accounting with a specific organization as a vendor is carried out on the corresponding synthetic account. If the organization becomes a customer, you do not need to create a new position in the list of organizations - it is already present in this list as a vendor. Consolidation of information for one organization from two accounts of synthetic accounts is largely performed by the 1C:Enterprise application automatically. Similarly, you can combine data from any accounts that have been used to record transactions involving a particular organization.
Multi-level analytics. Multi-level analytics allows to get accounting results with varying degrees of detail. If extra dimension is used to conduct the Analytical Accounting, then for the implementation of multi-level analytics it is necessary to assign the chart of characteristic types as the types the extra dimension. Using Charts of characteristic types objects allows to keep records with the required degree of detail. The limit on the number of nesting levels imposes the use of a characteristic value type (for example, any hierarchical reference can be selected as a value type, and a composite type may be used).
When conducting Analytical Accounting using sub-accounts, multilevel Analytical Accounting is realized by using sub-accounts of different levels.

11.4. Accounting Types

As mentioned above, 1C:Enterprise application allows to maintain an unlimited number of accounting types. Examples of accounting types are quantitative and currency accounting.

Designer provides an opportunity to set up an unlimited number of different accounts. Accounting types are defined by subordinate objects of the Accounting flag type of the Chart of accounts object.

11.5. Records and record sets of accounting registers

The basic concept of accounting is the concept of **accounting transaction**. A transaction is any economic activity that is capable of causing a change in the state of enterprise assets.
Documents are used to enter information about business operations in 1C:Enterprise application. A document allows to enter in the application information on the economic transaction, to fix the date and time of transactions, the amount and the contents of the transaction.
The transaction credentials are automatically generated based on the document (data object of the `Document` type) and are recorded in the accounting registers. The order of such generation is defined in Designer by means of 1C:Enterprise language. When you record data for an operation, you can describe the completion of the operation details with different information from the document that generated the transaction.
In 1C:Enterprise the accounting of economic operation is always associated with the document that generated it: if the document needs to be edited, then the transaction data will be re-formed; If you delete the document, the business transaction data will be deleted as well.
Each accounting transaction consists of a common case of an arbitrary number of postings. In 1C:Enterprise each accounting transaction corresponds to a record of the accounting register, and the record set of the accounting register is analogous to an accounting transaction.

11.6. Maintain Consolidated Accounting

In accordance with the current legislation, the organizations having structural subdivisions, including those allocated to a separate balance (branches, representations), make the reporting taking into account the activity of all structural subdivisions. For such organizations, an important point will be the capability to conduct in one infobase accounting of all structural subdivisions and receive consolidated reports. In the preparation of accounting reports, it is also advisable to

separately present the main indicators of the activities of structural subdivisions to avoid unnecessary questions from the tax authorities.
Tools for accounting in 1C:Enterprise allow to conduct in one infobase accounting of several enterprises (or structural subdivisions of one enterprise) simultaneously. The solution of such problems is carried out by forming the measurements of the accounting register, which will indicate the reference to the structural subdivision.
The information that is entered in this dimension when operating with the 1C:Enterprise application is used as a characteristic for the separate storage of accounting totals. 1C:Enterprise application automatically organizes the system of storing accounting results, dividing them by the value of the specified dimension. With the help of 1C:Enterprise language the information from the totals can be extracted both for the particular enterprise and in general for all enterprises.

11.7. Create Chart of Accounts

To maintain the charts of accounts in the 1C:Enterprise application, the configuration objects in the Chart of accounts are used. The data objects of this type are the accounting records - accounting registers that will be used to group the funds when operating with the 1C:Enterprise application. Designer allows to create almost unlimited number of charts of accounts. All charts of accounts created in Designer can be used at the same time.
Charts of accounts in the 1C:Enterprise application support a multilevel hierarchy of **account - subaccounts**.

11.7.1. Properties of chart of accounts

Editing the properties of Chart of accounts-type objects and creation of the subordinate objects is performed in the editing window, and in the property palette. This section will describe the specific properties of the configuration object of the Chart of accounts type.
The Code Mask - is used to describe the structure of the account code and subaccounts. The following special characters are valid in the Mask line:

- ! - any character entered will be converted to uppercase;
- 9 - any numerical character is allowed;
- # - any numerical character, - (minus sign), + (plus sign), and a space are allowed;
- N - any alphanumeric (letters or numbers) characters are allowed;

- U - any alphanumeric (letters or numbers) characters are allowed and all the characters entered will be converted to uppercase;
- X - any character is allowed;
- @ - any alphanumeric (letters or numbers) characters in uppercase or a space is allowed;
- ^ - this character cannot be entered by user in interactive mode, it should be established by the script;
- h - characters used in hexadecimal notation are allowed.

In order to use one of the special characters in a mask, it is necessary to put in front of it a "\" symbol.

If the code mask contains dots or does not contain commas, all points will be automatically replaced by commas when you enter the account code.

Auto-Order by Code - if the property is set, the ordering by the Order field will be used instead of ordering by account code.

Order Length - If the length is greater than zero, the Order field is the default for ordering the chart of accounts.

NOTE. Maximum length of attributes Code Length, Name Length and the Order Length equals to 628.

Attributes - specify different descriptions of the account or sub-account, for example, the name of account 10 "Materials".

Accounting flags - are defined by a list of accounting types. Typical examples of accounting types are foreign currency and quantitative accounting. Configuration developers can create as many accounting types as they need to implement a specific task.

Tabular parts - tabular parts are not used for "standard" chart of accounts. An example of the use of subordinate objects of Tabular part type is organizing of simultaneous accounting of data in different charts of accounts to ensure the expanded correspondence of accounts of different charts of accounts.

Maximum amount of extra dimension - determines the maximum amount of extra dimension (up to 50) used for organizing of Analytical Accounting.

Extra dimension types - specify an object of Chart of characteristic types type.

Extra Dimension Accounting Flags - extra dimension accounting flags are created. When you set up an analytical accounting for accounts, these characteristics allow to determine whether a particular type of extra dimension is used in the formation of predefined accounts.

11.7.2. Organizing of different types of accounting

You can set up an arbitrary number of accounting types for each chart of accounts.
For each account or sub-account, when creating one, specify whether each accounting type defined for this chart of accounts is used.

11.7.3. Creating a list of accounts (predefined accounts)

Accounting records as data objects are created in Designer.
To create a list of predefined accounts in the Properties palette of an object of Chart of accounts type, in the Predefined property, click the Open link. This displays the accounts list window.

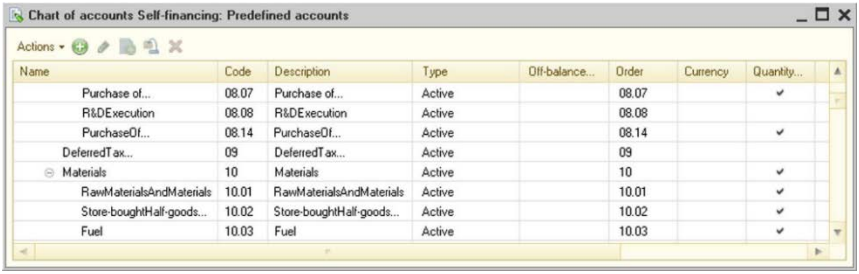


Chart of accounts Self-financing: Predefined accounts

Actions

Name	Code	Description	Type	Off-balance...	Order	Currency	Quantity...
Purchase of...	08.07	Purchase of...	Active		08.07		✔
R&DExecution	08.08	R&DExecution	Active		08.08		
PurchaseOf...	08.14	PurchaseOf...	Active		08.14		✔
DeferredTax...	09	DeferredTax...	Active		09		
Materials	10	Materials	Active		10		✔
RawMaterialsAndMaterials	10.01	RawMaterialsAndMaterials	Active		10.01		✔
Store-boughtHalf-goods...	10.02	Store-boughtHalf-goods...	Active		10.02		✔
Fuel	10.03	Fuel	Active		10.03		✔

Fig. 240. Chart of accounts

The accounts list is maintained by the Actions menu commands.

Account information is generated in the Account editing window.

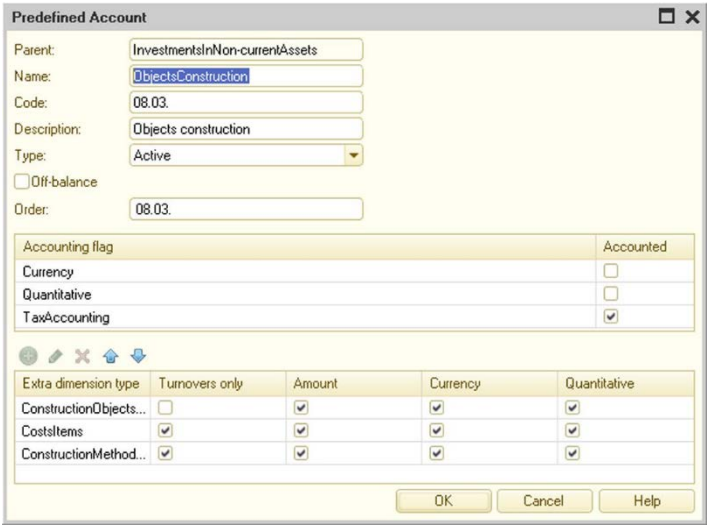


Fig. 241. Edit a Predefined Account

The type of the account is indicated in the Type attribute: Active, Passive or Active/Passive.

For off-balance accounts, select the Off-Balance check box. The totals of the off-balance sheet accounts are not accounted for in the balance.

The Order field is intended for the arbitrary ordering of invoices. The length of the field is set in the chart of accounts property Order Length. If the field length is greater than zero, this field is the default for ordering the chart of accounts. If the chart of accounts is set to Auto-Order by Code, the Order by field instead of ordering by account code or name will be used. To fill in the Order by account code field, the`GetCodeOrder()` method is used, which forms the invoice order line, considering the invoice code mask.

The first list shows the accounting characteristics that are defined for the Chart of accounts object. To indicate that the account you are editing belongs to a specific accounting type, select the check box in the Account column.

The following parameters are specified for each lower list entry. One of the predefined types of characteristics is selected in the Extra dimension type column. If subordinate objects are defined for this chart of accounts in the Extra dimension attributes branch, columns by the number of these objects are added to the list. For each extra dimension type, select a check box if you want to maintain an analytical accounting for this extra dimension. At this, you should remember the

following feature: if the account as a whole is not accounted for by any reason, then the corresponding accounting flag for the extra dimension should always be selected, otherwise the accounting flag for the extra dimension should be cleared according to the accounting needs.
In 1C:Enterprise mode, the end user can add their own accounts and sub-accounts to the chart of accounts and edit the properties of accounts (including those created in Designer). The deletion mark and deletion of invoices and sub-accounts (including those created in Designer) is subject to access rights. Description of the features of operating with predefined elements is given on p. 333.

11.8. Accounting registers

To reflect in accounting information about business transactions in the 1C:Enterprise application the accounting registers are used, described in the branch of the configuration tree Accounting Registers.

11.8.1. Accounting Register properties

In its view, the accounting register resembles the accumulation register. During its editing the structure of the register is developed: sets of dimensions, resources and attributes of the register are created; If necessary, screen and print forms are created to view the movements of the register.
The distinguishing feature of the accounting register is its connection with the chart of accounts and support of double entry mechanism. Each register entry contains the mandatory attributes Dt Account (debit account) and Kt Account (credit account) for the registers supporting the correspondence, and the Account attribute for the registers that do not support it.
The Enable Split Totals property allows to use the total separator mechanism, which provides a higher parallelism when writing to the register.

NOTE. The values of `UUID`, `BinaryData` types and rows of unlimited length cannot be used as accumulation register dimension types.

Period Refinement Length property allows to specify whether period refinement is used for this register and how many intervals the period refinement will include if it is enabled. This property indicates how many equal intervals one second will be split to:

- property is set to 0 - so the period refinement is disabled;
- property is set to a value other than 0- which means that the period refinement is enabled. In this case, each second can be broken down by 10 (the

property value is 1), 100 (the property value is 2), or 1,000 intervals (the property value is equal to 3).

When specifying movement in any of the period refinements, such movement will not be counted in the final balances of one period, but will be considered in the initial balances of the next, adjacent, period without distorting the turnovers of each of the two adjacent periods. At the same time, movements made in the refinement period will be reflected in the turnovers if the period of analysis is completely captures the point of time, in which the movements were performed with the specification of the period. The period refinement is logically located between the period and the recorder.

11.8.2. Accounting Register records

In order for the information entered by a document to make change in the accounting results, the document must generate accounting register records.

Records of a particular document make up a group (by the document-recorder) and at any changes in attributes of the document or in the records themselves are always "held" together. The order of generation of records is also determined by 1C:Enterprise language.

The structure of the record is built by the 1C:Enterprise application dynamically, depending on the setup of the various accounting elements that were executed when editing the chart of accounts. For example, a record can contain details for entering corresponding accounts, amounts, Analytical Accounting objects, quantity, currency type, and currency amount.

In addition to these attributes, Designer can write the necessary number of additional attributes to reflect any other necessary information. For example, it may be attribute to store a comment record.

When operating with the 1C:Enterprise application, the user can "disable" the recordings and then re-enable them. "Disconnected" entries do not affect the accounting totals - balances and turnovers on accounts. You can disable and enable records by using the Active property.

Such possibility is convenient for accounting of planned accounting transactions - of those, which should happen in the future. By disabling records of such an operation, you can not change the actual accounting totals. At the moment of actual fulfillment of accounting transaction it is enough to set activity for these records.

When you enter records, 1C:Enterprise application performs various predefined actions that facilitate and accelerate the user's operation with the application. For example, if an analysis account is specified for a corresponding account, the application will automatically open the required list of Analytical Accounting

objects (Extra Dimension) in order for the user to select the desired object. Many aspects of the behavior of the 1C:Enterprise application can be customized by editing the properties of the accounting register records.

Chapter 12. Periodic Calculations

12.1. General information

Periodic calculations - are calculations carried out at certain intervals, closely related to each other according to certain rules and mutually influencing each other within a certain period.
Periodic calculation mechanisms allow to adjust the order and relationship of calculations and to organize the accounting of their results.
The most typical example of the use of periodic calculations is the payroll calculation, in which calculations of accruals and deductions are made. Calculations are usually performed with monthly periodicity, and results of calculations of one kind can depend on availability and results of calculations of other kind.
Accounting in 1C:Enterprise is provided by the configuration objects Charts of Accounts and Accounting Registers. Since these objects are closely related, this chapter provides a general description of the capabilities provided by these objects.
The following sections will cover the configuration of the listed objects.

12.2. Basic concepts

Period. The concept of the period is important for calculations. Typically, the period is described by Start Date and End Date. If a periodicity is defined for the calculation (see description below), it is enough to specify any date to describe the period (activity, registration) of this calculation. This date calculates the start date of the period, and it is this date that will describe the period. This period definition allows to optimize the execution of queries, in which to select records that are related to the specified period.
Periodicity of calculations. Specifies the period, in which calculations that are considered by this register will (can) be performed. Set in the calculation register `Periodicity` property. The value of this property (if the register is periodic one) determines the validity period of the calculation register record. For example, the register has a periodicity of `Month`, then when the register entry is generated as the validity period, the document date is selected (for example, some calculation for November 2008), and the application determines the validity period at the beginning of the 01.11.2008.

Registration period - the start date of the period specified during the registration of the calculation (calculated by the date of the document-register). For example, in June 2008 the salary is accrued for May 2008 (calculations with `Month` periodicity). May 2008 - period of validity (the date of 01.05.2008 is recorded in the database), and June 2008 - is the registration period (the 01.06.2008 date is recorded in the database).

Validity Period - is the starting date of the period, determined according to the value of the `Periodicity` property. For example, the document indicates that the calculation is made in May 2008. For the value of the `Periodicity` property of the `Month` register, the validity period is determined by the 01.05.2008 date; for the `Quarter` value - 01.04.2008.

Calculation Validity Period - specifies the period, for which the calculation is performed. The period is determined by the start date and the period end date. For example, a sick-list record for May 2008 has a 01.05.2008 validity period, and the period of validity of the calculation is determined by the start date (for example, 06.05.2008) and the end date (for example, 15.05.2008).

Eviction mechanism or competition for the calculation validity period - display of link between the calculation types by the calculation validity period. The competition arises due to impossibility of execution of several calculations types simultaneously. You should select the calculation that will be performed in this period. Eviction mechanism is set in the description of a particular calculation type. This setting is performed in the `Evicting` (calculations types) section. For example, calculation Payment by Salary cannot be applied simultaneously with calculation of the Payment on the sick-list. It is said that the calculation of the sick-list evicts the calculation of salary, i. e., for the period, in which sick-list "acts", the salary is not accrued.

Actual Validity Period -if the calculation is not evicted by other calculations, the actual period coincides with the period of validity. If there are evicting calculation types, the actual period is defined as a set of non-intersecting periods, in which this calculation is not evicted. The fig. 242 shows graphic representation

of the actual validity period where the Salary calculation is evicted by the Sick-List calculation.

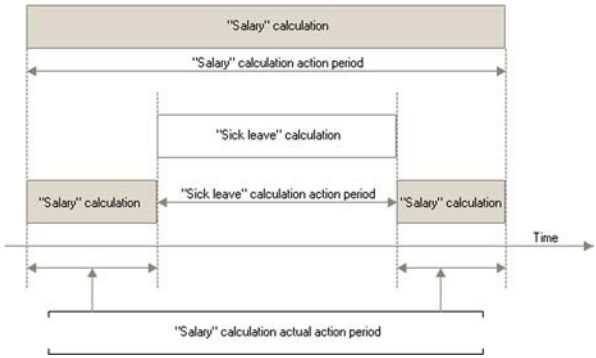


Fig. 242. Eviction Mechanism

The actual validity period is the secondary data, that is, the result of calculations performed by the application. The actual validity period is recalculated at any record of the set, including the blank one (cancellation of posting).

IMPORTANT! The result of calculations does not depend on the sequence of data input (registration).

Base Period - defines the period, for which the results of the calculations used (which are base) for this calculation will be selected. For example, when calculating the bonus for May 2008, the results of accruals performed by certain types of payments for a certain period (this may be Salary, Extra Pay, Vacation) are considered. This period will be the base for calculating the bonus. Setting the relationship of calculation types for the base period is made in the description of the specific calculation type in the Base (calculation types) section.

12.3. Charts of calculation types

Configuration objects of this type are designed to create the types of calculations that are used in calculation registers.
A reference to the calculation type - is one of the main properties of the calculation register records, which allows to give qualitative differences to different calculation register accounts.

Editing the properties of Chart of accounts-type objects and creation of the subordinate objects is performed in the editing window, and in the property palette. Next, the unique properties of the chart of calculation types will be described in addition to the common properties.

12.3.1. Specifying calculation properties

Dependence on base - if the Dependence on base property differs from the Does not depend, the calculation types of the chart of calculation types can specify a list of base calculation types. The composition of the base calculation types is determined by the Base charts of calculation types property.
Base charts of calculation types - specify a list of harts of calculation types whose calculation types can be included in the list of base calculation types. This list of base calculation types is used by the calculation register when the base receiving mechanism operates.
Uses the validity period - if the property is set, then this chart of calculation types can be assigned to the calculation register with the validity period. In this case, a list of evicting calculation types can be specified for each calculation type. The list of evicting calculation types determines the operation of the eviction mechanism of the corresponding calculation register.
The properties specified in this category determine the composition of the attributes that are included in the list when creating calculation types forms. If the Dependence on base- Depends on the validity (registration) period property is selected, you can place a table field on the form to select the calculations types that are the base of the calculation to be edited. If the Uses Validity Period property is set, you can place a table field in the form to select the calculations types that are evicting for the edited calculation type.

12.3.2. Predefined calculation types

The chart of calculation types features predefined calculation types that were created during setup. In 1C:Enterprise mode, the user can also edit such calculation types. The deletion mark and deletion of calculation types (including those created in Designer) is subject to access rights

To edit the list of predefined calculation types, go to the Other tab in the Properties palette and click the Predefined button. The screen displays a list of predefined calculation types of this chart of calculation types (see the example on fig. 243).

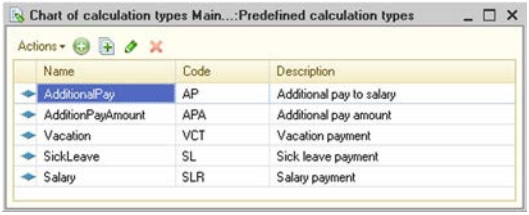


Fig. 243. Predefined charts of calculation types

Use the Actions menu to manage the list of predefined calculation types.
The structure of attributes (basic and leading) of predefined calculation types depends on the set properties of this object of the chart of calculation types placed in Calculation category.
Each predefined calculation type is edited in a modal window. The basic properties of the calculation type are entered on the Main tab.

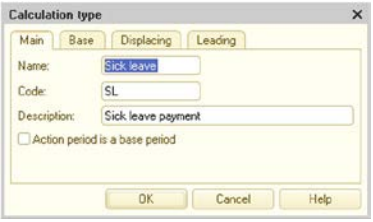


Fig. 244. Editing the chart of calculation types

If the Dependence on base- Depends on the validity (registration) period property is selected, the form contains Base tab. It specifies the calculation types

that are the basis for this calculation type and determines the mechanism of obtaining the base of the corresponding calculation register.

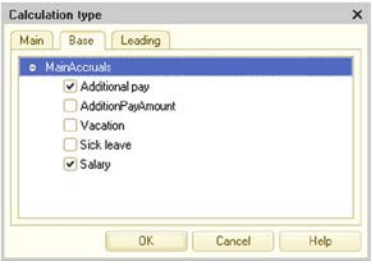


Fig. 245. Base accruals of calculation type

Thus, in the example (see fig. 245) for the calculation type, for the Sick-List the base calculation types are Salary and Extra Pay.
If the Uses Validity Period property is set, the form has an Evicting tab. It specifies the calculation types that evict this calculation type by the validity period and thus set up the mechanism of eviction of the corresponding calculation register.

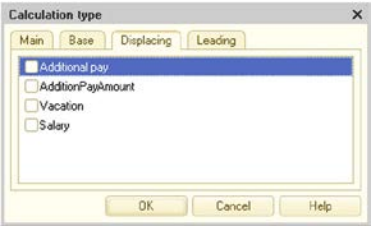


Fig. 246. Evicting Calculation Types

From the calculations listed in the list (see fig. 246), specify the eviction type of the calculation. In the example above, there is no Sick-List for the calculation type. But when describing the calculation of the Salary the Sick-List should be selected.

On the Leading tab, specify the calculation types that define the operation of the recalculation mechanism of the corresponding calculation register (i.e. the one, to which this chart of calculation types is assigned to).

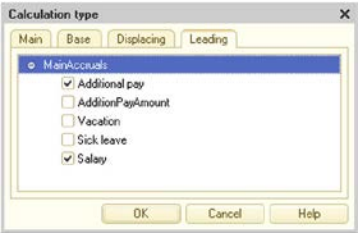


Fig. 247. Leading Calculation Types

In addition to predefined calculation types in 1C:Enterprise mode, the user can create their own calculation types and set the desired properties. For this purpose, in Desinger create an item form, in which you want to place table fields to specify the base, leading, and evicted calculation types, depending on the set properties of this chart of calculation types.

12.4. Calculation Registers

12.4.1. General information on calculation registers

Calculation Register - is a configuration object that allows to account the results of calculations performed by calculations specified in the chart of calculation types. Calculation registers allow, for example, to implement the registration of charges in favor of individuals (salaries, payment of sick-lists, payment of vacation days, etc.).
Calculation register accounts can affect the state of other register records. There are two factors of mutual influence of accounts of calculation register: competition for the period of validity and dependence on the base period. Let's explain these ways of mutual influence of accounts on examples.

12.4.1.1. Eviction Mechanism

The competition for the period of validity can be explained by the example of charges in favor of the individual the payment by the main form of remuneration (salary) and payment of vacation days. The account that corresponds to the salary payment and the account that corresponds to the vacation payment may not correspond to the same time interval, i. e. the time intervals, for which such accruals

are made, cannot overlap. In the calculation register, this behavior of accounts (register movements) is implemented by using the concept of the register record validity period. Each calculation register record has an actual validity period, which is set in the general case by a collection of several time intervals.
Thus, if the account corresponding to the vacation has a period of validity from 03.03.2008 to 13.03.2008, the account corresponding to the salary accrual may, for example, have an actual period of validity consisting of two gaps: from 01.03.2008 to 02.03.2008 and from 14.03.2008 to 20.03.2008.
The competition for the period of validity is supported by the mechanism of eviction of the calculation register, the operation of which is determined by the composition of evicting calculation types, validity period dates and so on.

12.4.1.2. Recalculations Mechanism

The dependence of accounts on the base period of validity can be considered on the example of accruals in favor of a physical person the salary payment and dependent on it accrual on average earnings. If the register changes the status of salary payment accounts in a certain period (deleting, changing or entering new accounts), this results in the need to revise the results of the payment accounts for the average earnings, if such records are available for the relevant period. To implement this reciprocal effect of the calculation register accounts, the concept of the base period of register record is introduced. So, if the account corresponding to the payment for average earnings has a base period from 01.01.2008 to 31.03.2008 (i. e. uses average earnings for three months), any changes in salary accounts for this period will lead to need to revise the payment account for the average earnings.

12.4.2. Editing Calculation Register

The Calculation Register branch of the configuration tree is used to operate the calculation registers.
When you edit a calculation register, you define a chart of calculation types, validity period and base period support, periodicity, and the register structure is developed: sets of dimensions, resources and attributes of the register are created; If necessary, screen and print forms are created to view the movements of the register.

NOTE. The values of `UUID`, `BinaryData` types and rows of unlimited length cannot be used as calculation register dimension types.

This section describes unique properties of the calculation register that are different from the general object properties.

Editing the properties of Calculation register-type objects and creation of the subordinate objects is performed in the editing window.

Chart of calculation types - is the main register characteristic. You can specify only one chart of calculation types for the calculation register.

Validity Period- if the property is set, the reciprocal effect of the movements of this register takes a competing character. An example of competing movements may be payroll and sick leave - a person cannot work and be on a sick-list at the same time, i. e., receive both salary and money on the sick-list. Such calculations are mutually exclusive in time, and the application must ensure that entering one of them will result in the exclusion of the other.

The calculation register with validity period support has a limit on the number of records with the same dimension values, registration period and calculation type. When calculating the actual validity period that occurs every time a record set is recorded, a low memory error may occur if a large number of identical records are involved in the actual validity period calculation. In fact, this means that there is a limit to the number of records in the same time cell. For example, one and the same employee has been repeatedly credited for the same period.

The value of the constraint depends on different conditions (DBMS used, computer technical data, etc.), but in any case, this restriction does not manifest itself in real practice.

Schedule - property is available if the Validity Period property is set. The property represents a reference to a case of information that describes the **Temporary schema** of the source data involved in the calculation. The schedule should be specified for calculations that depend on the source data distributed within the period of validity of a certain rule. For example, this can be a schedule for the organization's operating time by day, lecture hours by hour, and so on.

Schedule Value - property is available if the Validity Period property is set. The property selects the information register resource defined in the Schedule property.

For example, the chart shows the information register of the `OrganizationOperatingSchedule`. This register has the attributes of the `Workday` (`Boolean` type) and the `ServiceHours` (`Number` type). The first indicates whether the date is a workday and the second - number of service hours in the workday. The selection of `Workday` attribute means that the calculation will analyze whether a particular day of the calculation period is a work one (i.e., whether the payroll should be calculated for the specified day). The selection of `ServiceHours` attribute means that the calculation will select the number of service hours and based on this value the specified calculation will be performed . The distribution is set in 1C:Enterprise mode (manually or by means of 1C:Enterprise language).

Schedule Date - property is available if the Validity Period property is set. The property selects the dimension of the information register defined in the Schedule property and has a Date type. The value of this property relates to the values of the information register resource specified in the Schedule Value property.
Base period - if the property is set, the associated nature of the reciprocal effect of the movements of this register is established. An example of related movements is the relationship of calculating the accrual amounts to the average of the accrual amounts in the base period.
Periodicity - determines the period, with which movements are recorded and within which movements can affect each other (for registers that support the validity period).
Recalculations - register subordinate objects, which allow to define rules of mutual influence of registers movements. In the Relationship group of the object Properties palette, the register Dimension property specifies the main dimension of the current register to recalculate when the data of the leading registers specified in the leading register Data property changes. For example, recalculation of the deduction amount for an individual will be formed at the changes of accruals (salary, bonuses).
If the Base period property is set, the calculation of the recalculations data will be performed automatically. If the property is not set, the data must be generated manually by the user (the design should be designed with a special form for entering recalculations and the mechanism of their execution).

Chapter 13. Business processes and tasks

13.1. Basic concepts

Business processes in 1C:Enterprise are intended to combine individual operations into chains of interrelated actions that lead to accomplishment of specific goals. For example, issuing an invoice, accepting a cash payment, and shipping goods from a warehouse can be combined into a Product sale for cash.
Business processes in the 1C:Enterprise application allow to formalize processing procedures of the events arising in activity of the organization, and to ensure the participation of performers in them.
Application of mechanisms of business-processes in software allows to increase their efficiency, to improve the final result and to get new opportunities.
Business processes provide an opportunity to move to process management and qualitatively improve the company's operations through reengineering and automation of business processes.
The greatest effect is the automation of key business processes that start and end in an external environment.
Such chains of interrelated operations within a business process are presented in the business process flowchart. A flowchart describes the logic of the business process and its entire lifecycle, from start to finish, as a sequence of route points. The chain of interrelated actions will be called a business process movement.
A **route point** - reflects a stage in the lifecycle of a business process, normally involving the completion of a single automatic or manual operation.
Tasks in 1C:Enterprise allow to keep a record of jobs sorted by performer, and reflect the progress of business processes through the flowchart. In addition to business processes, tasks can be created by other infobase objects or directly by users.

13.2. General part

The mechanism of business processes in 1C:Enterprise is provided at once by several configuration objects:

- Business processes

- Tasks
- Information register
- Session Parameter

As a rule, the types of task addressing attributes and information register dimensions have a reference type (for example, `CatalogRef`, so more catalogs are added to the four types listed above).

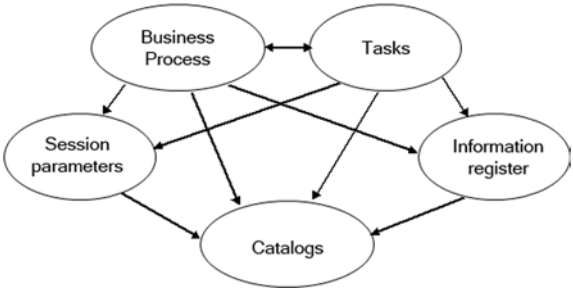


Fig. 248. Schema of Business Processes

The main objects - business processes and tasks - interact with each other (for example, the business process creates tasks, and the task in progress leads to its advancement on the route).
Auxiliary objects - session parameters, information register and catalogs - do not use each other or basic objects.
When creating a business process route map, catalogs with predefined data (roles, divisions, etc.) are used to set their values to address route points. Business processes create tasks when switching to route points and use Addressing (information register, see below) to process group points.
Tasks inform business processes about their performance, which causes them to move further along the route. They use the information register to filter the tasks for the current performer according to the set session parameter.

13.3. Routing

Business processes in 1C:Enterprise allow the following types of routing:

- **Hard**. Business process has a strict flowchart, which does not include conditional and parallel transitions, with hard-coded destinations for each route point. This type does not allow free or conditional routing.
- **Free**. The recipients of the business process flowchart point are not set and are determined programmatically or interactively during the business process lifecycle.

- **Conditional**. The flowchart includes checking conditions and switching by the respective branches. Switches can be either binary (condition) or multiple (option).
- **Parallel**. The flowchart provides for the division of the business process into parallel branches with the option of further merging (waiting). The advancement of the business process on each of the parallel branches takes place independently as the respective tasks are completed.

Typically, all these types of routing are found in real business process flowcharts.

13.4. Addressing System

Addressing system is the key concept in the mechanism of business processes and tasks in 1C:Enterprise. The main purpose of the addressing system - provide an opportunity not only for personal, but also for role-based addressing of tasks to business process participants.

Role addressing (role-based routing) - a set of rules and agreements fixed in the settings of metadata objects, which allows to determine the final addressee (performer), based on his/her assigned roles, affiliation with a division, as well as other addressing attribute.

Task addressing attributes define the dimension of address space in the context of an automated subject area and are used to determine the assignment of tasks to specific performers.

The definition of a specific performer is accomplished by using the properties of the task - Addressing, Main Addressing Attribute and Current Performer.

The process of determining the main addressing attributes from other addressing attributes is called **dereferencing**.

Addressing - a register of information that stores current information about the compliance of performers (the main addressing attribute) to structural units, workgroups, functions, etc., i. e. all other task addressing attributes.

One of the task addressing attributes is the main one and it means a certain employee - tasks performer.

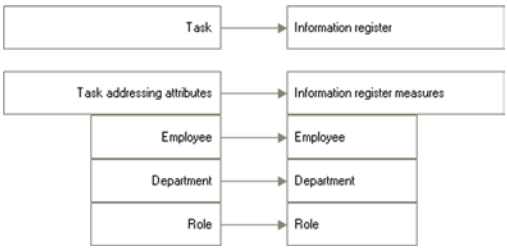


Fig. 249. Addressing Scheme

Let's explain the operation of the addressing system in the example.
Assume that there is an information register that consists of two dimensions- role and employee, in which the following entries are made.

Role	Employee
Cashier	Ivanov
Manager	Petrov

Assume that there is a business process (for example, Accept cash payment) where in one of the points the Cashier role is set in addressing properties. When a business process moves to this point, one task will be formed.

Task property	Value
Name	Accept cash payment
Role	Cashier
Employee	-

When an employee Ivanov looks through the list of tasks for himself, the addressing system will show him this task, because there is an entry in the information register that the role of the Cashier is set for Ivanov. Employee Petrov will not see this task.
Here is an approximate sequence of actions to create two different business processes:

4. We will assume that a 3-dimensional addressing system is selected - Employee, Role, Division.
5. Let's create handbooks for each of the planned addressing dimensions (`Employees`, `Roles`, `Divisions`) and fill them with the predefined values:

Employees	Roles	Divisions
Ivanov	Cashier	Accountancy
Petrov	Manager	Sales department
Sidorov	Department Manager	Warehouse
	Storekeeper	

6. Create an information register and add dimensions to it, one for each of the previously created catalogs. The dimension type should be set as a reference to the corresponding catalog:

Dimension	Type
`Employee`	`CatalogRef.Employees`
`Role`	`CatalogRef.Role`
`Division`	`CatalogRef.Divisions`

7. Create a session parameter `CurrentPerformer` and set it to the `CatalogRef.Employees` type.

8. Initialize the session parameter at application startup:

```
Procedure SessionParametersSetting(RequiredParameters)

  User = Catalog.Employees.
        FindByDescription(UserName());
  SessionParameters.CurrentPerformer = User;

EndProcedure
```

1. Create task.
2. Set the previously created information register to the `Addressing` task property.
3. Add the addressing attributes to the task similarly to the dimensions of information register:
   - Employee,
   - Role,
   - Division.
4. Set the type for the created task addressing attributes as a reference to the corresponding catalog and set a reference to the information register dimension in the Addressing Attributes property.

Addressing Attribute	Type	Addressing Dimension
`Employee`	`CatalogRef. Employees`	`Employee`
`Role`	`CatalogRef. Role`	`Role`
`Division`	`CatalogRef. Divisions`	`Division`

5. Select `Employee` attribute as the main addressing attribute, setting it in the corresponding task property.
6. Create the first business process and set a link to the task created earlier (`Task` property).

Let's design a business process flowchart, setting the necessary addressing attributes for the route points, selecting them from the predefined data of the corresponding catalogs.

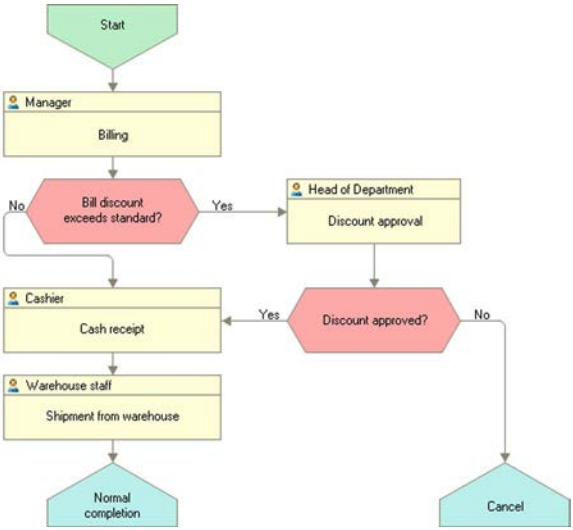


Fig. 250. Business process Flowchart

7. Create the first business process and set a link to the same task.
8. Let's design the flowchart of the created business process. And so on.

In the future, we will use this example to comment on the key features of business processes in 1C:Enterprise.

Let's consider in more detail a few key features of the mechanism of business processes.

13.5. Business process start

The business process starts when the `Start()` method is called or when the Start and Close button is clicked in the object form.

The following code fragment shows the programmatic creation, writing and start of a business process.

```
BP = BusinessProcesses.Agreement.CreateBusinessProcess();
BP.Date = CurrentDate();
BP.Write();
BP.Start();
```

The following sequence of actions is performed at start.

No.	Internal functionality	Handlers in 1C:Enterprise language
1		`BeforeStart` handler call at the start point
2	Moving along the flowchart to activity	
3	Generation of tasks	

Business process can be written, but not started. This can be useful if the creation of a business process and its start are separated in time. For example, when a business process starts when a certain event occurs.

13.6. Completing tasks

When the `ExecuteTask()` method is called, the execution check is performed, after which the task is marked as completed and the business process is notified of the fact. If all the necessary conditions are met, the business process moves to the next route point.

No.	Internal functionality	Handlers in 1C:Enterprise language
1	Transaction start	
2		`BeforeExecute` handler call from the task
3		`BeforeExecute` handler call from the relevant route point
4	Setting `Complete` task property as `True`	
5		`OnExecute` handler call from the task
6	Writing the task object	
7		`OnExecute` handler call from the relevant route point

No.	Internal functionality	Handlers in 1C:Enterprise language
8	Moving a business process to the next route point	
9	Generation of tasks at the new route point	
10	Transaction end	

13.7. Separation and merger

A separation point is used to divide a business process into several parallel (simultaneously and independently) executable branches. The separation point has one entry and an unlimited number of exits.
A merge point is used to synchronize previously separated branches. The business process will not be executed beyond the merge point until all branches in it have been completed. Thus, the merge point is the stage of the business process where all the tasks must be completed on previously separated branches.
Nested separation and merging is permitted. In this case, each merge point will synchronize only the branches of its own separation point.

IMPORTANT! Separation may be also completed without merging. In this case, the business process will have several parallel branches before it is completed.

Merging without separation is not allowed, and a corresponding message is generated when the flowchart is checked: Not all the lines at the merge point come out of the separation point..

13.8. Manual control

Business process completion and task execution flags can be set manually, bypassing the business process feature, but this must be done with a full understanding of all the possible consequences.

13.8.1. Business process completion flag

If the completion flag is set, the business process will be considered complete, even though the tasks associated with it are not yet completed. And when these tasks are completed, the completed business process will no longer move further along the route.

The completion flag can be set to a business process that has not been started. In this case, its start will no longer be possible.
If the completion flag is manually removed from the completed business process, then the tasks associated with it will still remain completed. Thus, the business process will not be completed, but there will not be any uncompleted task in it. It is impossible to restart such business process, since 1C:Enterprise will consider it to have been started (there is one or more tasks in it). Therefore, when manually removing the completion flag, the completion flags of the necessary tasks should be also removed in order to return the business process to the desired route point.

13.8.2. Task execution flag

If the task execution flag is set manually, this will not cause the advancement of the business process further along the route. At the same time, the `BeforeExecute()` and `OnExecute()` event handlers of the task and at the corresponding route point will not be called either.
Manual setting the execution flag can stop the business process - it will not be completed, but there will not be any uncompleted tasks in it.
The removal of the task execution sign can cause a parallel thread to appear in an uncompleted business process. Suppose a business process is not yet completed and there is one completed and one uncompleted task. If the task execution flag is removed from a completed task, two uncompleted tasks will appear for this business process. When executing each of them, the business process will move further along the flowchart from the point corresponding to the completed task. In this case, the business process will be considered complete when all the tasks in both parallel branches are completed.
Removing an execution flag for a task which business process has already been completed will result in the task being displayed as uncompleted in the lists, but its execution will not advance the business process further along the route.

13.8.3. Removing tasks

If the uncompleted tasks of the uncompleted business process are deleted, it may stop. Such business process will be uncompleted, but there will be no active (uncompleted) task in it.
Tasks are used to display a real flowchart of a business process in order to show route points already passed and active (uncompleted) route points. Therefore, deleting tasks can cause incorrect and controversial display of the passed and active route points.
Deleting all tasks for an uncompleted business process changes its status to not started.

13.8.4. Adding tasks

If a new task for the completed business process is created manually, the business process will still be considered completed and execution of this task will not cause its advancement along the flowchart.
If a new task is manually created for a business process that has not been yet started, it will receive the started status and execution of this task will cause it to advance along the flowchart.
Creating a new task for an already started and uncompleted business process will cause its parallelization.

13.9. Condition point

A condition point is used for conditional branching of a business process. An important feature of this point is the condition check handler, the presence of which is mandatory and controlled when checking the flowchart before saving the business process. If there is no handler, a warning will be generated: Condition point has no "Condition check" event handler.
This handler must return the condition check result, which will determine the selection of the next route point. If the result is `True`, the business process will follow the Yes branch, otherwise - the No branch. By default, the result is set to `False`.
The condition check handler may, for example, look as follows:

```
Procedure DiscountLimitConditionCheck(BPRoutePoint, Result)

  If GetDiscountByInvoice() > GetRegularDiscount() Then

    Result = True;

  Else

    Result = False;

  EndIf;

EndProcedure
```

To implement a multiple choice, several serially connected condition points may be used, but it is more convenient to apply the switch point for this.

13.10. Switch

To select one of several possible paths, the switch point is used. An important feature of this point is the switch handler, the presence of which is mandatory and controlled when checking the flowchart before saving the business process. If there is no handler, a warning will be generated: Switch point has no Switch event handler.

This handler must set the `Result` parameter to one of the provided options. A handler procedure may look as follows:

```
Procedure Switch (SwitchPoint, Result)

  If PaymentType = Enumerations.PaymentType.Cash Then

    Result = SwitchPoint.Options.Cash;

  ElseIf PaymentType = Enumerations.PaymentType.Noncash Then

    Result = SwitchPoint.Options.Noncash;

  ElseIf PaymentType = Enumerations.PaymentType.Setoff Then

    Result = SwitchPoint.Options.Setoff;

  ElseIf PaymentType = Enumerations.PaymentType.Loan Then

    Result = SwitchPoint.Options.Loan;

  EndIf;

EndProcedure
```

In this handler `PaymentType` - is the business process attribute.

If no value is set for the `Result` parameter in the switch handler procedure, this will result in an error and cancellation of the transaction where the switch was attempted.

13.11. Generating tasks

Tasks are generated only when a business process arrives at activites or nested business process points. When passing other points (condition point, separation, merging, processing, etc.), the business process automatically performs the intended actions and proceeds to the next point in accordance with the flowchart.

Consider, for example, the process of transition of a business process to the activity as a result of calling the `Start()` method.

When passing along the route, the business process at activities or nested business process points can create one or more tasks. Several tasks will be generated if a "group" flag is set at the route point. In this case, the business process selects all the records corresponding to the addressing attributes established at this point, in the information register (`Addressing`) and generates its own task for each of them.

For example, if addressing at the route point is set by the `Cashier` role only, and there are two records of the type in the information register, then two tasks will be generated, which will have both addressing attributes - and the role and the end performer.

Employee	**Role**	**Division**
Ivanov	Cashier	
Petrov	Cashier	

Thus, role-based routing is applied for group route points only once - at task list generation.

Consider the sequence of calling event handlers in the 1C:Enterprise language at the time of the transition to the first route point of Invoice issuance.

No.	**Internal functionality**	**Handlers in 1C:Enterprise language**
1	Transaction start	
2		Call of `BeforeCreateTasks()` handler
3	Generation of task list	
4		Call of `OnCreateTask()` handler
5	Writing all the tasks generated	
6		Call of `BeforeWrite()` handler of a task
7	Writing of task	
8		Call of `OnWrite()` handler of a task
9	Transaction end	

The `BeforeCreateTasks()` handler call is performed in the second step. This handler is called before the task list is generated by the business process itself, therefore an empty array of the generated tasks is passed to it in so that it could be generated independently without using the standard processing.

In the third step the business process checks, if the previous handler returned `StandardProcessing = True`. If yes, then the dereferencing of the addressing attributes set at the route point is performed and one task or a list of

tasks (for a group point) is generated based on the number of dereferencing results (for example, the number of department employees). At the same time, each generated task is assigned a name, a reference to the business process and a route point and the corresponding addressing attributes.
The `OnCreateTask()` handler call is performed in the fourth step. The task list is passed to this handler, which was previously generated in the `BeforeCreateTasks()` handler or by the business process itself. The tasks have not been written yet. In this handler, it is possible to provide for fine customization of the tasks generated - setting the deadline, priority and other additional attributes. It is also possible to add new tasks to the array of tasks generated in this handler.
Normal termination of `OnCreateTask()` handler is checked in the fifth step. If the handler returns `True` in the `Fail` parameter, the task generation process is interrupted and an exception is thrown. In our case, this will cancel the execution of the `Start()` method. If `Fail = False`, all the tasks from the generated array are written with a call of `BeforeWrite()` and `OnWrite()` handlers for each individual task (steps 6 and 8, respectively).
When a business process generates an array of tasks, these will be automatically filled in with the following attributes:

- The name is set to the name of the corresponding route point, for example, Invoice issuance.
- A reference to the business process instance that generated this task.
- A reference to the route point of the business process.
- The addressing attributes of the task are set equal to the addressing attributes of the corresponding route point. For example, if the route point is addressed to the `Cashier` role, then the `Cashier` will be set to the addressing attribute of the `Role` task.

13.12. Checking performance

Tasks can be performed not only by users, but also by automated procedures. For example, if a task involves posting a document, the automatic tracking procedure for such tasks may determine that the required document has already been posted and mark the task as completed by calling its `Execute()` method.
To arrange such automated procedures, the `PerformanceCheck()` method of the task and the corresponding handlers at the route points are intended.
Consider the sequence of actions that will occur as a result of running the following code in the 1C:Enterprise language.

```
If Task.CheckExecution() Then

  Task.ExecuteTask();

EndIf
```

No.	Internal functionality	Handlers in 1C:Enterprise language
1	Processing of `CheckExecution()` method call	
2		Call of `OnCheckExecuteProcessing()` handler of the task. If `Result` is `False`, then `CheckExecution()` method immediately returns `False`
3		Call of `OnCheckExecuteProcessing()` data processor of the corresponding route point
4	Return of the handler call result from the previous step and if it is `True`, call f `ExecuteTask()` method	

13.13. Completing nested processes

When designing a flowchart it is possible to provide for the start of nested business processes. In this case, the main business process will wait for the completion of the nested business process and only then move to the next route point. When going to a route point of Nested business process type, the following sequence of actions is performed.

No.	Internal functionality	Handlers in 1C:Enterprise language
1	Transaction start	
2		`BeforeCreateSubBusinessProcesses()`

No.	Internal functionality	Handlers in 1C:Enterprise language
3		Call of `BeforeCreateTasks()` handler for a route point
4	If `StandardProcessing`, the tasks array is generated	
5		`OnCreateTask()`
6	Writing an array of generated tasks and generating an array of nested business processes	
7		`OnCreateSubBusinessProcesses()`
8	Writing and starting the generated business processes	
9	Transaction end	

Consider the above in more detail.

In the second step, the `BeforeCreateSubBusinessProcesses()` handler is called, where it is possible to add the own business processes to the generated array of business processes (the array is empty by default). If business processes have been added to the array, the standard mechanics of generating business processes will be canceled.

The `BeforeCreateTasks()` handler call is performed in the third step. An empty, not yet generated array of tasks is passed to it. If this handler does not change the standard processing, then the array of tasks generated by it will be cleared in the third step and filled with tasks (one task for each starting nested business process) with the set name and references to the business process and a route point.

In the fifth step, it is possible to "additionally customize" the tasks that have been generated and add new ones to them if necessary.

In the sixth step, the generated tasks are written, after which an nested business process of the type set at the route point is created for each of them. The generated business processes have a date and a reference to the leading task.

The `OnCreateSubBusinessProcesses()` handler call is performed in the seventh step. The handler of this event can “additionally customize” automatically generated business processes (their number is equal to the number of tasks after the `OnCreateTask()` handler) or deleting some of them, and also add new business processes to them. Writing a list of business processes will be performed after the handler is completed.

13.14. Business process completion

Completion is the last step in the business process life cycle. The business process automatically becomes completed (the `Completed` property is set to `True`) when the completion point is reached and if there are no uncompleted tasks for this business process.
If the business process has the Leading task property set, that is, it is nested, then when it is completed, it marks this task as completed. This, in turn, leads to the advancement of the main business process further along the route.
When going to the completion point, the `OnComplete()` handler is called. If it sets `Fail` to `True`, for example, if all the necessary conditions for completing the business process are not met, the processing is interrupted. At the same time the task of the route point, the execution of which caused the transition to the completion point, remains uncompleted.
Setting the `Completed` property to `True` (using 1C:Enterprise language tools or interactively) can be used to interrupt the flow of a business process or to exclude it from the list of active (uncompleted) business processes. In this case, no handlers, except `BeforeWrite` and `OnWrite`, are not called. In this case the leading task is not performed.

13.15. Standard attributes of business processes and tasks

The tables list the predefined fields of business processes and tasks.
Standard attributes of business processes:

Attribute	Type
`DeletionMark`	`Boolean`
`Number`	`String` or `Number`
`Date`	`Date`
`Completed`	`Boolean`
`HeadTask`	`TaskRef.<Task name>`
`Link`	`BusinessProcessRef.<Business process name>`

Standard attributes of tasks:

Attribute	Type
DeletionMark	Boolean
Number	String or Number
Date	Date
Name	String
Completed	Boolean
BusinessProcess	BusinessProcessRef. <Business process name>
RoutePoint	BusinessProcessRef. <Business process name>
Link	TaskRef.<Task name>

13.16. Business processes with multiple starting points

Presence of multiple starting points implies that the choice of a specific starting point is determined by the conditions external to the business process.
If the business process has all the necessary information, so it is possible to independently make a decision on the choice of a route when starting, one starting point is enough, followed by a condition check or switch point.
If the business process has several starting points, then when calling the `Start()` method it is necessary to specify a specific point. Otherwise, an error message is generated. Therefore, when creating a business process with several starting points, the following must be done:

- define interactive start commands to specify the correct starting point in each case of starting a business process.
- if this business process is nested for other business processes, then it is necessary to define the `OnCreateSubBusinessProcesses()` handler at the corresponding route points so that all the business processes from the array are generated and started from the desired point.

Example:

```
Procedure NestedAgreementWhenGeneratingNestedBPs(BPRoutePoint,
GeneratedProcesses, Fail)

  For each BusinessProcess from GeneratedProcesses Do

    BusinessProcess.Write();
    Points = BusinessProcesses.DocumentApproval.RoutePoints;
    BusinessProcess.Start(Points.SimplifiedAgreement);

  EndDo

EndProcedure
```

The other aspects of using the business processes with several starting points is not different from ordinary business processes.

13.17. Feedback

Other objects of the infobase (documents, catalog elements) may be involved in business processes and may impact them.
To effectively use the business process feature, it becomes necessary to automatically execute the corresponding tasks when performing the required operations with other objects of the infobase (for example, when posting a document, when setting a discount on the invoice issued, when reserving goods in a warehouse, etc.).
An important feature of the business processes functionality in the 1C:Enterprise is that it does not require a significant change in the applications used. Therefore, the reaction of business processes and tasks to changes in other infobase objects can be configured without significant changes to these objects.
Consider the following example. Suppose that a task requires posting of a document and it is necessary that, when a document is posted, it is carried out automatically and the user does not need to open the task list, find the necessary task in it and execute it.
To do this, consistently perform the following steps:

- Add a posting notification to the document form:

```
Procedure AfterWrite(Fail)

  If Posted Then

    Notify("DocumentPosting", , ThisObject.Reference);

  EndIf;

EndProcedure
```

- Register an notification handler. This is possible in `OnStart()` handler of the managed application module:

```
AttachNotificationHandler("NotificationHandler");
```

- Call the server common module method (for example, `UsingBusinessProcesses`) from the notification handler, which will perform the necessary checks and a task:

```
// Notification handler in the managed application module
Procedure NotificationHandler(EventName, Parameter, Source) Export

  If EventName = "DocumentPosting" Then

    UsingBusinessProcesses.CompleteTaskByDocument(Source);

  EndIf;

EndProcedure

...

&OnServer
// Method of the UsingBusinessProcesses server common module
Procedure CompleteTaskByDocument(DocumentRef) Export

    Query = New Query;
    Query.SetParameter("Param", DocumentRef);
    Query.Text =
       "SELECT
       |Reference
       |FROM
       |Task.Task.TasksByPerformer
       |
       | WHERE
       |Document = &Param";
    Selection = Query.Execute().Select();

    While Selection.Next() Loop

      CurrentTask = Selection.Reference.GetObject();
      If CurrentTask.CheckExecution() Then

        CurrentTask.ExecuteTask();

      EndIf;

    EndDo;

EndProcedure
```

Another way of automated tasks completion is to create a scheduled job (with the necessary schedule), which will select all the tasks by the required executor, check their performance and, in case of successful check, automatically perform them.

Consider the specific features of business process objects and tasks configuring.

13.18. Flowchart

A flowchart describes the logic of the business process and its entire lifecycle, from start to finish, as a sequence of route points.
To edit a flowchart, click the Flowchart button in the Miscellaneous tab of the business process editing window

13.19. Editing business process

During configuration an arbitrary number of business process types can be created. The purpose of each business process determines its structure and properties, which are described in the configuration.
Designer allows to describe the structure of the business process, create forms and a business process flowchart.
Properties of business processes are edited in the property palette or in the object editing window.
In addition to the common properties available for all the metadata objects, business processes have a number of specific properties.
The Tasks property defines a reference to a previously configured task object. A business process must necessarily be assigned one task from among those already existing in the configuration. Tasks are used by the business process to generate tasks by the performers or to start the nested business processes. If the task is not assigned, then when saving the database configuration an error will be displayed: Business process task is not selected.
Autonumbering. Setting the property causes the newly entered business process number to be automatically assigned. Users can edit automatically assigned codes.
Number length. Sets the maximum business process number length. The Designer allows to set the number length to 0 if the number is not used in this type of business process.
Number type. The property allows to select the value type for the business process number - `Number` or `String`. This property is similar to document Document type property.
The choice of the code string type is useful when a complex numbering system is used, and the number may include letters and separators in addition to numbers.
Uniqueness check. If this property is set, then when entering a new business process, its number is checked for uniqueness within the limits set in the Periodicity property.

Periodicity. The property sets the limits for controlling the uniqueness of business process numbers and the number repeatability period. If the Uniqueness check property is set, then the Periodicity property indicates the extent to which this check will be performed.
When the Autonumbering property is set, the 1C:Enterprise will assign the next sequence number to each new business process. After the end of the period set in the `Periodicity` property, the numbering of business processes will start from 1.
In the Rights tab, it is possible to set privileged mode when creating tasks (the Privileged mode when creating tasks property):

- If the property is set, 1C:Enterprise will perform all actions on the generating the tasks in the privileged mode (when executed on server side and in the file mode). However, the privileged mode will not be set if task generation is performed in the client/server mode on the thick client side.
- When creating new business processes, the property is set to `True` if the configuration properties set the default run mode - managed application, and to `False` if the default run mode is set to normal.

In addition to the basic attributes it is possible to create a set of attributes that allow storing additional information.
If the subject area object, which the business process corresponds to, has not only such “simple” properties as, for example, date, number, importance or deadline, but also composite (list) properties, such as, for example, a list of documents for approval, a list of resolutions, a list of business process participants; a set of tabular sections can be created for a business process.

13.20. Editing tasks

Task type objects are intended for issuing and executing tasks by 1C:Enterprise users. Tasks can be generated both by users and specific business processes.
Tasks can be used independently or to ensure the functioning of business processes of various types.
In the process of configuration, an arbitrary number of tasks can be created, however, as a rule, a task is created for all types of business processes.
The structure and properties of the task described in the configuration are determined by the characteristics of the subject area being automated.
For each task several forms of list, selection, viewing and editing can be created.
All tasks are characterized by number, date, time and name. When generating business processes form tasks, the name is set to the same name as the corresponding route point of the business process.
Task properties are edited in the property palette or in the object editing window.

In addition to the common properties available for all metadata objects, tasks have specific properties.

Addressing. A task can be assigned a non-periodic information register, dimensions of which can be associated with task addressing details. Such linking allows to determine the value of the main addressing attribute of the task based on the data contained in the corresponding information register, which makes not only the direct addressing of tasks to specific performers, but also role-based addressing possible.

Main addressing attribute. One of the task addressing attributes can be assigned as the main one. In this case, a specific task performer will be necessary to be specified in this addressing attribute. If the performer is not specified, the value of this addressing attribute will be determined based on the information register associated with the task (see the Addressing property).

Current performer. This property sets a reference to the session parameter where the current performer will be stored. The property is used, for example, as the default value for the Performer property of the task list table field.

Task number automatic prefix. May be set to Do not use and Business process number. If this property is set to Business process number, then when creating a new task, its number is automatically supplemented with the number of the corresponding business process.

The group of subordinate objects Addressing attributes sets a set of attributes that determine the type and dimension of the addressing system for tasks of this type in the context of a subject area being automated. One of these attributes can be set as the main one (see Main addressing attribute property). Addressing attributes can be associated with information register dimensions. This association is used by 1C:Enterprise to determine the value of the main addressing attribute, if it is not specified, and makes not only direct, but also role-based addressing possible.

Number length. Sets the maximum task number length.

Number type. The property allows to select the value type for the task number - `Number` or `String`. The choice of the code string type is useful when a complex numbering system is used, and the number may include letters and separator characters in addition to numbers.

Uniqueness check. If this property is set, then when entering a new task, its number is checked for uniqueness.

Autonumbering. Setting the property causes the newly entered task number to be automatically assigned. Users can edit automatically assigned codes.

If the subject area object, which the task corresponds to, has not only such “simple” properties as, for example, date, number, importance or deadline, but also composite (list) properties, such as, for example, a list of documents for approval, a set of tabular sections can be created for it.

Chapter 14. Data analysis and forecasting

14.1. General information

The data analysis and forecasting feature allows to implement various tools in applications to identify patterns that are usually hidden behind large amounts of information.

For example, by analyzing data on product sales, it is possible to identify groups of products that are usually bought together. In the future (one of the several options), this information can be used in the layout of products in a retail store. Prodcuts can be placed nearby (the buyer came for the brazier, saw a kindling liquid, coal, meat, fishing rods, a rubber boat ... finally bought them all), can be placed in different corners of the sales area (the buyer came for milk, and until they reach the bread, they will walk through the whole store and buy something else).

Another example of the data analysis feature use is the prediction of the counterparty behavior based on the available data on them. Having carried out such analysis, it is possible to find out how the volume of their purchases depends on the geographical location, size of the company, time of cooperation and other indicators. Based on these dependencies, it is possible to predict the behavior of a new counterparty and select the appropriate strategy for working with them.

Using the forecasting feature it is possible to plan a procurement campaign. Consider an example: the pet store sold 100 guinea pigs last month. It is required to plan the purchase of prodcuts for the next month. One of the options (very often used) is to enter a correction factor for sales of past periods. Suppose that the correction factor (demand increase coefficient) is 1.5. As a result, when planning the purchases using the described method, we will plan the purchase of 150 guinea pigs by the next month. But if you analyze what customers usually buy after buying such pets, you can arrive at a completely different conclusion. Taking advantage of the data analysis feature, building a forecast for this analysis, we can conclude that next month you need to buy fodder, various fillers, hay and other "accessories".

It should be noted that the information in this chapter will primarily concern the review of the 1C:Enterprise features and will only indirectly touch upon (using simple examples) how to use the information retrieved.

14.2. Functionality layout

The general layout of the data analysis and forecasting feature can be represented as follows:

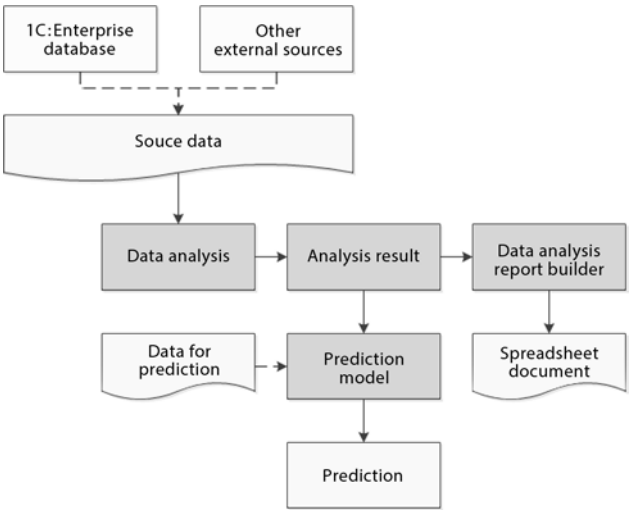


Fig. 251. Data analysis feature items interaction layout

The feature allows to work with data retrieved from the infobase and with the data obtained from another source, previously loaded to a value table or a spreadsheet document.

By applying one of analysis types to the source data, it is possible to retrieve the result of analysis. The result of the analysis is a model of data behavior; it can be displayed in the final document or saved for future use.

A further use of the analysis result is that a forecast model can be created based on it, which allows predicting the behavior of new data in accordance with the existing model.

For example, it is possible to analyze which products are bought together (in one invoice) and to save the forecast model created based on this analysis in the database. In the future, when creating a new invoice, the previously saved forecast

model can be retrieved from the infobase, input new data contained in this invoice, and retrieve a forecast as the output - i.e. the list of goods that the next customer will also buy (with a certain probability), if offered.

14.2.1. Feature main objects

Schematically, the relationship of data analysis and forecasting feature main objects
can be shown as follows:

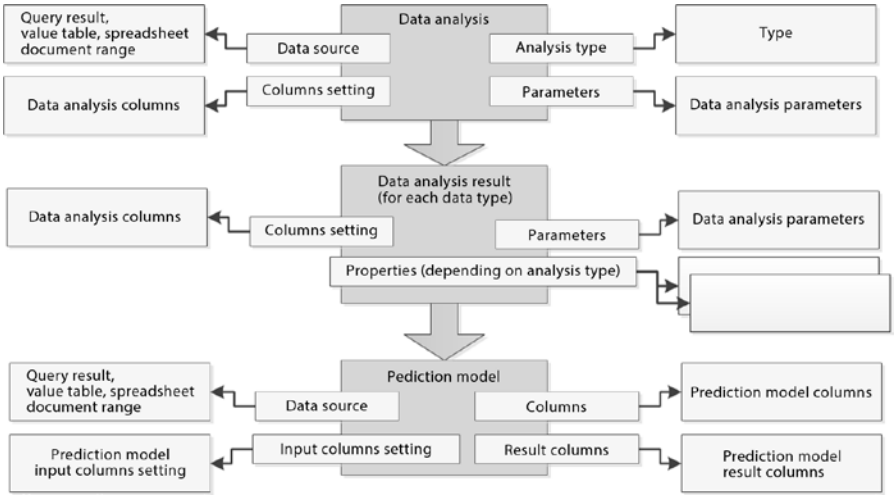


Fig. 252. Relationship of the main objects

Data analysis - the object that directly performs the data analysis. The data source, parameters and source data are set for it. The result of this object operation is the **data analysis result**, where there is own object for using with the analysis result for each type of analysis:

- `DataAnalysisSummaryStatisticsResult`,
- `DataAnalysisAssociationRulesResult`,
- `DataAnalysisSequentialPatternsResult`,
- `DataAnalysisDecisionTreeResult`,
- `DataAnalysisClusterizationResult`.

Data analysis columns settings - collection of input data analysis columns. The type of data contained, role of the column, additional settings, depending on the type of analysis are specified for each column.

Data analysis parameters - set of parameters for the data analysis performed. The composition of the parameters depends on the type of analysis. For example, for cluster analysis, the number of clusters into which the source objects are to be divided, the type of distance dimension between objects, etc. is specified.
Data source - source data for analysis. The data source can be the result of the query, cell area of the spreadsheet document, value table.
Data analysis result - specific object containing information about the analysis result. Each type of analysis has its own result. For example, Decision tree analysis type will result in `DataAnalysisDecisionTreeResult` type object. In the future, the result can be output to a spreadsheet document using the **data analysis report builder**, can be output using programmatic access to its contents, and can be used to create a **forecast model**. Any data analysis result can be saved for later use.
Forecast model - specific object that allows to perform a forecast based on the input data (selection for the forecast, selection columns and result settings, analysis result). The type of forecast model depends on the type of data analysis result. For example, a model, created for Search for associations, will be of `PredictionModelAssociationRules` type. Such model will be able to output forecasts of the following types: since this buyer has bought a given set of products, then with a certain probability they will buy another set of products. The input of the forecast model is the **forecast data source**. The result is a value table containing forecast values.
Input columns settings - a set of specific objects showing the correspondence between the columns of the forecast model and the columns of the forecast selection. For example, a Product forecast model column can correspond to a `Products` selection column.
Result columns settings - allows to control which columns will be placed in the resulting forecast model table. For example, to search for associations, it is possible to output the products that the client will most likely buy and the likelihood of such a purchase to the result.
Result columns - a value table consisting of the columns specified in the settings of the resulting columns and containing the forecast data. The specific content is determined by the type of analysis.

14.2.2. Data analysis types

The data analysis and forecasting feature implements several types of data analysis:

- general statistics,
- search for associations,
- search for sequences,

- cluster analysis,
- decision tree.

14.2.2.1. General statistics

General statistics analysis type is a feature for collecting general information about the data in the received data source. This type of analysis is intended for preliminary research of the analyzed information.
The analysis shows several characteristics of discrete and continuous fields. When a report is output to a spreadsheet document, pie charts are filled in to display the contents of the fields.

14.2.2.2. Search for associations

This type of analysis searches for groups of objects or characteristic values that are encountered together, and also searches for association rules. The search for associations can be used, for example, to determine the often purchased products or services.
This type of analysis can be used with hierarchical data, which allows, for example, to find rules not only for specific products, but also for their groups. An important feature of this type of analysis is the capability to operate both with an object data source, where each column contains a certain characteristic of the object, and with an event source, where the characteristics of the object are located in one column.

14.2.2.3. Search for sequences

Search for sequences analysis type allows to identify the sequential chains of events in the data source. For example, these may be a chain of products or services that customers often buy consistently.
This type of analysis allows to search by hierarchy, which makes it possible to track not only the sequence of specific events, but also the sequence of parent groups.

14.2.2.4. Cluster analysis

Cluster analysis allows to divide the source set of objects under study into groups of objects so that each object is more similar to objects from its group than to objects from other groups. Further analysis of the resulting groups, called clusters, makes it possible to determine what characterizes this or that group, to decide on the methods of operating the objects of different groups. For example, using cluster analysis, it is possible to divide customers with whom the company works into groups in order to apply different strategies when working with them.

Using the parameters of cluster analysis, the user can configure the algorithm that will be used for splitting, and also can dynamically change the composition of the characteristics considered in the analysis, adjust the weights for them.
The result of clusterization can be displayed in the dendrogram - a specific type of chart, designed for graphical display the results of cluster analysis.

14.2.2.5. Decision tree

Decision tree analysis type allows to build a hierarchical structure of classifying rules, represented in the form of a tree.
To build a decision tree, it is necessary to select a target attribute, which will be used to build a classifier, and a several input attributes that will be used to create rules. The target attribute may contain, for example, information about whether the client has moved to another service provider, whether the transaction was successful, whether the works have been performed efficiently, etc. Input attributes, for example, may be the age of the employee, their work experience, the financial condition of the client, number of employees in the company, etc.
The result of the analysis is presented in the form of a tree, each node of which contains some condition. To decide which class a certain object must belong to, it is necessary to go through the chain from the root to the leaf of the tree, while answering the questions in the nodes and passing to the child nodes in case of an affirmative answer and to the neighboring node - in case of a negative one.
A set of analysis parameters allows to adjust the accuracy of the resulting tree.

14.2.3. Forecast models

The forecast models generated by the feature are specific objects that are created based on the the result of data analysis and allow to automatically perform the forecast for new data in the future.
For example, the forecast model of the search for associations, built during the analysis of customer purchases, can be used when working with a purchasing customer in order to offer them the products that they will buy with a certain degree of probability together with the products they have chosen.

14.3. "General statistics" analysis type

General statistics analysis type can be used for preliminary data analysis (before performing another type of analysis), etc.
The source of the data to be analyzed can be the result of a query, a value table, or a cell area in a spreadsheet document.

The data in the source (in terms of the analysis conducted) can be continuous or discrete. The continuous ones include such types as `Number`, `Date`. The remaining types are discrete.
Obtaining different information is provided for columns of different types.
Discrete data:

- Number of values - is the number of values encountered in the data source column (`NULL` is not considered a value);
- The number of unique values (excluding duplicate values);
- Mode - the value that is most often found in the data source. If there are several values in the data that are encountered with the same frequency, the first found is taken as the mode;
- Frequency - the number of occurrences of the value in the data selection;
- Relative frequency - is defined as the ratio of the value occurrences number to the total number of values;
- Cumulative frequency - is counted as the sum of the frequency of the value and the sum of the previous values frequencies in the data selection;
- Cumulative relative frequency - is counted as the sum of the cumulative frequency of the value and the sum of the relative frequencies of the previous values.

Continuous data:

- Number of values;
- Minimum value;
- Maximum value;
- Mean value;
- Range - difference between maximum and minimum values;
- Standard deviation;
- Median - value lying in the middle of the selection.

It should be noted that if several fields of different types are analyzed simultaneously, their analysis is carried out independently of each other (mutual correlation is excluded).
Consider these characteristics in an example.
The data selection (source of analysis) has the following content:

Products	**Count**
Folding kitchen table	1
Round stool	2
Sofa "UYUT"	1
Sofa "Jeans"	1
Armchair "Jeans"	2
Table "Kitchen" 0.9x1.7	1

Products	Count
Sofa "Comfort"	1
Table "Kitchen" 0.9x1.7	1
Chair "Summer"	4
Sofa "UYUT"	1
Folding kitchen table	1
Square stool	3
Armchair "UYUT"	2
Armchair "UYUT"	2
Dresser "Wardrobe"	1
Folding kitchen table	1
Square stool	2
Dining table	1
Chair "Summer"	2
Round stool	2

As a result of the data analysis, the following characteristics will be calculated for the Count field (`Continuous` analysis data type):

Characteristic	Value
Total values	20
Min	1
Max	4
Avg	1.6
Range	3
Standard deviation	0.8208
Median	1

The following characteristics will be obtained for the Products field:

Characteristic	Value
Number of values	20
Number of unique values	12
Mode	Folding kitchen table

The frequency table for the product values will be as follows:

Frequency table

Value	Frequency	Relative frequency	Accumulated frequency	Accumulated relative frequency
Folding dining table	3	15,00	3	15,00
Round stool	2	10,00	5	25,00
Square stool	2	10,00	7	35,00
"COMFORT" armchair	2	10,00	9	45,00
"Coziness" sofa	2	10,00	11	55,00
"Kitchen" table 0.9x1.7	2	10,00	13	65,00
"Summer" armchair	2	10,00	15	75,00
Dining table	1	5,00	16	80,00
"Wardrobe" closet	1	5,00	17	85,00
"COMFORT" sofa	1	5,00	18	90,00
"Jeans" armchair	1	5,00	19	95,00
"Jeans" sofa	1	5,00	20	100,00

Fig. 253. Frequency table

Relative frequency in the form of a chart is shown below:

Frequency diagram

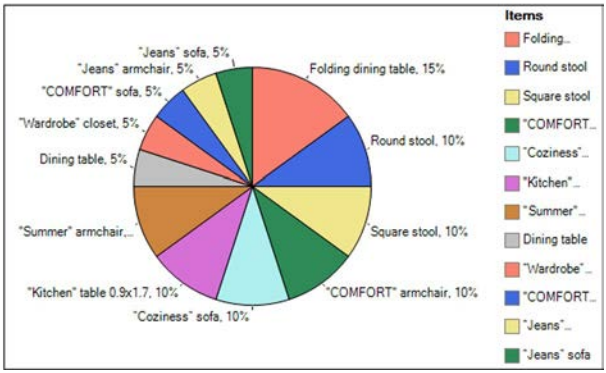


Fig. 254. Frequency chart

To perform this analysis, a code fragment similar to the one below can be used:

```
&OnClient
Procedure GeneralStatistics(Command)
  Result = AnalysisGeneralStatistics();
EndProcedure

&OnServerWithoutContext
Function AnalysisGeneralStatistics()
  Analysis = New DataAnalysis;
  Analysis.AnalysisType = Type("DataAnalysisSummaryStatistics");

  Query = New Query;
  Query.Text = "
    |SELECT
    |Sales.Products,
    |Sales.Count
    |FROM
    |AccumulationRegister.Sales AS Sales";

  Analysis.DataSource = Query.Execute();
  AnalysisResult = Analysis.Execute();

  Builder = New DataAnalysisReportBuilder();
  Builder.Template = Undefined;
  Builder.AnalysisType = Type("DataAnalysisSummaryStatistics");

  SpreadsheetDoc = New SpreadsheetDocument;
  Builder.Output(AnalysisResult, SpreadsheetDoc);
  Return SpreadsheetDoc;
EndFunction
```

The data analysis operations are performed in the server non-context function, which returns a spreadsheet document with the analysis results to the client. First, the `DataAnalysis` object is created. After this, the type of analysis performed is selected.

Further, the query is determined by the text. The query result is set as the analysis data source. The analysis itself is performed during the execution of `Execute()` method of the `DataAnalysis` object.

The analysis itself has no tools of visualizing the result of the analysis obtained. For this purpose, the `DataAnalysisReportBuilder` object is used. When creating this object, the type of analysis performed is specified again. Then, the result of the analysis received is passed as the first parameter of the `Output()` method, and the second parameter is the previously created `SpreadsheetDocument` object.

At the end of the algorithm, the spreadsheet document with the result of the analysis is returned to the client to the `Result` processing attribute of `SpreadsheetDocument` type.

As a result, data similar to above will be retrieved.

14.4. "Search for associations" analysis type

As mentioned above, this type of analysis searches for frequently encountered combinations of objects or characteristic values. Using it, it is possible to identify groups of simultaneously purchased products, identify the most attractive sources of information (in the process of "optimizing" their costs), etc.
Schematically, the Search for associations analysis process can be represented as follows:

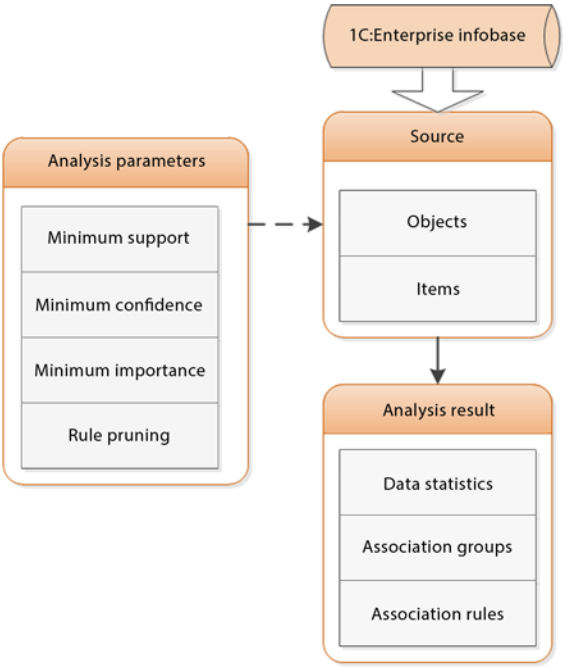


Fig. 255. “Search for associations” analysis layout

The data source can be the result of a query, a value table, or a cell area of a spreadsheet document. In terms of this analysis type, the source columns can be divided into the following:

- `NotUsed` - ignored by analysis;

- `Object` - data from this column are used as objects (or events) of the analysis conducted. Based on the value of this column, the values of the other column (`Item`) belong to the same associated group;
- `Item` - data from this column is used to obtain stable groups of values, to build associative rules.

In addition to setting the types of columns, the following analysis parameters affect the result of the analysis:

- `MinimumCasesPercentage` - the minimum percentage of cases where a combination of items should occur is determined. Groups which value is less than the specified value are not included in the analysis result;
- `MinimumConfidence` - shows the minimum percentage of cases when the rule is observed;
- `MinimumImportance` - groups for which this value is less than the specified value are not included in the analysis result;
- `RulesPruneType` - one of the options of system enumeration `AssociationRulesPruneType`:
  - `Excessive` - cut off the excessive rules,
  - `Covered` - cut off the rules, covered by the other rules.

As a result of the analysis, the following is retrieved:

- information about the data (number of objects, number of items, average number of items in the object, number of found groups, number of found association rules);
- found item groups - composition of the group, number of cases, percentage of cases in which this group occurs are indicated;
- found association rules - initial composition of the items, effect (composition of items), percentage of cases, confidence, significance of the rule are indicated.

Consider the features of this type of analysis on the following data selection (we will try to determine the composition of simultaneously purchased products):

Recorder	**Products**
Invoice No. 000000001	Folding kitchen table
	Round stool
Invoice No. 000000002	Sofa "UYUT"
Invoice No. 000000003	Sofa "Jeans"
	Armchair "Jeans"
Invoice No. 000000005	Table "Kitchen" 0.9x1.7
	Sofa "Comfort"
Invoice No. 000000004	Table "Kitchen" 0.9x1.7
	Chair "Summer"

Recorder	Products
	Sofa "UYUT"
Invoice No. 000000006	Folding kitchen table
	Square stool
Invoice No. 000000007	Armchair "UYUT"
Invoice No. 000000008	Armchair "UYUT"
Invoice No. 000000009	Dresser "Wardrobe"
Invoice No. 000000010	Folding kitchen table
	Square stool
	Dining table
Invoice No. 000000011	Chair "Summer"
	Round stool

The value of the recorder (products specified in one document is considered purchased at the same time) will be assumed as sign by which the data are attributed to one group. That is, `Recorder` will be analysis object, `Products` - will be its item.

The following code fragment will be used for the analysis:

```
&OnClient
Procedure SearchForAssociations(Command)
  Result = AnalysisSearchForAssociations();
EndProcedure

&OnServerWithoutContext
Function AnalysisSearchForAssociations()
  Analysis = New DataAnalysis;
  Analysis.AnalysisType = Type("DataAnalysisAssociationRules");

  Query = New Query;
  Query.Text = "
    |SELECT
    |Sales.Recorder,
    |Sales.Products
    |FROM
    |AccumulationRegister.Sales AS Sales";

  Analysis.DataSource = Query.Execute();

  // The string is given as an example,
  // this is the default column type value.
  Analysis.ColumnsSetting.Products.ColumnType =
DataAnalysisColumnTypeAssociationRules.Item;

  // The string is given as an example,
  // this is the default prune type value.
  Analysis.Perameters.RulePruneType.Value =
AssociationRulesPruneType.Excessive;

  AnalysisResult = Analysis.Execute();

  Builder = New DataAnalysisReportBuilder();
  Builder.Template = Undefined;
  Builder.AnalysisType = Type("DataAnalysisAssociationRules");

  SpreadsheetDoc = New SpreadsheetDocument;
  Builder.Output(AnalysisResult, SpreadsheetDoc);

  Return SpreadsheetDoc;
EndFunction
```

The result of the analysis will be as follows:

Data

Number of items:	12
Number of objects:	11
Average number of items in an object:	1,82

Analysis result

Frequent groups found:	1
Association rules found:	2

Fig. 256. "Search for assorications" analysis result

The selection uses data from eleven documents (the reference is contained in the Recorder field), number of different items is twelve:

Products
Folding kitchen table
Round stool
Sofa "UYUT"
Sofa "Jeans"
Armchair "Jeans"
Table "Kitchen" 0.9x1.7
Sofa "Comfort"
Chair "Summer"
Square stool
Armchair "UYUT"
Dresser "Wardrobe"
Dining table

The following product group is found:

Frequent groups

#	Number of cases	Percentage of cases
Contents		
1	2	18,18
Item = Square stool		
Item = Folding dining		

Fig. 257. Found product group

The whole group is found in the document only in two cases out of eleven (this is shown in the Number of cases and Percentage of cases columns).

The following association rules were retrieved:

Association rules

Source set	Consequent	Percentage of cases	Confidence	Importance
1		18,18	100,00	3,67
Item = Square stool	Item = Folding dining table			
2		18,18	66,67	3,67
Item = Folding dining table	Item = Square stool			

Fig. 258. Association rules

Let us analyze the second of them. In two cases out of eleven in the document, the Rectangular stool item was encountered along with the Kitchen folding table item. Based on this, the percentage of cases was calculated: (2 / 11 * 100 = 18.18 %).
Confidence was calculated as follows: both product items were purchased in two cases, the commodity item Kitchen folding table was found 3 times in purchases. Based on this the confidence is: 2 / 3 * 100 = 66.67 %.
Importance is defined as the ratio of the rule confidence to the percentage of finding a Rectangular stool in the purchased products. This item is found in two of the eleven documents (18.18%). Importance equals: 66.67 % / 18.18 % = 3.67.

14.4.1. Rule prune types

Consider such important parameter of this type of analysis, as the `RulesPruneType`. Possible prune values are contained in the system enumeration `AssociationRulesPruneType`, they contain:

- `Covered`,
- `Excessive`.

Before proceeding to consider the features of the prune options, consider a few general points applicable to the association rules.
Any rule consists of **prerequisite** and **consequence**. Example:

- Prerequisite: **If** Product 1 is bought.
- Consequence: **Then** Product 2 will be bought.

At the same time, it is necessary to remember that the consequence occurs with a certain confidence. When pruning rules, probabilistic characteristics can be considered, or they can be ignored (only the content of the rule is important).

14.4.1.1. Pruning covered rules

Consider the Covered pruning option.
The rule can be covered both by the prerequisite and by the consequence. Example:

- Rule 1: **If** products 1 and 3 have been bought, **Then** product 2 will be bought.
- Rule 2: **If** product 1 has been bought, **Then** product 2 will be bought.

In this case, rule 1 will be considered covered, since the prerequisite of the first rule is “excessive” in relation to the prerequisite of the second rule.
Example of coverage by consequence:

- Rule 1: **If** product 1 has been bought, **Then** products 2, 3 will be bought.
- Rule 2: **If** product 1 has been bought, **Then** product 3 will be bought.

Rule 2 will be covered by consequence, since the consequence of rule 1 is more complete.

14.4.1.2. Pruning excessive rules

Coverage does consider the probabilistic characteristics of the rules, these are considered if the `Excessive` prune option is used.
The rule will be considered excessive by the prerequisite if it is covered by the prerequisite and its confidence is equal to the confidence of the covering rule. Example:

- Rule 1: **If** products 1 and 3 have been bought, **Then** product 2 will be bought with confidence of 75%.
- Rule 2 **If** product 1 has been bought, **Then** product 2 will be bought will confidence of 75 %.

Rule 1 is excessive with respect to rule 2 (it contains an additional condition that does not introduce a “disturbance” to the probabilistic characteristics of the rule).
Rule 1 will be considered excessive by consequence if the number of cases of this rule is equal to the number of cases of the covering rule.

- Rule 1: **If** product 1 has been bought, **Then** products 2, 3 will be bought in three cases.
- Rule 2: **If** product 1 has been bought, **Then** product 3 will be bought in three cases.

Rule 2 will be considered excessive with respect to rule 1, since it contains a simpler consequence with the same probabilistic characteristics.

14.5. "Search for sequences" analysis type

This type of analysis allows to identify chains of occurring events (sequence templates). It can be used when one of the important analyzed indicators is the sequence of events occurrence in time. For example, it is possible to identify the sequence of products that are purchased one after another for a certain period of time, etc.

Schematically, the Search for sequences analysis process can be represented as follows:

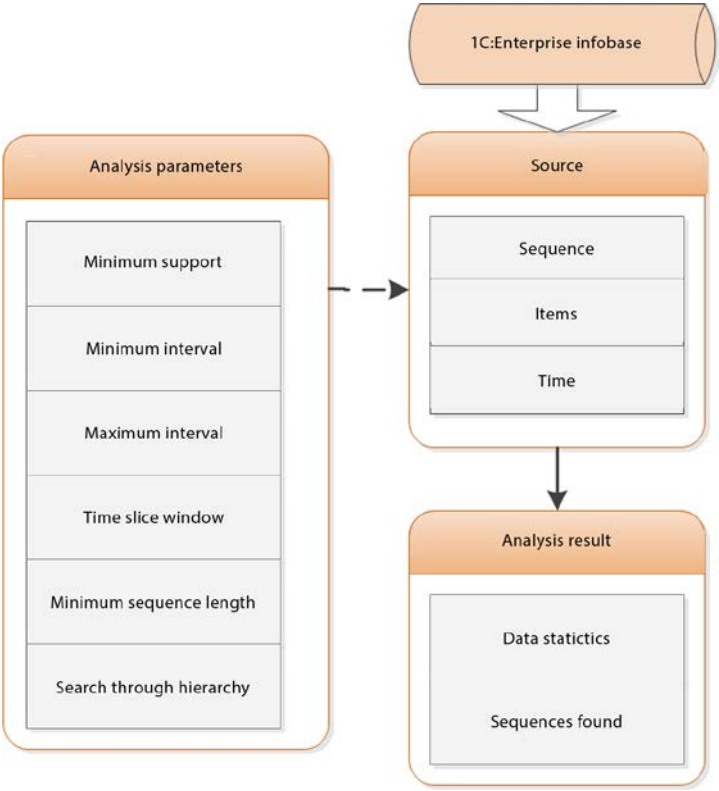


Fig. 259. "Search for sequences" analysis layout

The data source can be the result of a query, a value table, a cell area of a spreadsheet document. In terms of this analysis type, the source columns can be divided into the following:

- `NotUsed` - ignored by analysis;
- `Sequence` - data from this column is used for analysis as a sequence event object. Based on the value of this column, analysis associates the data with a single chain of events;
- `Item` - data from this column is used as sequence items;
- `Time` - this column determines the time of the event. Presence of this column is mandatory when conducting this type of analysis.

In addition to setting the types of columns, the following analysis parameters affect the result of the analysis:

- `MinimumCasesPercentage` - minimum percentage of sequences where the found sequence template is observed;
- `MinimumInterval` - flag of setting the minimum interval of the sequence (the interval unit of measurement and unit conversion factor must be set);
- `MaximumInterval` - flag of setting the maximum interval of the sequence (the interval unit of measurement and unit conversion factor must be set);
- `TimeEquivalenceInterval` - flag of setting the time equivalence interval (the time equivalence interval unit of measurement and unit conversion factor must be set);
- `MinimumLength` - minimum length of the desired sequences;
- `SearchByHierarchy` - flag of performing search by heirarchy (applies to `Item` type columns).

Several properties operate such notion as `DataAnalysisTimeIntervalUnitType`. This system enumeration contains the following values:

`Second`	
`Minute`	`CurrentMinute`
`Hour`	`CurrentHour`
`Day`	`CurrentDay`
`Week`	`CurrentWeek`
`Tendays`	`CurrentTenDays`
`Month`	`CurrentMonth`
`Quarter`	`CurrentQuarter`
`Halfyear`	`CurrentHalfYear`
`Year`	`CurrentYear`

The main result of the analysis is the sequence templates found. These templates contain the following information:

- the sequence template composition;
- the number of cases where this sequence was observed;
- maximum intervals between events (if there are 2 events, then there is one interval);
- minimum intervals between events (if there are 2 events, then there is one interval);
- percentage of cases when this sequence was executed;

- average intervals between events (if there are 2 events, then there is one interval).

We will consider the specifics of performing this type of analysis on the example of the following data selection:

Counter-party	**First purchase**	**Second purchase**	**Third purchase**	**Interval**
Bondarev V.I.	Folding kitchen table	Sofa "UYUT"	Armchair "UYUT"	25 days, 31 days
	Round stool			
Ivanov I.P.	Sofa "Jeans"			
	Armchair "Jeans"			
Petrov B.S.	Table "Kitchen" 0.9x1.7	Armchair "UYUT"		43 days
	Chair "Summer"			
	Sofa "UYUT"			
Sidorov G.O.	Folding kitchen table			
	Square stool			
Stepanov V.K.	Folding kitchen table			
	Square stool			
	Dining table			
Fedorov D.E.	Table "Kitchen" 0.9x1.7	Dresser "Wardrobe"	Chair "Summer"	58 days, 29 days
	Sofa "Comfort"		Round stool	

The data from `Counterparty` column will determine membership in a specific chain of events, i.e. these determine the sequence of analysis. `Products` will be the retrieved sequence item.

For analysis, a code fragment similar to the one below can be used:

```
&OnClient
Procedure SearchForSequences(Command)
  Result = AnalysisSearchForSequences();
EndProcedure

&OnServerWithoutContext
Function AnalysisSearchForSequences()
  Analysis = New DataAnalysis;
  Analysis.AnalysisType = Type("DataAnalysisSequentialPatterns");

  Query = New Query;
  Query.Text = "
    |SELECT
    |Sales.Counterparty,
    |Sales.Products,
    |Sales.Period
    |FROM
    |AccumulationRegister.Sales AS Sales";

  Analysis.DataSource = Query.Execute();

  Analysis.ColumnsSetting.Period.ColumnType =
DataAnalysisColumnTypeSequentialPatterns.Time;
  AnalysisResult = Analysis.Execute();

  Builder = New DataAnalysisReportBuilder();
  Builder.Template = Undefined;
  Builder.AnalysisType = Type("DataAnalysisSequentialPatterns");

  SpreadsheetDoc = New SpreadsheetDocument;
  Builder.Output(AnalysisResult, SpreadsheetDoc);
  Return SpreadsheetDoc;
EndFunction
```

`Period` field is determined as `Time` directly from the code (not determined by the analysis independently).

The default analysis parameters are:

Analysis parameters

Minimum percentage of cases:	10
Minimum interval:	---
Maximum interval:	---
Time slice window:	---
Minimum sequence length:	2

Fig. 260. Analysis parameters

After analysis, the following general data were retrieved:

Data breakdown:	
Number of items:	12
Number of sequences:	6
Analysis result	
Frequent groups found:	1
Association rules found:	2

Fig. 261. General analysis data

The number of items is twelve. Exactly that many product items are found in the above data selection.
Two sequences have been found:

#	Number of cases	Percentage of cases	Average interval	Minimum interval	Maximum interval
Contents					
1	2	33,33			
Items = "Coziness" sofa					
Items = "COMFORT" armchair			1 mon. 7 d.	1 mon. 1 d.	1 mon. 13 d.
2	1	16,67			
Items = "Kitchen" table 0.9x1.7					
Items = "Summer" armchair			2 mon. 28 d.	2 mon. 28 d.	2 mon. 28 d.

Fig. 262. Found sequences

The first sequence occurs in two cases out of five. Proceeding from this, the percentage of cases is 40%. Since the depth of a sequence is 2, there is one value for each of the reducible intervals.

14.6. "Decision tree" analysis type

Using this type of analysis, a causal hierarchy of conditions can be obtained, leading to a specific solution. For example, to obtain a condition tree under which (with a certain probability) it is possible to understand the reason for termination of contracts with the customers of the company, determining the conditions affecting the option of the contract being signed. It is possible to carry out "profiling" of company managers for various types of its clients, etc.

Schematically, the Decision tree analysis process can be represented as follows:

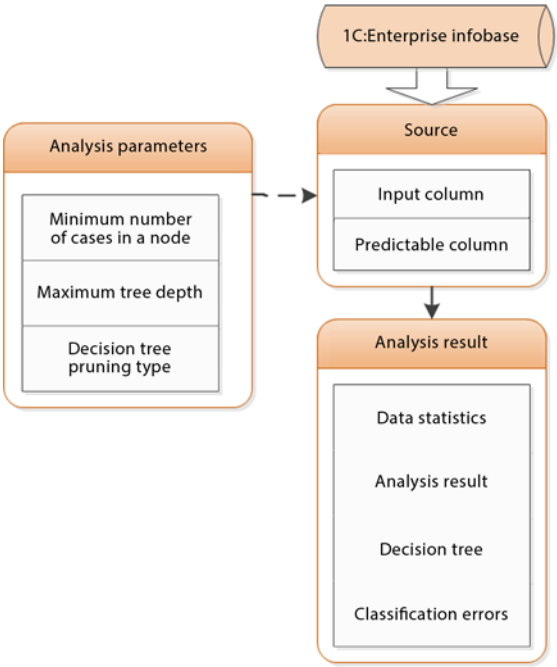


Fig. 263. "Decision tree" analysis layout

In terms of this analysis type, the source columns can be divided into the following:

- `NotUsed`,
- `Input`,
- `Forecast`.

Applied analysis parameters:

- `MinimumCasesNumber` - minimum number of items in the node;
- `MaximumDepth` - maximum tree depth;
- `SimplificationType` - decision tree simplification type.

As a result of the analysis, the following can be retrieved:

- decision tree,
- classification errors.

Let us deal with the features of this type of analysis on the example of the following data selection:

Counterparty	**Number of retail shops**	**Number of vehicles**	**Company operation time**	**Contract signing time**	**Contract type**	**Relations condition**
ZAO Igor	1	0	Less than a year	Less than a year	Dealer	Contract violation
ZAO TogrMebel	15	4	From three to ten years	Less than a year	Distributor	Terminated by counterparty
ZAO TogrMebel	1	10	From three to ten years	From one to three years	Distributor	Terminated by counterparty
IPE Dubrava	1	1	From one to three years	Less than a year	Dealer	Terminated by counterparty
Shop No. 15	1	1	Over ten years	From three to ten years	Permanent partner	Not terminated
OOo Gross	3	2	Less than a year	Less than a year	Permanent partner	Not terminated
OOO Intaris	7	3	From three to ten years	From one to three years	Permanent partner	Terminated by counterparty
OOO TogrTrest	2	2	Over ten years	From three to ten years	Permanent partner	Not terminated
PBOYL Kurochkin	0	1	Less than a year	Less than a year	Dealer	Not terminated

For analysis, a code fragment similar to the one below can be used:

```
&OnClient
Procedure DecisionTree(Command)
  Result = AnalysisDecisionTree();
EndProcedure

&OnServerWithoutContext
Function AnalysisDecisionTree()
  Analysis = New DataAnalysis;
  Analysis.AnalysisType = Type("DataAnalysisDecisionTree");

  Group = Catalogs.Counterparties.FindByDescription("Legal entities");
  Query = New Query;
  Query.Text = "
    |SELECT
    |Counterparties.Reference,
    |Counterparties.RetailShopsNumber,
    |Counterparties.VehicleNumber,
    |Counterparties.CompanyOperationTime,
    |Counterparties.ContractSigningTime,
    |Counterparties.ContractType,
    |Counterparties.RelationsTermination
    |FROM
    |Catalog.Counterparties AS Counterparties
    | WHERE
    |(Not Counterparties.IsFolder AND Counterparties.Parent =
&Parent)";

  Query.SetParameter("Parent", Group);

  Analysis.DataSource = Query.Execute();

  Analysis.Parameters.SimplificationType.Value =
DecisionTreeSimplificationType.DontSimplify;
  AnalysisResult = Analysis.Execute();

  Builder = New DataAnalysisReportBuilder();
  Builder.Template = Undefined;
  Builder.AnalysisType = Type("DataAnalysisDecisionTree");

  SpreadsheetDoc = New SpreadsheetDocument;
  Builder.Output(AnalysisResult, SpreadsheetDoc);
  Return SpreadsheetDoc;
EndFunction
```

The analysis resulted in the following decision tree:

Decision tree

Condition	Decision	Probability	Count
Root	ContractTermination = Not terminated	44.44	9
NumberOfCars <= 0.5	**ContractTermination = Contract infringement**	**100.00**	**1**
NumberOfCars > 0.5	ContractTermination = Not terminated	50.00	8
NumberOfCars <= 2.5	ContractTermination = Not terminated	80.00	5
CompanyAge = Over ten years	**ContractTermination = Not terminated**	**100.00**	**2**
CompanyAge = Less than a year	**ContractTermination = Not terminated**	**100.00**	**2**
CompanyAge = One to three years	**ContractTermination = Terminated by contractor**	**100.00**	**1**
NumberOfCars > 2.5	**ContractTermination = Terminated by contractor**	**100.00**	**3**

Fig. 264. Decision tree

This tree can be represented in the form of the following layout:

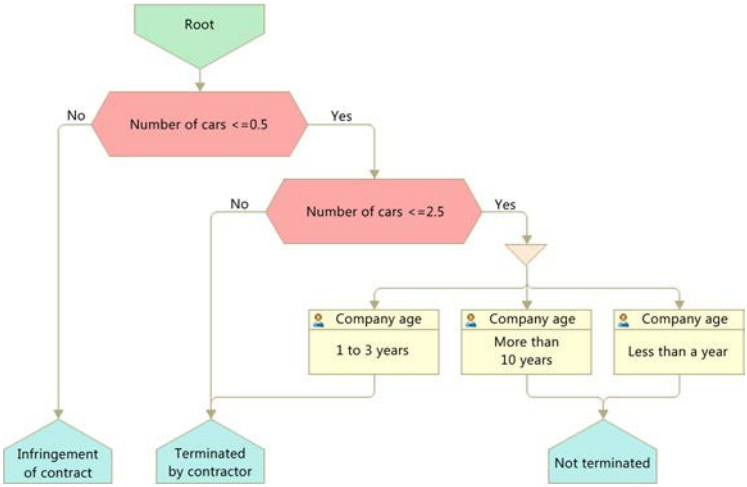


Fig. 265. Decision tree representation as a scheme

Classification errors show in which cases the obtained rules diverge from the reality (initial data selection):

Classification errors

Actual\Classified	ContractTermination = Not terminated	ContractTermination = Contract infringement	ContractTermination = Terminated by contractor	Total	Error percentage
ContractTermination = Not termina	4			4	
ContractTermination = Contract in		1		1	
ContractTermination = Terminatec			4	4	
Total	4	1	4	9	

Fig. 266. Classification errors

Based on the above data, it is clear that there are no errors in the resulting classification, that is, the data in the actual selection match the classification data.
The example above has been obtained based on the `DontSimplify` value of `SimplificationType` analysis parameter. This parameter value is set programmatically in the example above. If the value of the `Simplify` parameter is set, the decision tree will look as follows:

Decision tree

Condition	Decision	Probability	Count
Root	ContractTermination = Not terminated	44.44	9
NumberOfCars <= 0.5	**ContractTermination = Contract infringement**	**100.00**	**1**
NumberOfCars > 0.5	ContractTermination = Not terminated	50.00	8
NumberOfCars <= 2.5	**ContractTermination = Not terminated**	**80.00**	**5**
NumberOfCars > 2.5	**ContractTermination = Terminated by contractor**	**100.00**	**3**

Fig. 267. Decision tree

Simplification of the tree is that according to certain rules (formulas, which will be discussed below) tree nodes are turned into leaves (unnecessary branching is cut off).
When deciding whether a node is to be converted to a leaf, the following indicators are considered:

- `Errors` - number of errors in the node;
- `ChildErrors` - number of errors in child nodes;

- Leaves - number of leaves in a node;
- Cases - number of cases.

$$\text{StandardDeviation} = \sqrt{\frac{\left(\text{ChildErrors} + \frac{\text{Leaves}}{2}\right) * \left(\text{Cases} * \left(\text{ChildErrors} + \frac{\text{Leaves}}{2}\right)\right)}{\text{Cases}}}$$

The decision to convert a node to a leaf is made if the following condition is met:

$$\text{Errors} + 0.5 < \text{ChildErrors} + \frac{\text{Leaves}}{2} + \text{StandardDeviation}$$

In our example, for the Company operation time nodes, the condition is met (0.5 <1).

Due to the use of simplifications, classification errors also appeared, which is shown in the analysis result:

Classification errors

Actual\Classified	ContractTermination = Not terminated	ContractTermination = Contract infringement	ContractTermination = Terminated by contractor	Total	Error percentage
ContractTermination = Not terminated	4			4	
ContractTermination = Contract infringement		1		1	
ContractTermination = Terminated by contractor	1		3	4	25.00
Total	5	1	3	9	11.11

Fig. 268. Classification errors

For example, there is one case where there was a Terminated by the counter-party value in the real data selection, and according to the retrieved classification there should be Not terminated value, etc.

14.7. "Clusterization" analysis type

Cluster analysis - is a mathematical procedure for multidimensional analysis, which allows to group the object into clusters based on the set of indicators characterizing several objects, so that the objects in one cluster are more homogeneous, similar, compared to objects in other clusters.

The basis of this analysis is the calculation of distance between objects. Based on the distances between the objects these are grouped into clusters. The distance can be determined in different ways (according to different metrics). The following metrics are available:

- Euclidean metric,
- Squared Euclidean metric,
- City block metric,
- Maximum metric.

After determining the distances between objects, one of several algorithms for distributing objects among clusters can be used. The following clustering methods are available:

- Nearest neighbour,
- Furthest neighbour,
- k-means,
- Cetroid.

Schematically, the functionality of cluster analysis can be represented as follows:

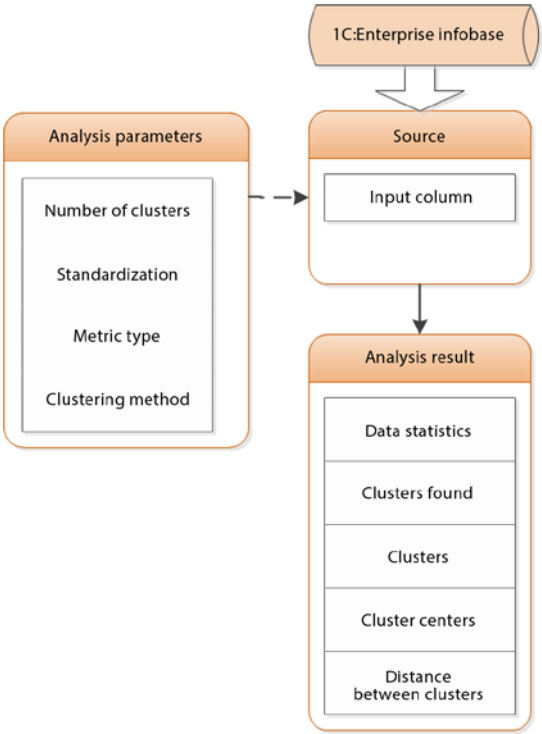


Fig. 269. Cluster analysis layout

The data source is passed to the `DataAnalysis` object. The data source can be the result of a query, a value table, a cell area of a spreadsheet document. Source columns are defined as input or unused. It should be noted that all the column values are contained in the `DataAnalysisColumnTypeClusterization` system enumeration. This enumeration contains more values (not only unused and input ones), but the other values are used when building forecasts.

The analysis is performed in accordance with the set analysis parameters.

We will use the following code fragment as an example illustrating the capability of cluster analysis:

```
&OnClient
Procedure ClusterAnalysis(Command)
  Result = AnalysisClusterization();
EndProcedure

&OnServerWithoutContext
Function AnalysisClusterization()
  Analysis = New DataAnalysis;
  Analysis.AnalysisType = Type("DataAnalysisClusterization");

  Group = Catalogs.Counterparties.FindByDescription("Legal entities");
  Query = New Query;
  Query.Text = "
    |SELECT
    |Counterparties.Reference,
    |Counterparties.RetailShopsNumber,
    |Counterparties.VehicleNumber,
    |Counterparties.CompanyOperationTime,
    |Counterparties.ContractSigningTime,
    |Counterparties.ContractType,
    |Counterparties.RelationsTermination
    |FROM
    |Catalog.Counterparties AS Counterparties
    | WHERE
    |(Not Counterparties.IsFolder AND Counterparties.Parent =
&Parent)";

  Query.SetParameter("Parent", Group);

  Analysis.DataSource = Query.Execute();

  // Selecting metric.
  Analysis.Parameters.DistanceMeasure.Value =
DataAnalysisDistanceMetricType.SquaredEuclidean;

  // Selecting clusterization method.
  Analysis.Parameters.ClusterizationMethod.Value =
ClusterizationMethod.KMeans;

  AnalysisResult = Analysis.Execute();

  Builder = New DataAnalysisReportBuilder();
  Builder.Template = Undefined;
  Builder.AnalysisType = Type("DataAnalysisClusterization");

  SpreadsheetDoc = New SpreadsheetDocument;
  Builder.Output(AnalysisResult, SpreadsheetDoc);

  Return SpreadsheetDoc;
EndFunction
```

Query is performed by the `Counterparties` catalog. According to the query condition, only detailed catalog entries from the Legal entities group are selected.
Execution of the above code will result in the following values being defined as the initial data analysis settings (some of them set explicitly, some of them - are defaults):

Analysis parameters

Number of clusters	3
Standartization	Standartize
Metric type	Squared euclidean
Clustering method:	Centroid

Data source columns

Input columns

Column name	Data type	Weight
Ref	Discrete	1
NumberOfRetailStores	Contiguous	1
NumberOfCars	Contiguous	1
CompanyAge	Discrete	1
CotractSignTime	Discrete	1
ContractType	Discrete	1
ContractTermination	Discrete	1

Fig. 270. Analysis parameters

The composition of the columns was determined based on the composition of the query selection fields. By default they are defined with equal weight. For `Number` and `Date` types the `Continous` data type has been defined, for other types - `Discrete`. If it is necessary to change the parameters of the columns, this can be done by analogy with the fragment below:

```
Analysis.ColumnsSetting.VehicleNumber.AdditionalParameters.Weight = 2;
```

In this line, the weight is increased for the `VehicleNumber` column.
The selection of data for which the analysis will be performed has the following content:

Counterparty	Number of retail shops	Number of vehicles	Company operation time	Contract signing time	Contract type	Relations condition
ZAO Igor	1	0	Less than a year	Less than a year	Dealer	Contract violation

Counter-party	**Number of retail shops**	**Number of vehicles**	**Company operation time**	**Contract signing time**	**Contract type**	**Relations condition**
ZAO TogrMebel	15	4	From three to ten years	Less than a year	Distributor	Terminated by counterparty
ZAO TogrMebel	1	10	From three to ten years	From one to three years	Distributor	Terminated by counterparty
IPE Dubrava	1	1	From one to three years	Less than a year	Dealer	Terminated by counterparty
Shop No. 15	1	1	Over ten years	From three to ten years	Permanent partner	Not terminated
OOo Gross	3	2	Less than a year	Less than a year	Permanent partner	Not terminated
OOO Intaris	7	3	From three to ten years	From one to three years	Permanent partner	Terminated by counterparty
OOO TogrTrest	2	2	Over ten years	From three to ten years	Permanent partner	Not terminated
PBOYL Kurochkin	0	1	Less than a year	Less than a year	Dealer	Not terminated

The result of the analysis will be obtained in the following form:

Data breakdown

Number of objects: 9

Analysis result

Clusters found: 3

Clusters

Cluster #	Object count	Percentage
1	6	66.67
2	2	22.22
3	1	11.11

Cluster centers

Field \ Cluster	1	2	3
Ref = ZAO TorgMebel			1.0000
Ref = ZAO Mebelcraft		0.5000	
Ref = OOO Intaris		0.5000	
Ref = PBOUL Kurochkin	0.1667		
Ref = OOO TorgTrest	0.1667		
Ref = Store 15	0.1667		
Ref = ZAO Igor	0.1667		
Ref = OOO Gross	0.1667		
Ref = ICP Dubrava	0.1667		
NumberOfRetailStores	1.3333	11.0000	1.0000
NumberOfCars	1.1667	3.5000	10.0000
CompanyAge = One to three years	0.1667		
CompanyAge = Less than a year	0.5000		
CompanyAge = Over ten years	0.3333		
CompanyAge = Three to ten years		1.0000	1.0000
CotractSignTime = Three to ten years	0.3333		
CotractSignTime = Less than a year	0.6667	0.5000	
CotractSignTime = One to three years		0.5000	1.0000
ContractType = Long-term partner	0.5000	0.5000	
ContractType = Dealer	0.5000		
ContractType = Distributor		0.5000	1.0000
ContractTermination = Not terminated	0.6667		
ContractTermination = Contract infringement	0.1667		
ContractTermination = Terminated by contractor	0.1667	1.0000	1.0000

Distance between clusters

Clusters	1	2	3
1		8.7740	15.4524
2	8.7740		11.5370
3	15.4524	11.5370	

Fig. 271. Cluster analysis result

Note that data is retrieved on the clusters found (their number, centers, distances between them) as a result of the analysis. The analysis does not result in obtaining the data on which objects (in our case, counterparties) are included in which clusters. This behavior is observed if the parameters of the analysis performed are not explicitly set (namely, the `TableFillingType` parameter).

In order to see the distribution of objects in clusters as a result of the analysis, it is necessary to define the following line of code before performing the analysis (but after determining its type):

```
Analysis.Parameters.TableFillingType.Value =
DataAnalysisResultTableFillType.UsedFields;
```

14.7.1. Used metrics

Immediately note the fact: despite the fact that in the previous example the input columns were of continuous type (for this type the concept of "distance" is obvious), the analysis can also use columns of discrete types (references to catalogs, enumeration values, etc.).
Let us consider the metrics that can be used when conducting cluster analysis.

14.7.1.1. Euclidean metric

In this metric, the distance between two objects is calculated by the formula:

$$\text{Distance} = \sqrt{\sum (W_i * (X_i - Y_i)^2)}$$

Where:

- Xi, Yi - attribute values of two objects (between which the distance is determined);
- Wi - attribute weight factor (set in the analysis column);
- i - attribute number, from 1 to n;
- n - number of attributes.

Suppose that objects are characterized by one property, which is 9 for one object and 5 - for another. The weight factor of this attribute is one. The distance between the objects will be:

$$\sqrt{\sum (1_1 * (9_1 - 5_1)^2)} = \sqrt{\sum (1 * (4)^2)} = 4$$

14.7.1.2. Squared Euclidean metric

In this metric, the distance between two objects is calculated by the formula:

$$\text{Distance} = \sum (W_i * (X_i - Y_i)^2)$$

Where:

- Xi, Yi - attribute values of two objects (between which the distance is determined);
- Wi - attribute weight factor (set in the analysis column);
- i - attribute number, from 1 to n;
- n - number of attributes.

Suppose that objects are characterized by one property, which is 5 for one object and 3 - for another. The weight factor of this attribute is two. The distance between the objects will be:

$$\sum(2_1*(5_1-3_1)^2) = \sum(2+1*(2)^2) = 8$$

14.7.1.3. City block metric

In this metric, the distance between two objects is calculated by the formula:

$$\text{Distance} = \sum(W_i*|X_i-Y_i|)$$

Where:

- Xi, Yi - attribute values of two objects (between which the distance is determined);
- Wi - attribute weight factor (set in the analysis column);
- i - attribute number, from 1 to n;
- n - number of attributes.

Suppose that objects are characterized by two attributes that have values 3 and 5, 7 and 3. The weight of the first is 2, the weight of the second is 1:

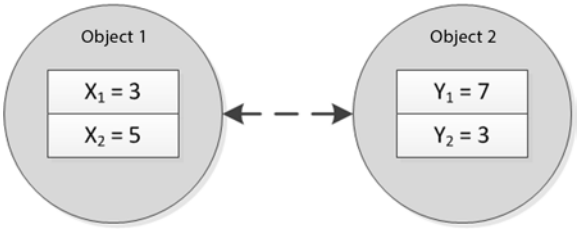


Fig. 272. Object characteristics

$$(W_1*|X_1-Y_1|)+(W_2*|X_2-Y_2|) = (2*|3-7|)+(1*|5-3|) = 10$$

14.7.1.4. Maximum metric

In this metric, the distance between two objects is calculated by the formula:

$$\text{Distance} = MAX(W_i*|X_i-Y_i|)$$

Where:

- Xi, Yi - attribute values of two objects (between which the distance is determined);

- Wi - attribute weight factor (set in the analysis column);
- i - attribute number, from 1 to n;
- n - number of attributes.

Suppose that objects are characterized by two attributes that have values 3 and 5, 7 and 3. The weight of the first is 2, the weight of the second is 1 (see fig. 272).

$$MAX(W_1*|X_1-Y_1|, W_2*|X_2-Y_2|) = MAX(2*|3-7|, 1*|5-3|) = 8$$

14.7.2. Clusterization methods

A clustering method option determines, based on what principles an object is attributed to one group or another, according to which algorithm generation of clusters is performed.

It can be said that the goal of any clustering algorithm is:

- minimization of variability within clusters,
- maximization of variability between clusters.

Differences between the methods will be considered on the objects presented in the figure (see fig. 273).

Assume that the objects form two groups. The first one consists of objects 1, 2 and 3. The second group consists of objects 4, 5 and 6.

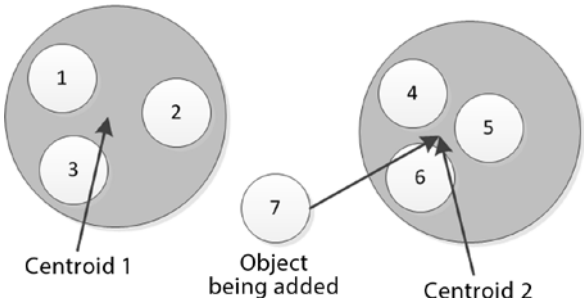


Fig. 273. Object groups

14.7.2.1. Nearest neighbour

The clustering method where the object joins the group for which the distance to the nearest object is minimal.

In this example, object 7 will be included in the group in which object 4 is located. The closest objects of the two groups are objects 4 and 3. The distance to the object 4 is minimal.

14.7.2.2. Furthest neighbour

The clustering method where the object joins the group for which the distance to the furthest object is minimal.
In this example, object 7 will be included in the group in which object 5 is located. The closest objects of the two groups are objects 1 and 5. The distance to the object 5 is less.

14.7.2.3. Cetroid

The clustering method where the object joins the group for which the distance to the center of gravity is minimal:

In the example in the figure, object 7 will be added to the group containing objects 4, 5 and 6. The distance to the center of gravity (of some mythical object with average values of attributes) is minimal.

14.7.2.4. k-means

In this method, the objects that are first in the selection are selected. They are considered cluster centers. Next, the next object is selected and, in accordance with the distance to the centers of the clusters, it is attributed to a particular cluster. The center of the cluster which the object was added to is recalculated.
The procedure is repeated until the exhaustive search of all the objects. Then a new selection of objects is made again (starting with the first one). The procedure is repeated as long as the centers of the clusters change:

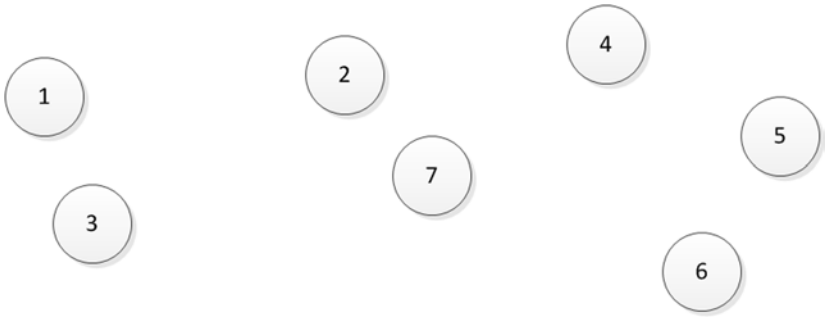


Fig. 274. Objects location example

Suppose that objects 1 and 2 are arbitrarily chosen as centers of clusters. Object 3 is added to a cluster which center is object 1. The center of the first cluster is recalculated (it is located between object 1 and 3). Object 4 is added to the second cluster (its center is also recalculated).

After iterating through all the analyzed objects, objects 1 and 3 belong to the first cluster, other objects - belong to the second cluster (its center is presumably located in the center of a triangle of objects 4, 7, 6).
Then again, objects are selected and distributed among clusters (relatively constantly calculated cluster centers).
Somewhere on the third sample of objects, most likely, object 2, which was originally the center of the second cluster, will be attributed to the first cluster.
At the end of the algorithm, objects 1, 2, 3 will be assigned to the first cluster. Objects 4, 5, 6, 7 - will be assigned to the second cluster.

14.7.2.5. Data output to a dendrogram

When outputting cluster analysis data, if an algorithm other than the k-means is used, the results of cluster analysis are output in the form of a dendrogram (the analysis algorithm must provide for outputting the distribution of the analyzed objects by clusters):

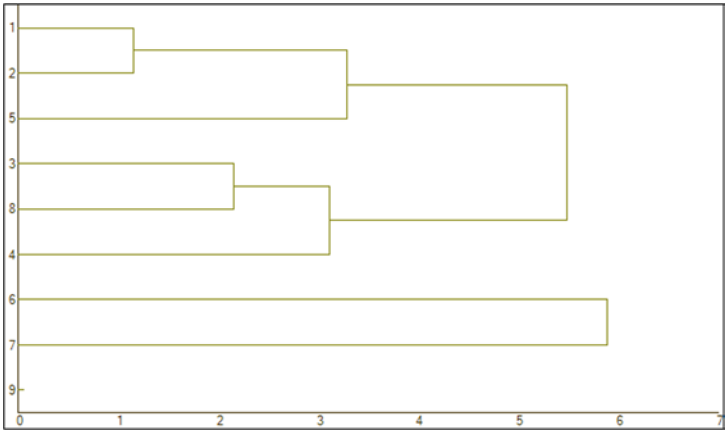


Fig. 275. Dendrogram

Chapter 15. Data exchange features

15.1. Tasks and goals

15.1.1. General information

Data exchange features - are a set of 1C:Enterprise tools for arranging data exchange between various infobases, as well as between infobases and external software systems. Data exchange features can be divided into two levels:

- universal data exchange features,
- distributed infobases.

15.1.2. Universal data exchange features

Universal data exchange features can be used both together and separately, in various combinations, for arranging the data exchange from the 1C:Enterprise infobases with various software systems. Other 1C:Enterprise infobases can act as software systems which the exchange is arranged with. In this case, the infobases exchanging among themselves may in general have different configurations.

In addition, universal data exchange features can be used to arrange exchange with software that is not based on the 1C:Enterprise. The following factors contribute to this:

- the data exchange format is based on the XML language, which is currently the generally accepted tool of presenting data;
- due to their modular arrangement and high flexibility, data exchange tools can be used to arrange various data exchange schemes;
- protocols provided by data exchange features are simple and can be replicated in external software systems.

15.1.3. Distributed infobases

The distributed infobase is a hierarchical structure consisting of separate infobases of the 1C:Enterprise - the nodes of the distributed infobase, between

which data exchange is arranged in order to synchronize the configuration and data.
Distributed infobase management features are based on universal data exchange features, but contain some additional functions that are not available through universal features.
The main difference between distributed infobases and universal data exchange features is that universal data exchange features allow building fairly arbitrary data exchange schemes, while distributed infobases have a narrower specialization and also transfer configuration changes to subordinate nodes.

15.2. Universal data exchange features

15.2.1. General information

The universal data exchange features may include:

- XML documents reading and writing tools,
- XML serialization,
- exchange plans.

15.2.2. XML documents reading and writing tools

The participants of the data exchange are supposed to exchange messages in XML format. Thus, the XML documents reading and writing tools are the basic level of data exchange.
XML documents reading and writing tools ensure that XML documents are handled in the most general way. This set of tools does not define the methods for presenting data of the 1C:Enterprise in XML format.
The XML documents reading and writing tools provided by 1C:Enterprise include: `ReadXML`, `WriteXML` and `XSLTransform`. The platform also supports handling XML data in the FastInfoset format, for which purpose the `FastInfosetWriter` and `FastInfosetReader` objects exist.

15.2.3. XML serialization

The main task of XML serialization - is to support 1C:Enterprise data objects reading/writing of to/from XML.

Basic tools for reading and writing XML documents do not provide a sufficient basis for solving this problem. They do not define the presentation of the 1C:Enterprise data in XML and do not provide tools for reading/writing data objects to/from XML in the accepted format as a whole.

15.2.3.1. Data representation in XML serialization

Ultimately, each data object of the 1C:Enterprise is represented as an XML element containing the value of the data object.
In terms of representation in XML, the value types are divided into simple and complex.
Simple data types include the types which values are represented by the XML serialization subsystem as XML elements with text content only.
Values of complex types are represented as XML elements containing nested elements.
An XML data type is assigned to each of 1C:Enterprise data types, the values of which can be represented in XML.
Each XML data type is characterized by the type name and the namespace which the type belongs to.
The XML data type can be as follows:

- one of the types defined in the XML Schema Part 2 document: Datatypes of W3C consortium (name space - http://www.w3.org/2001/XMLSchema);
- predefined type of 1C:Enterprise (namespace - http://v8.1c.ru/data);
- type derived from the configuration metadata of the 1C:Enterprise (does not belong to any namespace).

The XML data type can be specified explicitly in the representation of a data object in XML. To specify an XML data type, the XML element containing the value representation must contain a `type` attribute that belongs to the http://www.w3.org/2001/XMLSchema-instance namespace, which value contains the XML data type.
Another way to specify the XML data type is the name of the XML root element containing the value representation. The name of the root element representing the data object is not strictly specified and can be arbitrary. However, if the name of the root element is not specified when writing a value in XML, it will be set according to the type of the value being written. When reading data from XML, the value type, if it is not specified in the `type` attribute, can be set based on the root element name.
When considering examples of representing different values in XML and in further discussion, we will proceed from the assumption that the following namespace matches are defined:

```
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema instance"
xmlns:v8="http://v8.1c.ru/data"
```

15.2.3.1.1. Representation of simple type values in XML

The simple types it terms of representation in XML are the following 1C:Enterprise types:

- `Number`;
- `String`;
- `Date`;
- `Boolean`;
- `BinaryData`;
- `NULL`;
- `UUID`;
- `ValueStorage`;
- all references to database objects;
- references to enumerations defined in metadata.

Number

Description:

`Number` type matches the XML `decimal` data type belonging to the http://www.w3.org/2001/XMLSchema namespace.
The rules for representing values of this type are defined in the XML Schema Part 2 document: Datatypes.

Example:

```
<!-- Root element name is not explicitly specified -->
<decimal>45684.087</decimal>
<!-- XML root element name is not explicitly specified -->
<Amount>523</Amount>
<!-- XML data type is explicitly specified -->
<Data xsi:type="xsd:decimal">64793.01</Data>
```

String

Description:

The `String` type corresponds to the `string` data type from the http://www.w3.org/2001/XMLSchema namespace. `String` is written in XML as is.

Example:

```
<!-- Root element name is not explicitly specified -->
<string>This is string</string>
<!-- XML root element name is not explicitly specified -->
<Name>Ivanov</Name>
<!-- XML data type is explicitly specified -->
<Data xsi:type="xsd:string">This is string</Data>
```

Date

Description:

`Date` type values are represented as `YYYY-MM-DDTHH:MM:SS`, where:

- `YYYY` - year, represented by four digits;
- `MM` - month, represented by two digits;
- `DD` - day of a month represented by two digits;
- `T` - Latin letter `T`;
- `HH` - hour of the day;
- `MM` - minute;
- `SS` - second.

This date format is defined as valid in XML Schema Part 2 document: Datatypes.

Example:

```
<!-- Root element name is not explicitly specified -->
<dateTime>2008 11 21T12:00:00</dateTime>
<!-- XML root element name is not explicitly specified -->
<Started>2001 10 30T19:00:00</Started>
<!-- XML data type is explicitly specified -->
<Data xsi:type="xsd:dateTime">1980 08 25T10:00:00</Data>
```

Boolean

Description:

The `Boolean` type corresponds to the `boolean` data type from the http://www.w3.org/2001/XMLSchema namespace.

`False` value is represented by the `false` string, and `True` value - is represented by the `true` string. This format is provided in XML Schema Part 2 document: Datatypes.

Example:

```
<!-- Root element name is not explicitly specified -->
<boolean>false</boolean>
<!-- XML root element name is not explicitly specified -->
<Posted>true</Posted>
<!-- XML data type is explicitly specified -->
<Data xsi:type="xsd:boolean">true</Data>
```

BinaryData

Description:

`BinaryData` type matches the XML `base64Binary` data type belonging to the namespace http://www.w3.org/2001/XMLSchema.

Values of this type are specified as binary data encoded by using the `Base64` algorithm described in RFC 2045 (http://tools.ietf.org/html/rfc2045.html).

Example:

```
<!-- Root element name is not explicitly specified -->
<base64Binary>YWJjZGVm</base64Binary>
<!-- XML root element name is not explicitly specified -->
<BinaryData>YWJjZGVm</BinaryData>
<!-- XML data type is explicitly specified -->
<Data xsi:type="xsd:base64Binary">YWJjZGVm</Data>
```

NULL

Description:

Типу `NULL` type matches the XML `Null` data type belonging to the namespace http://v8.1c.ru/data. This type has a single value that is manifested by an empty string.

Example:

```
<!-- Root element name is not explicitly specified -->
<v8:Null/>
<!-- XML root element name is not explicitly specified -->
<Selected/>
<!-- XML data type is explicitly specified -->
<Data xsi:type="v8:Null"/>
```

UUID

Description:

`UUID` type type matches the XML `UUID` data type belonging to the namespace http://v8.1c.ru/data.

Values of this data type are represented in XML in compliance with common practice and standards (ISO-11578, DCE 1.1: Remote Procedure Call - Universal Unique Identifier).

Example:

```
<!-- Root element name is not explicitly specified -->
<v8:UUID>3294be0f c039 41a9 bd65 596da0dcfe68</v8:UUID>
<!-- XML root element name is not explicitly specified -->
<Id>da035e32 3f7a 4d87 accf7db8cb4b</Id>
<!-- XML data type is explicitly specified -->
<Data xsi:type="v8:UUID">08839b0b 5ec3 4a53 a9f5 173312316919</Data>
```

ValueStorage

Description:

`ValueStorage` type type matches the XML`ValueStorage` data type belonging to the namespace http://v8.1c.ru/data.

Values of this type are specified in XML as `ValueStorage` data saved in a file, and then encoded by using the `Base64` algorithm.

Example:

```
<!-- Root element name is not explicitly specified -->
<v8:ValueStorage>AQEOAAAAAAAAAO+7v3siUyIsIjHQoSJ9</v8:ValueStorage>
<!-- XML root element name is not explicitly specified -->
<Data>AQEOAAAAAAAAAO+7v3siUyIsIjHQoSJ9</Data>
<!-- XML data type is explicitly specified -->
<Data
xsi:type="v8:ValueStorage">AQEOAAAAAAAAAO+7v3siUyIsIjHQoSJ9</Data>
```

Links to Database Objects

Description:

Each type of link to database objects matches its own XML data type. The name of the XML data type for links to database objects matches the English name of the link value type of the 1C:Enterprise system.

So, for example, for the `Currencies` catalog, the English name of the link type will look like the `CatalogRef.Currencies`. The corresponding XML data type will be spelled in the same fashion.

XML data types for links to database objects do not belong to any namespace.

Link values are specified in XML as values of the `UUID` type acquired from links.

Example:

```
<!-- Root element name is not explicitly specified -->
<CatalogRef.Banks>911b5b8b 11f5 4993 9673
2c9a7a8995d5</CatalogRef.Banks >
<!-- XML root element name is not explicitly specified -->
<Ref>911b5b8b 11f5 4993 9673 2c9a7a8995d5</Ref>
<!-- XML data type is explicitly specified -->
<Data xsi:type="CatalogRef.Banks">
        911b5b8b 11f5 4993 9673 2c9a7a8995d5</Data>
```

Links to Enumerations Defined in Metadata

Description:

Each type of link to enumeration values specified in application matches its own XML data type. The XML data type name for the links to enumeration values corresponds to the English language type name of the 1C:Enterprise system.

Thus, for example, for the `AddressTypes` enumeration, an English name of this value link type will be `EnumRef.AddressTypes`. The corresponding XML data type will be spelled in the same fashion.

The XML data type name for the links to enumeration values doesn't belong to any namespace. In XML, links to enumeration values are represented as the name of the corresponding enumeration value.

Example:

```
<!-- Root element name is not explicitly specified -->
<EnumRef.AddressTypes>Registered</EnumRef.AddressTypes>
<!-- XML root element name is not explicitly specified -->
<Ref>Registered</Ref>
<!-- XML data type is explicitly specified -->
<Data xsi:type="EnumRef.AddressTypes">Business</Data>
```

15.2.3.1.2. Representation of Compound Data Types in XML

Compound data types, the values of which can be declared in XML, include the following types of the 1C:Enterprise system:

- `Type`;
- `TypeDescription`;
- `ConstantValueManager.<Constant name>`;
- all database objects;
- Record sets of registers, sequences, recalculations;
- `ObjectDeletion`.

Type

Description:

The`Type` type matches the XML Type data type belonging to the namespace http://v8.1c.ru/data. The XML element representing the value of this type contains the text with the name of the XML type that matches the “1C:Enterprise” system data type.

On the face of it, the `Type` type should be referred not to the compound data types but to simple ones, since the element representing the value of this type does not contain nested elements. However, it is not the case. Indeed, it doesn't contain nested elements. Yet, the text of the element containing the name of the XML data type contains a namespace prefix of the type that must be defined in an element or one of its parent elements, which makes the text of such element not quite self-contained. That is why the given type does not qualify as a simple one.

Example:

```
<!-- Root element name is not explicitly specified -->
<v8:Type>v8:ValueStorage</v8:Type>
<!-- XML root element name is not explicitly specified -->
<Tp>xsd:string</Tp>
<!-- XML data type is explicitly specified -->
<Data xsi:type="v8:Type">v8:ValueStorage<Data>
```

TypeDescription

Description:

`TypeDescription` type matches the XML `TypeDescription` data type belonging to the namespace http://v8.1c.ru/data. The root element that represents the value of the `TypeDescription` type includes a series of nested elements, each of them containing a certain part of the type description.

The nested `Types` element from the namespace http://v8.1c.ru/data contains representations of individual types included in the type description. The `NumberQualifiers` element belonging to the namespace http://v8.1c.ru/data contains numeric value qualifiers. `StringQualifiers` and `DateQualifiers` elements belonging to the same namespace contain string and date qualifiers, respectively.

Example:

```
<v8:TypeDescription>
  <v8:Types>
    <v8:Type>v8:UUID</v8:Type>
    <v8:Type>CatalogRef.Banks</v8:Type>
    <v8:Type>xsd:boolean</v8:Type>
    <v8:Type>xsd:decimal</v8:Type>
  </v8:Types>
  <v8:NumberQualifiers>
    <v8:Digits>10</v8:Digits>
    <v8:FractionDigits>2</v8:FractionDigits>
    <v8:AllowedSign>Any</v8:AllowedSign>
  </v8:NumberQualifiers>
  <v8:StringQualifiers>
    <v8:Length>30</v8:Length>
    <v8:AllowedLength>Variable</v8:AllowedLength>
  </v8:StringQualifiers>
  <v8:DateQualifiers>
    <v8:DateFractions>Date</v8:DateFractions>
  </v8:DateQualifiers>
</v8:TypeDescription>
```

ConstantValueManager.<Constant name>;

Description:

Each of the `ConstantValueManager.<Constant name>` type matches the XML the XML `ConstantValueManager.<Constant name>` data type that does not belong to any namespace.

Example:

```
<ConstantValueManager.OrganizationName>
<Value>OOO "Moebius"</Value>
</ConstantValueManager.OrganizationName>
```

Database objects

Description:

Database objects are represented in XML as a set of values of attributes and tabular sections. The name of the XML data type that matches the corresponding database object is defined as the English name of the value type of the 1C:Enterprise system. XML data types for database objects do not belong to any namespace. The composition of XML elements nested in the root element is determined by a particular object type, as well as by the composition of its attributes and table sections.

Each of the attributes is represented by an XML element whose name matches the attribute name. If an attribute value type cannot be uniquely identified from

the metadata, then the XML element representing the attribute contains the `xsi:type` attribute that indicates the type of the XML value.

Each of the tabular sections is represented by an XML element whose name matches the name of the tabular section.

Each row in a tabular section is represented by an XML element named `Row`. Attributes of the tabular section are represented by XML elements nested in the `Row` element.

Example:

Representation of Object of `DocumentObject.CustomerOrder` type in XML

```
<DocumentObject.CustomerOrder>
<Ref>8d106783 9726 11d7 9334 0050ba8480bd</Ref>
<DeletionMark>false</DeletionMark>
<Date>2008 04 15T12:00:00</Date>
<Number>00000006</Number>
<Posted>true</Posted>
<CompanyDivision>
317f130d 5a08 11d7 9324 0050ba8480bd</Company Division>
<StructuralUnit xsi:type="CatalogRef.CompanyCashRegisters">
317f12f4 5a08 11d7 9324 0050ba8480bd</StructuralUnit >
<CounterParty>12952ac7 5a08 11d7 9324 0050ba8480bd</CounterParty>
<CounterPartyLegalPrivateEntities xsi:type="CatalogRef.LegalEntities">
0aadfe81 5a08 11d7 9324
0050ba8480bd</CounterPartyLegalPrivateEntities>
<DocumentCurrency>
029156b4 5a08 11d7 9324 0050ba8480bd</DocumentCurrency>
<DocumentCurrencyRate>1</DocumentCurrencyRate>
<ApplyVAT>true</ApplyVAT>
<ApplySalesTax>false</ApplySalesTax>
<SumIncludesVAT>false</SumIncludesVAT>
<SumIncludesSalesTax>false</SumIncludesSalesTax>
<Comment/>
<DocumentSum>44077.14</DocumentSum>
<TransactionType>Invoice</TransactionType>
<OffsetsContract>
b0401f23 6e84 11d7 932c 0050ba8480bd</OffsetsContract>
<CompanyWarehouse>
317f1317 5a08 11d7 9324 0050ba8480bd</CompanyWarehouse>
<PriceType>317f1311 5a08 11d7 9324 0050ba8480bd</PriceType>
<PaymentDate>2008 04 15T00:00:00</PaymentDate>
<AutoProvisioning>false</AutoProvisioning>
<AutoPlacement>false</AutoPlacement>
<OffsetsCurrencyRate>33.4209</OffsetsCurrencyRate>
<DiscountSurchargeType>
00000000 0000 0000 0000 000000000000</DiscountSurchargeType>
<Organization>317f1308 5a08 11d7 9324 0050ba8480bd</Organization>
<ShipmentDate>2008 04 15T00:00:00</ShipmentDate>
<PersonInCharge>
4ff40e0b 5ac5 11d7 9325 0050ba8480bd</PersonInCharge>
<DocumentFactor>1</DocumentFactor>
<OffsetsFactor>1</OffsetsFactor>
<Items>
<Row>
<Products>
297c6534 5a08 11d7 9324 0050ba8480bd</Products>
<MeasurementUnit>
297c6535 5a08 11d7 9324 0050ba8480bd
</MeasurementUnit>
<Price>4537.56</Price>
<Sum>13612.68</Sum>
<VATRate>VAT20</VATRate>
<VATSum>2722.54</VATSum>
<SalesTaxRate>1ac73736 5a08 11d7 9324 0050ba8480bd</SalesTaxRate>
```

```
<SalesTaxSum>0</SalesTaxSum>
<ProductSpecification>
00000000 0000 0000 0000 000000000000
</ProductSpecification>
<Placement xsi:nil="true"/>
<Factor>1</Factor>
<Quantity>3</Quantity>
<DiscountSurchargePercent>0</DiscountSurchargePercent>
</Row>
<Row>
<Products>
317f12d8 5a08 11d7 9324 0050ba8480bd</Products>
<MeasurementUnit>
317f12d9 5a08 11d7 9324 0050ba8480bd
</MeasurementUnit>
<Price>4915.55</Price>
<Sum>19662.2</Sum>
<VATRate>VAT20</VATRate>
<VATSum>3932.44</VATSum>
<SalesTaxRate>1ac73736 5a08 11d7 9324 0050ba8480bd</SalesTaxRate>
<SalesTaxSum>0</SalesTaxSum>
<ProductSpecification>
00000000 0000 0000 0000 000000000000
</ProductSpecification>
<Placement xsi:nil="true"/>
<Factor>1</Factor>
<Number>4</Number>
<DiscountSurchargePercent>0</DiscountSurchargePercent>
</Row>
</Items>
<Returnables/>
</DocumentObject.CustomerOrder>
```

NOTE. In the example above, as well as in other examples of this chapter, there are long lines (for example, `<MeasurementUnit> 317f12d9-5a08-11d7-9324-0050ba8480bd </MeasurementUnit>`). Due to the limitation imposed by the book formatting, such lines are hyphenated and split with line breaks. In code, such lines are put in the module in one line.

Record Set

Description:

The XML representation of a record set includes the filter the record set was filtered by, and the records themselves that the resulting selection is composed of. The filtering values are represented in a nested XML element named `Filter` that does not belong to any namespace. Whereas all the records that make up a record set are located in a nested element named `Records` that does not belong to any namespace, either. Records are represented by XML elements named

`Record`, nested in a `Records` element. Likewise, the element name `Record` does not belong to any namespace.

Example:

Representation of the `Company Product Balance` accumulation register record set in XML:

```
<AccumulationRegisterRecordSet.CompanyProductBalance>
<Filter>
<Recorder xsi:type="DocumentRef.ProductSales">
1725f36e 6f35 11d7 932d 0050ba8480bd</Recorder>
</Filter>
<Records>
<Record>
<Recorder xsi:type="DocumentRef.ProductSales">
1725f36e 6f35 11d7 932d 0050ba8480bd</Recorder>
<Period>2008 04 13T17:45:39</Period>
<MovementType>Expense</MovementType>
<Active>true</Active>
<Products>
297c6556 5a08 11d7 9324 0050ba8480bd</Products>
<CompanyWarehouse>
317f1317 5a08 11d7 9324 0050ba8480bd</CompanyWarehouse>
<Order xsi:nil="true"/>
<RetailPrice>0</RetailPrice>
<ProductSpecification>
00000000 0000 0000 0000 000000000000
</ProductSpecification>
<Number>2</Number>
<CompanyDivision>
317f130d 5a08 11d7 9324 0050ba8480bd
</CompanyDivision>
</Record>
<Record>
<Recorder xsi:type="DocumentRef.ProductSales">
1725f36e 6f35 11d7 932d 0050ba8480bd</Recorder>
<Period>2008 04 13T17:45:39</Period>
<MovementType>Expense</MovementType>
<Active>true</Active>
<Products>
297c6558 5a08 11d7 9324 0050ba8480bd</Products>
<CompanyWarehouse>
317f1317 5a08 11d7 9324 0050ba8480bd</CompanyWarehouse>
<Order xsi:nil="true"/>
<RetailPrice>0</RetailPrice>
<ProductSpecification>
ac47d77e 5ec7 11d7 9329 0050ba8480bd
</ProductSpecification>
<Number>2</Number>
<CompanyDivision>
317f130d 5a08 11d7 9324 0050ba8480bd
</CompanyDivision>
</Record>
</Records>
</AccumulationRegisterRecordSet.CompanyProductBalance>
```


ObjectDeletion

Description:

The ObjectDeletion type matches the XML ObjectDeletion data type belonging to the http://v8.1c.ru/data namespace. The root element of the XML representation of the ObjectDeletion type value contains one nested element named Ref belonging to the http://v8.1c.ru/data namespace that contains the representation of a link to the database object.

Example:

Representation of Object of ObjectDeletion Type in XML:

```
<v8:ObjectDeletion xmlns="http://v8.1c.ru/data">
<v8:Ref xsi:type="CatalogRef.Banks">
60c5cec3 7f6f 4ec3 9620 e757fe3614ca</v8:Ref>
</v8:ObjectDeletion>
```

15.2.3.2. Access to XML Serialization Tools from 1C:Enterprise Language

To handle XML representations of simple type values, two methods of the global context are utilized, -XMLString() and XMLValue().

The XMLString() method has a single parameter,- that is the value which you want to get an XML representation for. This value should belong to a simple type, in terms of XML serialization. Otherwise, an exception is thrown. Upon normal completion, the function returns a string that can be used as a text of the XML element representing a simple type value.

The XMLValue() method performs the opposite task. This methods has two parameters:

- the type of value that should be retrieved from a string;
- and a string itself.

To convert the 1C:Enterprise system data type to the XML data type and vice versa, the XMLType() and FromXMLType() methods are designated. The XMLType() method has one parameter, -that is a type which you need to get the corresponding XML data type for. If the corresponding XML data type is identified, the method returns a value of XMLDataType type. Otherwise, if there is no corresponding XML data type, the method returns Unidentified value

The FromXMLType() method has two calling options. For the first option, the method has a single parameter of XMLDataType type. In the second option, there are two parameters: XML type name and namespace. In both cases, the method returns a 1C:Enterprise data type that matches some XML data type, if any, or Unidentified otherwise.

The global context methods WriteXML() and ReadXML() are used to write and read various values to/from XML.

The `WriteXML()` method has two required parameters. The first parameter- is an object of `WriteXML` type by using which the XML data is getting saved; and the second one - is a value to be saved in XML. Optional parameters form three different method calling options.
In the simplest case there are three parameters in the calling method, and the third one specifies the enumeration value `XMLTypeAssignment` that determines whether the XML data type should explicitly specified in the xsi:type attribute of the root XML element.
The next calling option uses a string value as the third parameter that specifies the name of the root XML element. This implies that the namespace is undefined. The fourth parameter- is a value of the `XMLTypeAssignment` type that specifies whether or not an XML data type should be explicitly indicated.
And finally, the last calling option has one other parameter, -a string value designating a namespace which the root element belongs to, following the parameter specifying the root XML element name. Likewise, the last parameter belongs to the `XMLTypeAssignment` type.

```
...
Val = "String so and so";
WriteXML(obj, Val);
WriteXML(obj, Val, "Root", XMLTypeAssignment.Explicit);
WriteXML(obj, Val, "Root", "urn:some namespace");
...
```

An execution of the snippet above will output the following piece of XML code.

```
...
<string>String so and so</string>
<Root xsi:type="xsd:string">String so and so</Root>
<d1p1:Root xmlns:d1p1="urn:some namespace">String so and
so</d1p1:Root>
...
```

If a value of a type that cannot be represented in XML is passed as the value to be placed in XML, an exception will be thrown.
`ReadXML()` method is provided for reading values from XML. This method has one required parameter,- the `ReadXML` object which a value should be read from. For the second parameter, a type of value that should be read from XML can be specified. If the value type is explicitly specified in XML, then the value of the second parameter can be specified as `Undefined`, or it can be omitted altogether. In this case, the `ReadXML()` method tries to identify the type of the value being read by the contents of the xsi:type attribute, or by the element name if the xsi:type attribute is missing. If it fails to identify the type or value of the specified type cannot be read from XML, then an exception is thrown. On success, the `ReadXML()` method returns the retrieved value.

Attention should be paid to how constant value managers, database objects, and record sets are read. Upon success, the `ReadXML()` method returns the value read from XML, however, this value has not yet been written to the database. If, for example, a catalog item is read, then in order for the catalog item to be get written to the database, it is necessary to address its `Write()` method, like it is done during the regular saving of the changed object state. This also applies to other database objects, constant value managers, and record sets.
When reading a database object from XML, the database searches for an object with the same reference value. If such an object is found, then reading from XML looks as though the object was read from the database, and then the values of its properties, tabular sections, etc., are overwritten with the values acquired from XML. Contrariwise, if an object is not found by reference, then reading from XML is similar to generating a new object, setting its reference value and filling its content with values read from XML.
`CanReadXML()` method determines whether it is possible to read a value from the `ReadXML` object located at the current position of an XML document. A `ReadXML` method is being passed to the given method as its parameter. If the method returns `True`, then the reading is possible; if `False`,-then the value cannot be read.
`GetXMLType()` method allows to get an XML data type corresponding to the current position of the XML document, from a `ReadXML` object. This method also has one single parameter, -`ReadXML`.

15.2.4. Exchange plans

Exchange plans form a core which the rest of the data exchange tools are grouped around. In a single configuration, an arbitrary number of exchange plans can be defined. Each exchange plan specifies a set of data to be exchanged through this exchange plan. Along with the data set, specific presentation formats of this data can be defined.
It is assumed that the data formats are based on XML, but due to the flexibility of XML language and the availability of quite advanced XML tools in the 1C:Enterprise system, there is still a large enough room for creativity in regard to the data presentation methods.
Two most significant components can be distinguished in the exchange plans:

- message infrastructure,
- and change registration service.

The data items of an exchange plan are the nodes of this exchange plan, just as the data items of a catalog are the items of that catalog. Each of the exchange plan nodes designates a participant in the exchange of data for this exchange

plan. One of the nodes corresponds to this infobase, and the rest of them -to other participants whom this infobase can exchange data with.
Data is transferred between the nodes by means of messages. Message handling tools form the **message infrastructure**. Each message belongs to a specific exchange plan, it has a specific **sending node, a Sender,** and a specific **receiving node, a Recipient**. A message cannot be sent to an unspecified node and cannot be received from an unspecified node. A unique integer number is assigned to each message.
Change registration service is designed to register modifications in data made by the 1C:Enterprise system, in order to transfer only modified data rather than all the data when the data exchange occurs.
Thus, exchange plans define a set of tools designed to organize the data exchange. Let's review these tools in more detail.

15.2.4.1. Exchange Plan Nodes

When you create a new exchange plan, it automatically creates one node, -this node, or the exchange plan node, corresponds to this infobase. The remaining nodes, that is, nodes which the given node can exchange data with, are not being automatically created as part of the exchange plan.
To each node, a unique code must be assigned, as a node is identified by code in the course of data exchange. Codes of nodes are assigned is such a way that the exchange parties could "recognize" each other.

NOTE. Exchange plan must have a code and a name. In other words, the standard `Code` and `Name` attributes of an exchange plan cannot be set to a length of 0.

Suppose it is required to organize a data exchange between two infobases through an exchange plan named `RemoteWareHouses`. One of these infobases serves as the headquarters, and the other -as a remote warehouse.
In this case, it would be reasonable to set the code value of this `RemoteWareHouses` exchange plan in the first infobase to `Office` (certainly, if the code length allows), and in the second one- to `WareHouse1`. In this fashion, these nodes will get named. But this is not enough, as you still need to specify the nodes which the data should be exchanged with. To do this, in first infobase, a `RemoteWareHouses` exchange plan node with `WareHouse1` code should be created, and likewise, in the second infobase- a node with `Office` code.
Thus, the first infobase will "get to know' that, within the `RemoteWareHouses` exchange plan, it is called `Office`, and it is to exchange data with a node called `WareHouse1`; and the second infobase -will "get to know" that its name is `WareHouse1`, and that it is to exchange data with the `Office` node.

15.2.4.2. Message infrastructure

The most important part of the message infrastructure is the messages themselves. Like it is stated above, the messages are transmitted through an exchange plan from one node to another. That is, each message is strictly associated with a certain exchange plan, it has one Sender and one Recipient.

Let's see what a message is about. The messages are XML documents with predefined structure. Lets take the following message as an example:

```
<v8msg:Message xmlns:v8msg="http://v8.1c.ru/messages">
  <v8msg:Header>
    <v8msg:ExchangePlan>RemoteWareHouses</v8msg:ExchangePlan>
    <v8msg:To>WareHouse1</v8msg:To>
    <v8msg:From>Office</v8msg:From>
    <v8msg:MessageNo>20</v8msg:MessageNo>
    <v8msg:ReceivedNo>15</v8msg:ReceivedNo>
  </v8msg:Header>
  <v8msg:Body>
    <!--Message body -->
  </v8msg:Body>
</v8msg:Message>
```

The entire message is contained inside an XML element named `Message` belonging to the http://v8.1c.ru/messages namespace. A message consists of the header and the message body. Accordingly, the `Message` element contains two nested elements named `Header` and `Body`. Both belong to the http://v8.1c.ru/messages namespace.

The`Header` element contains a message title. The header structure is strictly fixed. The title's data is presented in several XML elements nested in a `Header` element. All the elements nested in the `Header` element belong to the http://v8.1c.ru/messages namespace:

- `ExchangePlan` contains the exchange plan name this message belongs to.
- `To` contains a code of the node this message is to be sent to.
- `From` contains the sending node's code.
- `MessageNo` contains the identifying number of this message. The message number is a positive integer and is assigned by a sending node. The number of each subsequent message is equal to the number of the previous sent message incremented by 1.
- `ReceivedNo` contains the maximum message number that the message's sending node has received from the receiving node. This value is included in the message header to confirm receipt of the messages.

The message body is contained in an XML element named `Body` that belongs to the http://v8.1c.ru/messages namespace. This element can have arbitrary

content determined by the application needs. The content of the message body is by no means regulated by the message infrastructure.

15.2.4.3. Change Registration Service

The essence of registering changes is to have a list of modified data items. These data items should be transmitted in the next message to a certain node that the data is exchanged with. With each modification of data, it must be registered that there are changes, and that they have to be transferred to all nodes which this data exchange is supported with. Once the receipt of a message in which the changes have been sent is confirmed, the change registration records should be deleted.

Registration of changes can be performed for the following data items:

- `ConstantValueManager.<Constant name>`;
- Database objects:
  - `CatalogObject.<Catalog Name>`;
  - `DocumentObject.<Document Name>`;
  - `ChartOfAccountsObject.<Chart Of Accounts Name>`;
  - `ChartOfCharacteristicTypesObject.<Chart Of Characteristic Types Name>`;
  - `ChartOfCalculationTypesObject.<Chart Of Calculation Types Name>`;
  - `BusinessProcessObject.<Business Process Name>`;
  - `TaskObject.<Task Name>`.
- Record Sets:
  - `InformationRegisterRecordSet.<Information Register Name>`.
  - `AccountingRegisterRecordSet.<Accounting Register Name>`.
  - `AccumulationRegisterRecordSet.<Accumulation Register Name>`.
  - `SequenceRecordSet.<Sequence Register Name>`.
  - `CalculationRegisterRecordSet.<Calculation Register Name>`.
  - `RecalculationRecordSet.<Recalculation Name>`.

For each of the data items listed above, its own modification registration table is being maintained. The tables have a varying structure, depending on which data items are recorded for modifications, yet still the structure of the tables is quite similar. Thus, three components can be recognized in the structure:

- a data item key the modifications are being registered for;
- a reference to the node a modification should be passed to;

- message number a modification was passed in for the first time.

The structure of the change registration tables for different kinds of data differs by key, since the keys for different kinds of data are different, too:

- for a constant, the key is a constant identifier;
- for database objects, an object reference is used as a key;
- for record sets with a specified recorder, a reference to a the recorder object is used as a key;
- for an information register record set, if the recorder is not specified, a set of dimensions included in the main filter is used as a key; and if the information register is periodic and the main filter by period is included, then the period is also included in the key.

A data item modification should be registered for all nodes which the modification is to be passed to. Thus, as a result of a data item modification, N-number of records should appear in the change registration table, where N- is the number of nodes which the modifications are registered for. Each of these records contains the same data item key value and differing values of the reference to a node. Immediately after registering a modification, the message number is `NULL`. When a change is being sent for the first time, the number of the message that carries the change is placed in this field.

When configuring an exchange plan, the so-called exchange plan composition is specified. Exchange plan composition includes meta-data objects. If a meta-data object is included into the exchange plan composition, it shows that the data modifications that correspond to this meta-data object can be registered for the nodes of this exchange plan. If the meta-data object is not included in any exchange plan, then a change registration table is not created for this object and no data change registration is performed.

When determining whether a meta-data object is included in the exchange plan, the `AutoRecord`property is specified. Auto-registration can be enabled or disabled. If auto-registration is enabled, then, once the data changes, the registration will be performed automatically. Otherwise, if it is disabled, then the change can be registered manually.

15.2.4.4. Accessing the Exchange Plan Tools by Means of 1C:Enterprise Language

A set of objects has been designed, to handle exchange plans by means of 1C:Enterprise language.

`ExchangePlansManager`object, apart from the features typical for this kind of managers, contains the following methods to handle the change registration service, `RecordChanges()`, `DeleteChangeRecords()`,

`IsChangeRecorded()`, `SelectChanges()`, as well as methods for generating objects that read and write messages: `CreateMessageReader()`, `CreateMessageWriter()`.

For the `ExchangePlanManager.<Exchange Plan Name>` object, the most valuable feature is `ThisNode()` method that returns a link to the node of this exchange plan corresponding to this infobase (hereinafter referred to as “this node”). To identify an exchange plan node that corresponds to this infobase, the standard attribute `ThisNode` is also of use. A value of `ThisNode` attribute can be changed. The most common case of using this operation -is the restoration of the infobase correct state, following the recovery of data from an XML file. In this case, in the restored database, it is necessary to assign “this node” to the exchange plan item that was set in the source database. This is especially relevant in the case when the references to "this node" of the exchange plan were stored in any objects of the infobase. Let us consider the process of replacing "this node" in more detail.

A regular re-writing of an exchange plan item with the set (or reset) `ThisNode` property cannot be performed (if there is an exchange plan item in the system with `ThisNode` property being set),- as the system will throw an exception. In order to perform this operation successfully, the following steps should be done:

1. For an item that is planned to be specified as “this node”, `ThisNode` property in the `DataExchange.Download` mode should be set to True.
2. For an item that was “this node” previously, `ThisNode` property should be reset (in a regular way).
3. It is recommended to perform these operations (setting a new node and disabling the old one) in a transaction.

Here is an example of reassigning "this node" item:

```
BeginTransaction();

CurrentThisNode = ExchangePlans.DataExchange.ThisNode();

NewObject = NewThisObject.GetObject();
NewObject.DataExchange.Import = True;
NewObject.ThisNode = True;
NewObject.Write();

CurrentObject = CurrentThisNode.GetObject();
CurrentObject.ThisNode = False;
CurrentObject.Write();

CommitTransaction();
```

When changing ThisNode, a few peculiarities of the system behavior should be borne in mind:

1. If you shut down the system at a junction when an exchange plan does not have the "this node" item, then the system will automatically create a new exchange plan item at startup, with ThisNode property set to True.
2. If the system has two items with ThisNode property value set to True, then the system behavior grows rather unpredictable.

ExchangePlanObject.<Exchange Plan Name> corresponds to the exchange plan node. SentNo and ReceivedNo properties deserve special attention. The SentNo property contains the number of the last message sent from this infobase to the address of the node that corresponds to the ExchangePlanObject.<Exchange Plan Name> object. The ReceivedNo property contains the largerst of the message numbers received by this infobase from the node that corresponds to the ExchangePlanObject.<Exchange Plan Name> object.

15.2.4.4.1. Change Registration

As mentioned above, the registration of changes can be performed automatically when recording or removing a data item. Let us see how this happens. Each of the objects whose modifications can be registered by an exchange plan has DataExchange property of DataExchangeParameters type. This read-only property is designed to manage various parameters in the course of data exchange.

DataExchangeParameters object has Recipients property of NodeSet type. This property stores a list of nodes whose modifications will be recorded when writing or removing data. The list of recipients is automatically populated prior to calling the BeforeWrite() handler when data writing is

being performed, or BeforeDelete() handler when data deletion is carried out. However, the automatic filling will be performed only if the AutoFill property of the NodeSet object is set to True (with True being the default value for the AutoFill property). With AutoRecord on, the list of recipients gets links to all nodes of all exchange plans that include the corresponding metadata object, provided that the value of the Auto-Registration property is Enabled. It goes without saying that the nodes corresponding to this infobase will not be included in the list of recipients. Prior to automatic filling, a list of recepients is cleared.

In BeforeWrite() (and/or BeforeDelete()) handler, changes to the list of recipients can be made, by adding or removing links to the nodes. Yet, it should be remembered that the list of recipients can only contain links to nodes related to those exchange plans that include the corresponding metadata object.

In the example below, BeforeWrite() handler rules out from the list of recipients a node with the code Special of the RemoteWareHouses exchange plan.

```
Procedure BeforeWrite()
  Node = ExchangePlans.RemoteWareHouses.FindByCode("Special");
  DataExchange.Recipients.Remove(Node);
EndProcedure
```

By setting the AutoFill property to False, one can ensure that the recipient list will not be automatically populated. In this case, operations with the list of recipients can be performed not only in the BeforeWrite() handler, but also in any code fragment, as shown in the example.

```
Object = Link.GetObject();
Node = ExchangePlans.RemoteWareHouses.FindByCode("WareHouse1");
Object.DataExchange.Recipients.AutoFill = False;
Object.DataExchange.Recipients.Add(Node);
Object.Write();
```

A number of methods for registering changes contains ExchangePlansManagerobject. First of all, it is RecordChanges() method. This method allows to register modifications for single data items or for entire groups for one or more nodes. The first parameter of this method- is a link to an exchange plan node or an array of links to nodes whose changes are registered. If the first parameter is a single link to a node, then the second parameter can be omitted. Meanwhile, the registration of changes for all data items that are currently present in the database and whose modifications can be registered for this node is being performed.

This can be of use when configuring the initial transfer of data to a newly created node.

```
Node = ExchangePlans.RemoteWareHouses.FindByCode("New");
ExchangePlans.RecordChanges(Node);
```

If the first parameter is an array of links to nodes, then the second parameter must be specified. However, the second parameter may also be present if the first parameter- is a single node link. Depending on the method of specifying the second parameter, there is an option to register the modifications of either a single data item or all data related to one metadata object.

To register changes to one item, the data element itself, a link to a database object, or a link to an object of the `ObjectDeletion` type can be specified as the second parameter.

If a data item is specified, then its modification is registered. If a link to a database object is specified, then a modification of this object is registered.

If the second parameter is of the `ObjectDeletion` type, then a modification to the database object, a link to which is contained by the `ObjectDeletion` object, is registered.

```
Nodes = New Array(2)
Nodes[0] = ExchangePlans.RemoteWareHouses.FindByCode("WareHouse1");
Nodes[1] = ExchangePlans.RemoteWareHouses.FindByCode("WareHouse2");
Data = Catalogs.Products.FindByCode("TP00127");
ExchangePlans.RecordChanges(Nodes, Data);
```

To register changes in all data related to the metadata object, the corresponding metadata object must be specified as the second parameter.

```
Nodes = New Array(2)
Nodes[0] = ExchangePlans.RemoteWareHouses.FindByCode("WareHouse1");
Nodes[1] = ExchangePlans.RemoteWareHouses.FindByCode("WareHouse2");
ExchangePlans.RecordChanges(Nodes, Metadata.Catalogs.Products);
```

To remove records of registration changes, `ExchangePlansManager`object has `DeleteChangeRecords()` method. By using it, one can delete change registration records for all data items that are registered for a node.

```
Node = ExchangePlans.RemoteWareHouses.FindByCode("WareHouse1");
ExchangePlans.DeleteChangeRecords(Node);
```

One can remove the change registration records for a specific data item for one or more nodes.

```
Nodes = New Array(2)
Nodes[0] = ExchangePlans.RemoteWareHouses.FindByCode("WareHouse1");
Nodes[1] = ExchangePlans.RemoteWareHouses.FindByCode("WareHouse2");
Data = Catalogs.Products.FindByCode("TP00127");
ExchangePlans.DeleteChangeRecords(Nodes, Data);
```

One can also delete records of all modifications made to a metadata object for one or several nodes.

```
Nodes = New Array(2)
Nodes[0] = ExchangePlans.RemoteWareHouses.FindByCode("WareHouse1");
Nodes[1] = ExchangePlans.RemoteWareHouses.FindByCode("WareHouse2");
ExchangePlans.DeleteChangeRecords(Nodes, Metadata.Catalogs.Products);
```

Moreover, if a single node is specified as the first parameter, a message number can be specified as the second parameter. In this case, `DeleteChangeRecords()` method deletes from all the change registration tables all the entries related to a specified node whose message number is less than or equal to the value of the second parameter (but not `NULL`). This form of the method is earmarked to remove the change registration records with acceptance of changes confirmed from the node specified in the first parameter.

To check whether a data item change is registered for a particular node, use `IsChangeRecorded()` method. The first parameter of this method -is a node link, and the second parameter -is a `Data` item, a link to a database object or an `ObjectDeletion`.

15.2.4.4.2. Recording Data Exchange Messages

To record messages, use `ExchangeMessageWriter`object.

`ExchangeMessageWriter`object is generated when addressing CreateMessageWriter() `method of ExchangePlansManager object`.

`ExchangeMessageWriter` object has three methods:

- `BeginWrite()`,
- `EndWrite()`,
- `CancelWrite()`.

`BeginWrite()` method has two parameters: an object of `WriteXML` type, through which the message will be recorded, and a link to the node which the message is addressed to. `BeginWrite()` method calculates the message number by incrementing the number of the previous message by 1; it records the opening part of the XML element containing the whole message, the entire message header, as well as the opening part of the XML element containing the message body. Following that, one can start recording the contents of the message body.

To complete the recording of a message in a regular way, use the `EndWrite()` method. When calling this method, the closing part of the XML element containing the message body and the closing part of the XML element containing the entire message are recorded. Upon successful recording of the XML elements that conclude the message, the message is considered to be sent, and its number is stored as the number of the last message sent from this node to the receiving node.

If there is a need to interrupt the recording of a message and not to consider it sent, `CancelWrite()` method should be addressed.
The following is a typical code snippet that records a data exchange message.

```
WriteXML = New WriteXML();
WriteXML.OpenFile(MessageFileName);
Node = ExchangePlans.RemoteWareHouses.FindByCode(NodeCode);
MessageRecord = ExchangePlans.CreateMessageWriter();
MessageRecord.BeginWrite(WriteXML, Node);
// Recording the message body
MessageRecord.EndWrite();
```

The message body can contain any data presented in XML, depending on what format is adopted for a particular exchange plan. Also, we shall consider a case when a message contains data whose modifications have been registered by the change registration service. To implement this, you should select registered changes, iterate them and put the XML representations of the modified data into the message.
To select changes, use the `SelectChanges()` method of the `ExchangePlansManager`object. This method has two parameters: a link to the node whose changes are selected, and a number of the message in which the changes should be placed. `SelectChanges()` method returns an object of `DataSelection` type containing the keys of the data whose modifications were registered for the node passed as the first parameter. Besides, `SelectChanges()` method puts the message number passed as the second parameter into the corresponding fields of the selected entries in change registration tables if these fields have previously contained a `NULL` value.
If a modification has already been selected at the previous access, the message number in the corresponding record of the change registration table is not modified; and if the modification has not yet been selected, then the selection will remember the number of the first message which the modifications have been selected for.
To iterate the selected modifications, the `DataSelection` object has the `Next()` and `Get()` methods. When calling `Next()` method, a transition to the next key in the selection occurs. When `Next()` method is addressed for the first time, a transition to the first key takes place. When addressing `Get()` method, a data item corresponding to the current selection key is selected from the database. Here it is necessary to make a special note with regard to the remote database objects. If the registration of a change in a database object was performed as a result of its deletion, then `Get()` method will return not a database object, but an object of the type `ObjectDeletion`.
Placing the XML representation of the data items in a message is performed by using the global context `WriteXML()` method.

Below is a snippet of code in which a message is formed, and then the registered data modifications are placed in its body.

```
WriteXML = New WriteXML();
WriteXML.OpenFile(MessageFileName);

Node = ExchangePlans.RemoteWareHouses.FindByCode(NodeCode);

MessageRecord = ExchangePlans.CreateMessageWriter();
MessageRecord.BeginWrite(WriteXML, Node);

Selection = ExchangePlans.SelectChanges(Node,
MessageRecord.MessageNo);
While Selection.Next() Loop

  Data = Selection.Get();
  WriteXML(XMLWriter, Data);

EndDo;

MessageRecord.EndWrite();
```

15.2.4.4.3. Reading data exchange messages

The `ExchangeMessageReader` object is desgined to read data exchange messages.
The object is created by calling `CreateMessageReader()` method of the `ExchangePlansManager` object.
The `ExchangeMessageReader` object has three methods:

- `BeginRead()`,
- `EndRead()`,
- `CancelRead()`.

The `BeginRead()` method has two parameters: the object of `ReadXML` type used to read messages and `AllowedMessageNo` enumeration value.
The `BeginRead()` method reads the beginning of XML item, containing all messages. It reads message header and verifies its eligibility:

- whether the exchange plan is defined, to which the message is attached;
- whether the sender and receiver are set correctly;
- whether the message number is acceptable.

The message number eligibility is defined considering the value of the second parameter. If the second parameter value is set to `AllowedMessageNo.Any`, the message read may have any number. If the second parameter value is set to `AllowedMessageNo.Next`, the message number must be greater than the maximum number of previously received messages exactly by 1. And if the second parameter is set to `AllowedMessageNo.Greater`, the message number

must be just greater than the numbers of previously received messages. The default value for the second parameter is `AllowedMessageNo.Greater`.
Then the beginning of XML element containing message body is read. If the message is not eligible or the error occurred when reading, the exception is called. If everything is fine, the message body can be read.
For normal ending of message reading, the `EndRead()` method is used. This method reads the end of XML element containing message body and the end of XML element containing all messages. If everything is fine, the message is considered received and the message number (if it's greater than numbers of previously received messages) is recorded as the maximum number of the received message.
The `CancelRead()` method allows to cancel message reading at any moment.
The following code fragment, where the usage of `ExchangeMessageReader` object is demonstrated, is a standard one.

```
ReadXML = New ReadXML();
ReadXML.OpenFile(MessageFileName);
MessageReading = ExchangePlans.CreateMessageReader();
MessageReading.BeginRead(ReadXML, AllowedMessageNo.Greater);
// Reading message body
MessageReading.EndRead();
```

As noted above, the message body may contain any information, but here we will review changed data reading from the message body. The code fragment for reading a message with the changed data looks as follows.

```
ReadXML = New ReadXML();
ReadXML.OpenFile(MessageFileName);

MessageReading = ExchangePlans.CreateMessageReader();
MessageReading.BeginRead(ReadXML);

ExchangePlans.DeleteChangeRecords(MessageReading.Sender,
MessageReading.MessageNo);
While CanReadXML(ReadXML) Loop

  Data = ReadXML( XMLReader);
  Data.DataExchange.Sender = MessageReading.Sender;
  Data.DataExchange.Load = True;
  Data.Write();

EndDo;

MessageReading.EndRead();
```

Calling `DeleteChangeRecords()` method of the `ExchangePlansManager` object is designed to delete registration records used to confirm the receival. Then the `ReadXML()` method is used to read data

from the message body while there's an opportunity to do so. Before writing data to the database, two properties of `DataExchangeParameters` object, which belongs to the object providing data read from XML. The node that sent data is specified in `Sender` property to avoid change registration to be sent to the node, from which the data were just received. Setting `True` value to the `Load` property means that the writing is made within data uploading, not simple writing. In this case, some verifications will not be performed when writing.

15.2.4.4.4. Guaranteed message delivery

In the examples of data exchange message writing and reading above, it was supposed that the sent messages could be lost for some reason. That's why when writing messages they included all registered changes, including those that were already sent without the receival confirmation.
When receiving, the message with the number that is greater than the maximum number from received messages is acceptable. The changes registration is deleted only for numbers of messages, for which the receival was confirmed.

```
MessageReading.BeginRead(ReadXML);
ExchangePlans.DeleteChangeRecords(MessageReading.Sender,
MessageReading.ReceivedNo);
```

This logic is standard when data exchange is implemented. However, you can implement an option, where the message delivery is guaranteed. In this case, changes registration must be deleted right after the message is successfully written to avoid sending changes that were already sent. Only messages with the number greater than the maximum number of previously received messages by 1 can be received.
In this case, the above code fragment that writes a message will look as follows.

```
WriteXML = New WriteXML();
WriteXML.OpenFile(MessageFileName);

Node = ExchangePlans.RemoteWareHouses.FindByCode(NodeCode);

MessageRecord = ExchangePlans.CreateMessageWriter();
MessageRecord.BeginWrite(WriteXML, Node);

Selection = ExchangePlans.SelectChanges(Node,
MessageRecord.MessageNo);
While Selection.Next() Loop

  Data = Selection.Get();
  WriteXML(XMLWriter, Data);

EndDo;

MessageNo = MessageRecord.MessageNo;
MessageRecord.EndWrite();

ExchangePlans.DeleteChangeRecords(Node, MessageNo);
```

The code fragment, in which the message is received, will look as follows:

```
ReadXML = New ReadXML();
ReadXML.OpenFile(MessageFileName);

MessageReading = ExchangePlans.CreateMessageReader();
MessageReading.BeginRead(ReadXML, AllowedMessageNo.Next);
While CanReadXML(ReadXML) Loop

  Data = ReadXML( XMLReader);
  Data.DataExchange.Sender = MessageReading.Sender;
  Data.DataExchange.Load = True;
  Data.Write();

EndDo;

MessageReading.EndRead();
```

Remember that it may be quite hard to set the guaranteed message delivery and in most cases it's recommended to send changes repeatedly until getting receival confirmation.

15.2.4.4.5. Resolving collisions

In the message writing and reading examples above it was not considered that during data exchange the same data item can be changed in two data sharing nodes at the same time. In this case it's not clear, which change must be accepted. This situation is called **collision**.

One of the ways to resolve collision - to define the master node and the subordinate node. The change made in the master node must be accepted and the change made in the subordinate mode must be denied.
To implement this process during message receival and before writing data, define, whether the data change was registered, and depending on the node role in this sender-receiver pair, decide whether the data should be written or not.
Below is the example of "master - subordinate" strategy implementation when reading messages. It's supposed that the `Master` attribute of `Boolean` type was provided in exchange plan to store node role.

```
ReadXML = New ReadXML();
ReadXML.OpenFile(MessageFileName);

MessageReading = ExchangePlans.CreateMessageReader();
MessageReading.BeginRead(ReadXML);

ExchangePlans.DeleteChangeRecords(MessageReading.Sender,
MessageReading.MessageNo);
Sender = MessageReading.Sender;
Master = Sender.Master;
While CanReadXML(ReadXML) Loop

  Data = ReadXML( XMLReader);
  If Master Or Not ExchangePlans.IsChangeRecorded(Sender, Data) Then

    Data.DataExchange.Sender = MessageReading.Sender;
    Data.DataExchange.Load = True;
    Data.Write();

  EndIf;

EndDo;

MessageReading.EndRead();
```

15.2.4.4.6. Namespace Mapping

When writing a message, after executing `BeginWrite()` method of the `ExchangeMessageWriter`, no namespace mappings are defined. At the same time a set of namespaces can be used several times when writing specific data items. In this case the mappings of the same namespaces will be included in message body several times. This is why after message writing is started and before whiting the body, it's reasonable to place several code strings that will define mapping of the most frequently used namespaces.
The example of this is provided in message writing fragment below.

```
WriteXML = New WriteXML();
WriteXML.OpenFile(MessageFileName);

Node = ExchangePlans.RemoteWareHouses.FindByCode(NodeCode);

MessageRecord = ExchangePlans.CreateMessageWriter();
MessageRecord.BeginWrite(WriteXML, Node);
WriteXML.WriteNamespaceMapping("xsd",
"http://www.w3.org/2001/XMLSchema");
WriteXML.WriteNamespaceMapping("xsi",
"http://www.w3.org/2001/XMLSchema instance");
WriteXML.WriteNamespaceMapping("v8", "http://v8.1c.ru/data");
Selection = ExchangePlans.SelectChanges(Node,
MessageRecord.MessageNo);

While Selection.Next() Loop

  Data = Selection.Get();
  WriteXML(XMLWriter, Data);

EndDo;

MessageNo = MessageRecord.MessageNo;
MessageRecord.EndWrite();

ExchangePlans.DeleteChangeRecords(Node, MessageNo);
```

In some cases such method will help to significantly reduce the size of XML document, in which the data exchange message is located.

15.3. Distributed infobases

15.3.1. General principles

Distributed infobase - a combination of 1C:Enterprise infobases (distributed infobase nodes) that support configuration and data synchronization. Distributed infobase has the hierarchical structure. Each distributed infobase node may have the **master** node and any number of **subordinate** nodes. The master node or the node without the master node is called the **root** distributed infobase node

(Master database base on fig. 276). Each node can only share data with its neighbours, i.e the master and subordinate nodes.

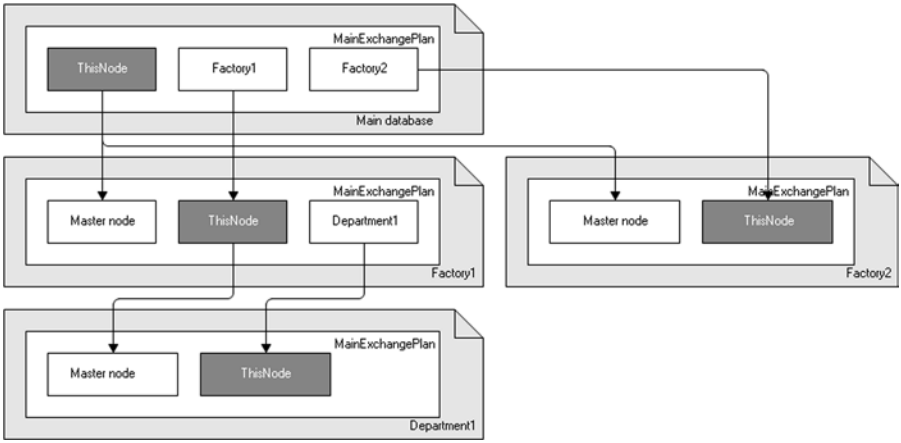


Fig. 276. Hierarchical distributed infobase

Configuration can only be modified in the root distributed infobase node with further distribution, according to the hierarchy, from the root node to the subordinate node and so on. The distributed infobase management mechanism ensures the same configuration in all distributed infobase nodes.
The data can be changed in any distributed infobase node. Data synchronization is provided by distributing data changes, made in one node, to the whole distributed infobase structure.
When organizing document sequences operation in distributed infobase, consider that including document in a sequence only makes sense in one distributed infobase node. This might be a node, in which the document was generates or any other node, but it should be only one node. Breaching this principle may result in various issues, for example inability to restore document sequences.
If the completely identical configurations are supported in the distributed infobase, the data must be completely identical. The content of data with changes sent in distributed infobase can be regulated "vertically" (by defining the cluster of metadata objects containing data that participate in the exchange) or "horizontally" (by setting conditions for sending and receiving modification on the data items level).

15.3.2. Exchange plans

Exchange plans are in the heart of distributed infobase management. However, to consider the exchange plan eligible for organization of a distributed infobase, the Distributed infobase property must be set when configuring this exchange plan.

The distributed infobase data are transferred using messages provided by the message infrastructure. Unlike universal data exchange mechanisms, the content of messages sent between distributed infobase nodes can't be arbitrary and is regulated by the exchange protocol, accepted for the distributed infobase.

The variety of data, whose changes will be used to perform exchange in distributed infobase, is defined by the exchange plan content. If a meta-data object is included into the exchange plan composition, it shows that the data modifications that correspond to this meta-data object can be registered for the nodes of this exchange plan. Unlike the universal data exchange mechanisms, the variety of data that can be exchanged within the distributed infobase is strictly limited by the content of the associated exchange plan.

To register data changes in the distributed infobase, the change registration service is used. The data items are placed in the message using the XML serialization mechanisms. Apart from data changes , configuration changes and some service information are sent between the distributed infobase nodes. Configuration changes registration and their transfer in the distributed infobase are completely automated and are unavailable to the user and configuration developer.

Unlike the universal data exchange mechanisms, generation and receival of data exchange messages in distributed infobase is performed "as one action", meaning that all message content is generated by calling one 1C:Enterprise language method. The reading of message content is also performed by calling one method. To control the content of data that is put in the message and read from the message to be put in the database, the event handlers can be set on the exchange plan module data items level:

- `OnSendDataToSlave`,
- `OnSendDataToMaster`,
- `OnReceiveDataFromSlave`,
- `OnReceiveDataFromMaster`.

Thus, almost all universal data exchange mechanisms are used in the distributed infobase, but there are additional features that are exclusive to the distributed infobase.

15.3.2.1. Master and subordinate nodes

As stated above, each distributed infobase node may have one master node and an arbitrary number of subordinate nodes (see fig. 276). For the master node, the

node is subordinate and for subordinate nodes - master node. Node, which doesn't have the master node is the distributed infobase root node.

IMPORTANT! The distributed infobase root node - the only place, where infobase configuration can be changed.

The distributed infobase can be built based on several exchange plans with the set Distributed infobase property. Interaction in each "master - subordinate" pair happens in accordance with one of the exchange plans set in configuration. There are no restrictions for using exchange plans in distributed infobase nodes. Each distributed infobase node, just like with universal data exchange mechanisms, only "knows" its neighbours, i.e. the master and subordinate nodes. The complete scheme of distributed infobase is not known to any node if the scheme has more than two levels.

15.3.2.2. Resolving collisions

The standard procedure for resolving collisions in distributed infobase is based on "master - subordinate" relationships and it's performed automatically when receiving a message. It's assumed that the data item change made in the master node has a higher priority than the change made in the subordinate node. Thus, if the message from subordinate node contains data item with changes registered for this subordinate node, no actions will be taken, meaning that the data item will not be written in the database and the change registration record will not be deleted.

If the message from master node contains data item with changes registered for the master node, the data item will be written in the database and the change registration record will be deleted.

15.3.2.3. Initial image of distributed infobase node

Usually when operating the distributed infobase, the subordinate node is generated from the master node. It means that for the subordinate node specified in the exchange plan the new infobase that corresponds to the subordinate node is created based on configuration and data in the master mode. Such infobase is called the initial distributed infobase node image.

When creating an initial image, the following actions are taken:

- The configuration is transferred from the master node to the initial node without change.
- The required extensions are transferred from the master node to the initial image. The transferring is performed in case the extension support is enabled for the exchange plan used and there are extensions that can be used in the distributed infobase.

- The object nodes are created in the initial image exchange plan. They are initialized in a way that additional setting is not required to start exchanging data between the master node and the subordinate nodes.
- The data are transferred from the master node to the initial image according to the rules set by the exchange plan (the content of exchange plan and result of `OnSendDataToSlave` event handler).

The procedure for creating the initial image can be repeated for the same node. Doing so makes sense when the infobase of the subordinate node was lost irrevocably. When creating the initial image all change registration records of the node, for which the initial image is created, are deleted because it's assumed that the history of this subordinate node begins from the fresh start.

15.3.2.4. Data exchange message in the distributed infobase

To transfer changes of data and configuration in distributed infobase, the data exchange messages are used that are provided by the messages infrastructure. In case of universal data exchange mechanisms the developer defines what goes into the message body and how, but in case of distributed infobase the structure and content of data stored in the message body are strictly defined.

Let's review the structure of data exchange message used in the distributed infobase. Lets take the following message as an example:

```
<v8msg:Message>
  <v8msg:Header>
    <v8msg:ExchangePlan>RemoteWareHouses</v8msg:ExchangePlan>
    <v8msg:To>WareHouse1</v8msg:To>
    <v8msg:From>Office</v8msg:From>
    <v8msg:MessageNo>20</v8msg:MessageNo>
    <v8msg:ReceivedNo>15</v8msg:ReceivedNo>
  </v8msg:Header>
  <v8msg:Body>
    <v8de:Changes>
      <v8de:Signature>7b4d5320 f69c 4a7b 9273
ff56607fc8ab</v8de:Signature>
      <v8de:Config>
        <!--Changed configuration objects -->
        <v8de:Digest1>88d3f3a6ba3f4df03c7ec00f154837fc</v8de:Digest1>
        <v8de:Digest2>00cf636b02a488103a64c7a2cf81069e</v8de:Digest2>
      </v8de:Config>
      <v8de:ConfigurationExtensions>
        <ConfigurationExtensionDeletion>
          <!--deleted extension identifiers -->
        </ConfigurationExtensionDeletion>
        <v8de:ConfigurationExtension>
          <!--Extension data 1 -->
        </v8de:ConfigurationExtension>
        …
        <v8de:ConfigurationExtension>
          <!--Extension data N -->
        </v8de:ConfigurationExtension>
      </v8de:ConfigurationExtensions>
      <v8de:Nodes>
        <v8de:Node>
          <!--Master node data -->
        </v8de:Node>
        <v8de:Node>
          <!--Subordinate node data -->
        </v8de:Node>
      </v8de:Nodes>
      <v8de:Data>
        <!--Changed data items -->
      </v8de:Data>
    </v8de:Changes>
  </v8msg:Body>
</v8msg:Message>
```

As you can see from the example, all feature of data exchange message used in distributed infobase are in the message body. The message body (`Body` element) only contains one XML element - `Changes`. All data sent during the data exchange in distributed infobase are focused in this element:

- `Changes` may contain several nested items associated with the same namespace:

- `Signature` contains "signature" of the exchange plan that was used to get the message.
- `Config` contains configuration changes as well as the data identifying configuration state.
- Optional `Metadata` items nested in `Config` contain changes of specific configuration objects. If the configuration changes are not sent in the message, the `Metadata` items are not present. Such items can only be included in messages sent from master node to the subordinate node. `Digest1` and `Digest2` items contain digital signatures of configuration changes and the whole configuration excluding changes, which both can be transferred in the message. `Digest1` and `Digest2` items are included in all messages sent from the master node to the subordinate node and vice versa.
- The optional `ConfigurationExtensions` item contains information on the extensions that can be sent within distributed infobase. This item contains several nested `ConfigurationExtension` items and each of them describes one extension. Also, the `ConfigurationExtensionDeletion` item can be used as the subordinate that includes identifier of extension, which should be deleted in the receiving infobase of this message exchange..

- `Nodes` can only be included in messages sent from master node to the subordinate node. This item includes two nested `Node` items. The first one contains master node (sender) data and the second - subordinate node (receiver) data.
- The `Data` item includes changed data items sent in the message. Data items are included in the message using the XML serialization.

15.3.2.5. Configuring exchange plan for distributed infobase operation

As noted above, to use exchange plan in distributed infobase organization, the Distributed infobase property must be set for this exchange plan.

When configuring the exchange plan used to organize the distributed infobase, define the exchange plan content is a must, because this content defines the variety of data used to keep the changes register and the data to be sent during the data exchange in the distributed infobase.

Note that if the exchange plan of the configuration that already uses distributed infobase includes another metadata object, this will not result in automated registration of changes in data item corresponding to this metadata object. Thus, if you include the new object in the exchange plan content and you need to transfer the existing data to other nodes, ensure this individually.

15.3.2.6. Exchange plan event handlers

When needed the ExchangePlansObject.<Exchange plan name> object may have the following event handlers: OnSendDataToSlave, OnSendDataToMaster, OnReceiveDataFromSlave, OnReceiveDataFromMaster. They allow to manage placement of data item into the message and reading the separate data item.

The OnSendDataToSlave event handler is called when placing data item into the message sent to the subordinate node of the distributed infobase. The first parameter contains the data item itself. The value of the second parameter when calling the handler is DataItemSend.Auto, but this value can be changed by the handler. If the DataItemSend.Auto is still the second parameter value after the handler execution, this is a default behavior (the data item will be placed into the message). If the second parameter is set to DataItemSend.Delete by the handler the message will include the object that corresponds to the deletion of data item. For database objects - it's ObjectDeletion object type initialized by the link to the deleted database item, and for the record set - it's an empty record set. The behavior corresponding to DataItemSend.Delete, value only is the same as behavior corresponding to DataItemSend.Auto value for constant record manager. If the second parameter will be set to DataItemSend.Ignore, the message will not include anything that corresponds to the current data item sent in the first parameter.

The OnSendDataToMaster event handler is only different from the previous one because it's called when the data item is transferred to the message sent to the master node.

The OnReceiveDataFromSlave handler is called after reading the data item from the message and before writing the item in the database. The first parameter includes data element read from the message. The second parameter, when calling the handler, is set to DataItemReceive.Auto, which corresponds to the default behavior. It means that if the data item change for sending node was not registered in this node, the element will be written to the database. If the changes are registered, the item will not be written to the database. The value of the second parameter can be changed by the handler. If the second parameter value is set to DataItemReceive.Accept, the data item will be written to the database irregardless of the sending node changes registration. If the changes were registered, the change registration records is deleted. If the second parameter is set to DataItemReceive.Ignor, no actions will be taken (the item will not be written in the database and no actions with the changes registration records will be taken). The third parameter allows to manage data item changes registration for the sending node. When calling the handler, this parameter value is set to False. If this value is not changed by the handler, no additional actions are

taken. If the parameter value is set to `True` in the handler, there will be no registration in case there are no registered changes of the data item for sending node. The `OnReceiveDataFromMaster` event handler has the same set of parameters as the previous handler. The difference is that this handler is called on reading a message from the master node of the distributed infobase. The actions taken when the data item is received differ too. The `DataItemReceive.Auto` and `DataItemReceive.Accept` values of the second parameter, set by the handler, have the same effect, because according to the accepted strategy for resolving collisions in the subordinate node, the data item read from the message must be written to the database regardless of the changes registration. In other aspect, the parameter values and the sense of actions taken are the same as described above.

15.3.3. Operating distributed infobase

There are several aspects of distributed infobase operation:

- creating an initial image for the distributed infobase subordinate node;
- writing data exchange message for sending to other distributed infobase node;
- reading data exchange message sent from other distributed infobase node.

If you need 1C:Enterprise language to access universal mechanism features, the described actions may be taken using the 1C:Enterprise language means or interactively with the More menu commands (All actions) of the exchange plan list form or other means available during configuration.
Besides, the master node can be set and retrieved for each distributed infobase node. This operation can be considered the main operation of the distributed infobase, so there are no interactive action for it.

15.3.3.1. Distributed infobase operating using 1C:Enterprise language

15.3.3.1.1. Creating initial image of distributed infobase subordinate node

To create the initial image of the distributed infobase subordinate node, the `ExchangePlansManager` object has the `CreateInitialImage()` method.

The first parameter of this method must be passed as `ExchangePlansLink.<Exchange plan name>`-type value, which represents the link to the subordinate node of the distributed infobase or `ExchangePlansObject.<Exchange plan name>`-type value which represents this node.
The second parameter must contain the connection string identifying the infobase, where the initial image of the subordinate node will be located. The infobase used to create the initial image must not be empty or non-existent.
As noted above, the infobase configuration is transferred from t. The data corresponding to the metadata objects that are not included into the exchange plan will not be placed in the initial image. Besides, when the data items are transferred to the initial image, the `OnSendDataToSlave` event handler is called for each data item. It means that the initial image includes the data that are supposed to get there according to the rules stipulated by the set exchange plan.
In the exchange plan used to create the initial image, two nodes will be created and initialized in the initial image infobase. One for the master node and another one for "this" node. Thus, the infobase of the subordinate node is ready to exchange data with the master node right after creating the initial image.
All operations for transferring data to the initial image are performed within one transaction of the master node database. This ensures consistency of data placed in the initial image. However, when the master node infobase data are changed a lot by other users, there might be conflicts between transactions of initial image creation and other users transactions. In such cases, before creating the initial image, set the exclusive mode for the master node infobase.

NOTE. Switching the infobase to exclusive mode does not automatically switch the MS SQL database to single-user mode.

15.3.3.1.2. Writing data exchange message in distributed infobase

As shown above, the data exchange message of the distributed infobase is different from the general data exchange message because it has different message body content. The procedure for writing data exchange message of the distributed infobase is different from the general message writing in the similar way.
To write the data exchange messages of the distributed infobase, the exchange plan manager uses `WriteChanges()` method. The `ExchangeMessageWriter`-type object used to write messages is passed to this method as the first parameter. The `BeginWrite()` method must be called for this object, but `EndWrite()` method must not be called for this project yet.

The optional second parameter specifies the maximum number of data items, which will be placed in the message within one transaction. If the parameter value is set to 0 (default value), the whole message generation will be performed within one transaction. This option ensures the best consistency of data placed in the message, but conflicts with transactions performed by other users are possible. The fewer items processed within one transaction there are, the less the probability of transaction conflicts. However, in this case the likelihood of placing inconsistent data in the message is higher. That's why it's recommended to not use custom second parameter values without urgent need.

The `WriteChanges()` method writes `Changes` XML item into the data exchange message body by placing all the required information there, as shown above. The configuration objects and data elements with changes registered for the message receiving node are placed in the message fragments. The actual placement of data in the message is defined by the result of `OnSendDataToMaster` (`OnSendDataToSlave`) event handler executions called from the `ExchangePlansObject.<Exchange plan name>`-type object representing the message receiving node.

Below is the standard code fragment that writes data exchange message of the distributed infobase.

```
WriteXML = New WriteXML();
WriteXML.OpenFile(MessageFileName);

Node = ExchangePlans.RemoteWareHouses.FindByCode(NodeCode);
MessageRecord = ExchangePlans.CreateMessageWriter();

MessageRecord.BeginWrite(WriteXML, Node);
ExchangePlans.WriteChanges(MessageRecord);

MessageRecord.EndWrite();
```

15.3.3.1.3. Reading data exchange message in distributed infobase

To read the body of data exchange messages of the distributed infobase, the exchange plan manager uses `ReadChanges()` method.

The `ExchangeMessageReader`-type object used to read the message in general is passed as the first method parameter. The `BeginRead()` method must be called for this object, but `EndRead()` object must not be called for this object.

TIP. When calling the `BeginRead()` method of the `ExchangeMessageReader` object, it's highly recommended not to set the

second parameter value or to set the default value - `AllowedMessageNo.Greater`.

The reason is that the whole data exchange logic of distributed infobase considers that data exchange messages might be lost, but the repeated receival of the same message is not allowed.
The value of the second, optional, parameter can be set to the maximum number of data items that are read from messages and placed into the database within one transaction. The default parameter value is 0 and it means that all message reading is performed within one transaction. If all reading is performed in one transaction, then - in case of errors - the data items can't be read and placed in the database partially. However, in this mode, the number of database changes to be implemented within one transaction might be too large. Besides, the likelihood of conflict between transactions with message reading and transactions performed by other users is higher. To avoid such issues you can limit the number of data items processed in one transaction. If there's no urgent need, we recommend using the default mode, i.e. reading all data items in one transaction.
The sequence of actions taken by the `ReadChanges()` method:

- The exchange plan "signature" is read and verified to ensure that the message comes from the expected exchange plan of the expected configuration.
- The configuration changes are read from the message. When reading, the digital configuration signatures are verified to ensure compatibility of configurations in the sending and current distributed infobase nodes. For each of the configuration changes read, it's verified, whether the configuration object actually changed. Remember that the changed configuration objects can be included in the messages sent from the master node to the subordinate node.
- The change registration records of objects and data items sent in the messages. for which the receival confirmation is received, are deleted. Remember that the maximum number received by the message sending node is included in the message header and is available via `ReceivedNo` property of the `ExchangeMessageReader` object.
- If the configuration in the current distributed infobase node is different from the configuration of the sending node and it becomes apparent, there are two options:
  - If the current node is a master node with respect to the sending one, this message can in no event be further accepted.
  - If the current node is subordinate to the sending node, then before continuing the receival you have to update the database configuration to align it with the sending node configuration. When database configura-

tion is getting updated, previously received, saved, and modified configuration objects will be transferred to the database. Next, in both cases the exception with the error message is called. In the first case, the message can be read, but in the second case this message can be read and accepted after updating the database configuration in the Designer mode.

- When receiving data exchange message sent from the master node of the distributed infobase, the data of the current node and the sending node are aligned with the data on these nodes, contained in the data exchange message. If the data exchange message is received from the subordinate node, such data shouldn't be there.
- The data items are read from the message and written to the data items database. The message may only contain data items that correspond to the metadata objects included in the exchange plan content, associated with this message. For each data item read from the message of the `ExchangePlansObject.<Exchange plan name>`-type object the `OnReceiveDataFromMaster` (`OnReceiveDataFromSlave`) event handler is called. Further actions on each data item are defined by the result of handler actions.

15.3.3.1.4. Getting and setting the distributed infobase master node

The distributed infobase root node lacks the master node. All other nodes have a master node. In the standard scenario of distributed infobase operation, you wouldn't need to force the master node setting. However, in some cases this may come in handy. For example, you may need to separate a subtree of the distributed infobase into an independent infobase or to reassign a distributed infobase node.

The `SetMasterNode()` method of the `ExchangePlansManager` object is used t set the distributed infobase master node. The method has one parameter. If the value of `ExchangePlansObject.<Exchange plan name>` or `ExchangePlansLink.<Exchange plan name>` type is passed to this method as a parameter, the exchange plan associated with the link of object must have the Distributed infobase property set. In this case the master node will be set for the infobase. If the value of `Undefined` type is passed, the master node setting is cancelled.

To execute this method successfully there must be no active infobase users, including the Designer mode.

The `MasterNode()` method of `ExchangePlansManager` object is designed to get the master node. If the current infobase is not the distributed infobase node or the master node is not set for it (it's the root node), the method returns `Undefined`. If the master node is set for the infobase, the method returns value of `ExchangePlansLink.<Exchange plan name>` type.

NOTE. To cancel the setting of the infobase master node, use /ResetMasterNode parameter of the Designer batch mode command line.

In case the predefined data are used in the distributed infobase, the operation with the master node must be performed in a specific way:

- Disconnecting the infobase from the distributed infobase:
  1. For infobases used in 1C:Enterprise 8.3.3 to 8.3.5 (regardless of the compatibility mode), the `SetPredefinedDataInitialization(True)` method must be called for all configuration objects, for which the predefined items are created, and Update predefined data property is set to Auto;
  2. Disable the infobase by calling `SetMasterNode(Undefined)` method or using /ResetMasterNode parameter of the Designer batch mode command line.
- Restoring configuration in distributed infobase node:
  1. Set the infobase predefined values updating mode to `PredefinedDataUpdate.DontAutoUpdate`. This can be done using `SetInfoBasePredefinedDataUpdate()` language method or with the /SetPredefinedDataUpdate -DoNotUpdateAutomatically parameter of the Designer batch mode command line.
  2. Get the link to infobase master node using `MasterNode()` method and remember the link.
  3. Disable the infobase by calling `SetMasterNode(Undefined)` method or using /ResetMasterNode parameter of the Designer batch mode command line.
  4. Restore configuration and perform other operation to restore node performance.
  5. Restore the link to master node using `SetMasterNode()` method and by setting the link to the node, which was not previously set as the master node (retrieved on Step 2), as a link parameter.
  6. Set the infobase predefined values update mode to `PredefinedDataUpdate.Auto`. This can be done using `SetInfoBasePredefinedDataUpdate()` language method or with the /SetPredefinedDataUpdate -Auto parameter of the Designer batch mode command line.

15.3.3.2. Interactive infobase operation

As noted above, a number of frequently used distributed infobase operations can be performed interactively with More (All actions) menu of the exchange plan list form (or list form icon on the command bar). The commands are inserted into the menu during automated filling.

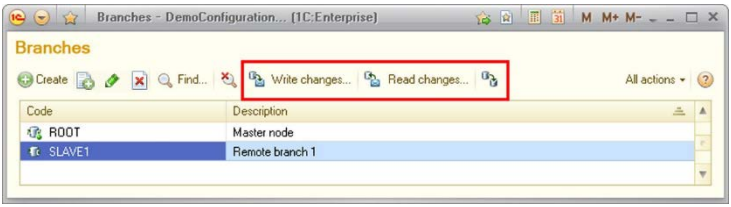


Fig. 277. Exchange plan list

To execute this commands, choose the exchange plan node, for which the command is executed, and choose the required item from More (Actions) menu.

The More - Create initial image... (All actions - Create initial image...) command creates initial image of the distributed infobase subordinate node. Upon calling this menu item, the dialog box will be displayed for choosing the infobase location type and its parameters (when creating an image in client/server mode). After clicking Create initial image the initaial image creation will be started in this dialog box. The procedure for creating an initial image interactively is the same as when you call `CreateInitialImage()` method of the `ExchangePlansManager` object.

The More - Write changes... (All actions - Write changes...) command is designed for writing in exchange message file. The dialog box will be displayed after calling clicking item. Here you have to set the number of data items processed in one transaction. Then click Write and save to file button. This opens dialog box for choosing an exchange file. After choosing, click Open and the data upload will begin.

The More - Read changes... (All actions - Read...) command is designed for reading from exchange message file. The dialog box will be displayed after calling clicking item. Here you have to set the number of data items processed in one transaction. Then click Choose file and read changes. This opens dialog box for choosing an exchange file. After choosing, click Open and the data download will begin.

All data upload/download operations (including initial infobase image creation) are performed on the 1C:Enterprise server side.

15.3.4. Scenarios of data exchange in the distributed infobase

The `OnSendDataToSlave`, `OnSendDataToMaster`, `OnReceiveDataFromSlave`, and `OnReceiveDataFromMaster` event handlers ensure flexible management of data exchange in the distributed infobase. A large number of scenarios can be performed using these handlers. This sections describes several sample scenarios.

15.3.4.1. Default behavior

This scenario is the simplest one and it corresponds to the default behavior of distributed infobase. There are several aspects of this scenario:

- each data item change made in any distributed infobase node tries to distribute itself to all nodes;
- collisions are resolved based on the "master - subordinate" node relations.

To implement such scenario, all handlers must not change the values of the parameter passed to them. The handlers can also be not set at all.

15.3.4.2. Data distribution between subordinate nodes

This scenario implies that for some data items, for which it's implemented, the following is performed:

- all data items are in the master node;
- the presence of data item in subordinate nodes is defined by comparing values of some attributes with the attribute of the exchange plan subordinate node;
- collisions are resolved based on the "master - subordinate" node relations.

To implement this scenario, ensure that when writing data exchange message in the master node, the data items, which should be in the subordinate node, don't get to the message.

Besides, if the attribute values of the data item can be changed in the subordinate node, you need to ensure that when data exchange messages are received, the master node registered changes of objects that shouldn't be in the subordinate node according to their values and attributes.

To dig deeper into this example, let's suppose that we use `Invoice` document as a type of data item, for which the scenario is implemented. This document has a `Warehouse` attribute of `CatalogRef.Warehouses` type. The data exchange is organized in accordance with the `Warehouses` exchange plan. This exchange plan also has a `Warehouse` attribute of

CatalogRef.Warehouses type specified. The exchange plan is used to organize a distributed infobase - the root node is the central office, and subordinate nodes are warehouses. For each subordinate node of the exchange plan, the Warehouse attribute is set to identify, which warehouse correspond to this node. All Invoice documents must be present in the root node, and the conditions of documents presence in the root node is the equality of Warehouse attribute values in the document and exchange plan node.

In this case, to ensure that Invoice documents don't get to the subordinate nodes, where they shouldn't get, the OnSendDataToSlave event handler must look as follows:

```
Procedure OnSendDataToSlave(DataItem, ItemSend)

  DataType = TypeOf(DataItem);
  If DataType = Type("DocumentObject.Invoice") Then

    If DataItem.Warehouse <> Warehouse Then

      ItemSend = DataItemSend.Delete;

    EndIf;

  EndIf;

EndProcedure
```

In the sample handler, the data type is analyzed and if it's equal to DocumentObject.Invoice, the value of Warehouse attribute of the document are compared to the value of Warehouse attribute of the exchange plan node. If the attribute values are equal, the ItemSend parameter value can stay unchanged (when calling the parameter is set to DataItemSend.Auto). The XML presentation of the document will be included in the message. If the attribute values are not equal, the ItemSend parameter is set to DataItemSend.Delete. In this case the message will include XML presentation of the ObjectDeletion object, initialized by the link to the Invoice document.

In may seem strange that in case of Warehouse attribute value inequality, the ItemSend parameter is set to DataItemSend.Delete not DataItemSend.Ignore, because in case of such value as DataItemSend.Delete, the XML presentation of the ObjectDeletion object will be placed in the messages, sent to all subordinate node, except the node, to which the document itself is sent. Thus, in many cases the ObjectDeletion object will be sent to the nodes that never contained the document that has to be deleted.

This is true, but in this example we review the most common case. If it's known that the value of Warehouse attribute of Invoice document can only be set when the document is created can't be changed later then the ItemSend parameter in this handler can really be set to DataItemSend.Ignore.

If the value of the Warehouse attribute of the Invoice document can be changed in the subordinate node, then the OnReceiveDataFromSlave event handler must be specified in the exchange plan. This even handler must look as follows:

```
Procedure OnReceiveDataFromSlave(DataItem,  ItemReceive, SendBack)

  DataType = TypeOf(DataItem);
  If DataType = Type("DocumentObject.Invoice") Then

    If DataItem.Warehouse <> Warehouse Then

      SendBack = True;

    EndIf;

  EndIf;

EndProcedure
```

In the sample handler, the data type is analyzed and if it's equal to DocumentObject.Invoice, the value of Warehouse attribute of the document are compared to the value of Warehouse attribute of the exchange plan node. If the attribute values are equal, then value of ItemReceive and SendBack parameters can stay the same and this will ensure the default behavior when receiving the data item. If the attribute values are note equal, the SendBack parameter is set to True. This ensures that document changes will be registered and when sending messages to the subordinate node the ObjectDeletion object will be sent, providing the Warehouse document attribute is not changed in the main node in a way that it's equal to the value of Warehouse attribute corresponding to the exchange plan node.

If the value of Warehouse attribute of the Invoice document can't be changed after creating the document, the OnReceiveDataFromSlave handler specification is optional.

15.3.4.3. Unusual resolving of collisions

This scenario implies that for some data items, for which it's implemented, the following is performed:

- each data item change made in any distributed infobase node tries to distribute itself to all nodes;

- collisions are resolved based on the "master - subordinate" relations but the subordinate node has a higher priority.

To review this case, let's use the example with `Invoice` document and `Warehouses` exchange plan above.

In this case, the `OnReceiveDataFromSlave` and `OnReceiveDataFromMaster` event handlers must be specified. The `OnReceiveDataFromSlave` handler will look as follows:

```
Procedure OnReceiveDataFromSlave(DataItem,  ItemReceive, SendBack)

  DataType = TypeOf(DataItem);
  If DataType = Type("DocumentObject.Invoice") Then

    ItemReceive = DataItemReceive.Accept;

  EndIf;

EndProcedure
```

This handler is quite simple: the data item type is verified and if the data type is the one you need, the `ItemReceive` parameter value is set to `DataItemReceive.Accept`, which results in unconditional acceptance of data item, regardless of whether its changes were registered.

The `OnReceiveDataFromMaster` event handler looks as follows:

```
Procedure OnReceiveDataFromMaster(DataItem,  ItemReceive, SendBack)

  DataType = TypeOf(DataItem);
  If DataType = Type("DocumentObject.Invoice") Then

    If ExchangePlans.IsChangeRecorded(Link,DataItem) Then

      ItemReceive = DataItemReceive.Ignore;

    EndIf;

  EndIf;

EndProcedure
```

This handler is more complex. If the data item refers to the type needed, the verification is performed: whether the changes of data item are registered for the node that sent the message. If the changes are registered, the `ItemReceive` parameter value is set to `DataItemReceive.Ignore`. As a result, the data item read is not written in the database, and changes registration is saved, which allows to place data item into the message sent to the master node.

15.3.4.4. Other scenarios

These are some of the possible scenarios for organization of data exchange in the distributed infobase. The examples above can be used by the developers to implement various scenarios. Some of the may combine the above scenarios - some may be completely different.

Chapter 16. Operation various data formats

16.1. XDTO mechanism

16.1.1. General information

The XDTO functionality is a versatile way of presenting data for interacting with various external data sources and software systems.
XDTO means XML Data Transfer Objects.
The XDTO mechanism allows to create data presentation model (types and values model), which will provide an ability to easily change data in 1C:Enterprise and is well adapted to convert data from one formats to others, mainly XML, in a transparent way.

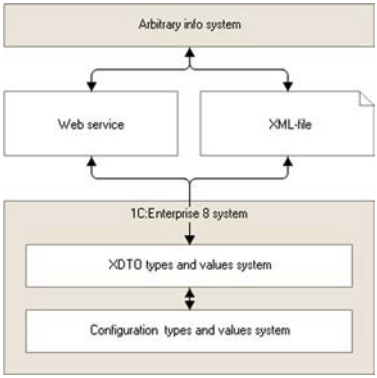


Fig. 278. General structure of XDTO

There are seceral tasks that XDTO mechanism performs:

- data exchange between 1C:Enterprise configurations with different ER diagrams;
- data exchange based on XML schemas that are not linked to configurations (for example, exchange with third-party information systems).

- enabling operations with Web services. XDTO mechanism allows to describe types of parameters and return values of Web services as well as manipulate the data sent and returned.

The key features of XDTO mechanism:

- Enables operations with XML,
- The usual data operation model.

Currently, the data exchange with various platforms and systems is implemented via XML: XML documents are designed to present data, while the XML schema is used to describe data formats and structures. The XDTO mechanism allows to create the XML schemas required for exchange and format XML documents that satisfy these schemes.

At the same time, the XDTO mechanism allows to perform these actions in a manner that most 1C:Enterprise developers know.

The developers deals with data types and objects, data items contain properties, the values are set for the properties and so on. When manipulating data using the XDTO, the developer is, at the most, free of details associated with the way these data are presented in the XML. Naturally, you can't get rid of these details completely, but it's crucial that they only appear where it's actually required.

16.1.2. XDTO factory

16.1.2.1. General information

The key concept of XDTO mechanism is the XDTO factory. The XDTO factory contains description of all types operated by the system. In particular, each 1C:Enterprise configuration has a global XDTO factory, which writes all types used in the configuration in XDTO terms (this XDTO factory is available via `XDTOFactory` property f the global context).

All types descriptions, included in the XDTO factory, are grouped in one or several XDTO packages. When comparing XDTO and XML, you can say that XDTO package is similar to XML schema. Thus, the XDTO factory may correspond to several XML schemas.

The XDTO factory is completely self-scufficient. It means that each type registered in XDTO factory can only refer to the types from this XDTO factory.

In general the XDTO factory is created once based on the description of all types that must be registered in the factory. To create the XDTO factory using 1C:Enterprise language means, the `XDTOFactory` object wizard is used, which receives a set of XML schemas contained in `XMLSchemaSet` object. The scenario, where the XDTO types are added in the factory one by one or in groups is not supported.

As opposed to the arbitrary XDTO factory, which can be created by the developer, the global XDTO factory is created automatically by the system when the

new infobase is created and it allows adding XDTO types one by one or in groups. The visual construction means are used for this, which allow to add XDTO packages in the General - XDTO packages branch of the metadata tree.
All packages in the global XDTO factory can be divided into three types:

- A single XDTO package containing description of platform types. This package is the same for all 1C:Enterprise configurations.
- A single XDTO package containing description of configuration types created as a result of metadata editing (creating and changing properties of catalogs, documents and so on).
- One or several XDTO packages described directly in the General - XDTO packages branch of the configuration object tree.

The XDTO package contains description of some set of types belonging to one namespace - the package namespace. Apart from the XDTO packages description, it may contain links to packages, which are used by this package as well as the list of global package properties definitions.
Links to other packages are contained in the `Dependencies` property of the XDTO package. It is the `XDTOPackageCollection` object. The packages of this collection include types from the namespace, for which there are links in this package.
The package may include links to global properties from other packages.
The loops in XSD schemas importing and including directives are not supported.

16.1.2.2. Retrieving XDTO factory from XSD schema file

Below is the example of XDTO factory creation based on the XML schema from the XML file. Because the XDTO mechanism is an abstraction based "above" the XML, to retrieve the XML schema from the XML file, you need to sequentially "pass" several layers of XML data operation:

- First, low-level reading/writing XML files;
- Then the XML object model, from which the `XMLSchema` 1C:Enterprise language object can be retrieve that contains XML schema data.

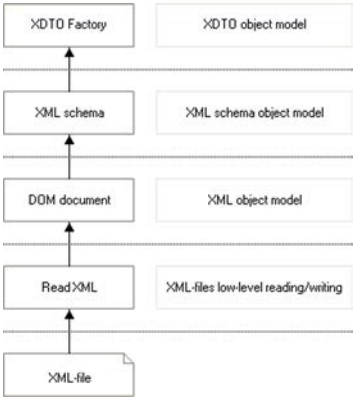


Fig. 279. Creating XDTO factory

Example:

```
// Create the XDTO factory based on the XML schema
// from the XML file

// Create the default XML reading object
ReadXML = New ReadXML;
// Open the XML file
ReadXML.OpenFile("c:\temp\my schema.xsd");
// Create the default DOM document builder
DOMBuilder = New DOMBuilder;
// Read XML file in DOM document
DOMDocument = DOMBuilder.Read(ReadXML);
// Create the default XML schema builder
XMLSchemaBuilder = New XMLSchemaBuilder;
// Retrieve the XML schema from DOM document
XMLSchema = XMLSchemaBuilder.CreateXMLSchema(DOMDocument);
// Create the default XML schemas set
XMLSchemaSet = New XMLSchemaSet;
// Add XML schema in the XML schemas set
XMLSchemaSet.Add(XMLSchema);
// Create XDTO factory based on the XML schemas set
NewXDTOFactory = New XDTOFactory(XMLSchemaSet);
```

In the example above, the `ReadXML` object is created and then the XML file located on the disk is opened. After this the DOM document builder is used to create the `DOMDocument` object containing XML file data. Then the XML schema builder uses the DOM document as a basis to create the new `XMLSchema` object containing XML scheme data. Finally, the empty set of XML schemas is created where the existing XML schema is included and the XDTO factory is created based on this set.

16.1.2.3. Retrieving the XDTO factory from multiple sources

In some cases you might need to create a factory based on several schemas. Some of them might be provided as a file and some need to be "inherited" from other XDTO factories (including the global factory).

```
Schemas = New Array;
Schemas.Add("c:\temp\schema 1.xsd");
Schemas.Add("c:\temp\schema 2.xsd");
Packages = New Array;
Packages.Add(XDTOFactory.Packages.Get("Package 1 URI"));
Packages.Add(XDTOFactory.Packages.Get("Package 2 URI"));
MyFactory = CreateXDTOFactory(Schemas, Packages);
```

In this example, the XDTO factory is created based on 4 sources:

- XML schema located in c:\temp\schema 1.xsd file.
- XML schema located in c:\temp\schema 2.xsd file.
- Schemas from package 1 of the current configuration.
- Schemas from package 2 of the current configuration.

In general, it's not required to retrieve packages from the global XDTO factory. This can be a custom factory.

16.1.3. XDTO data types

16.1.3.1. General information

Each type of XDTO data is either a XDTO value type or XDTO object type. Thus, to describe the value type, the `XDTOValueType` object is used, and to describe the object type - `XDTOObjectType`.

The `XDTOValueType` object is used to describe types of simple indivisible values, which can't be divided into parts. These are, for example, various strings, numbers, date and so on.

The `XDTOObjectType` object is used to describe to types of data instances that have a state presented as a set of this data instance property values. The property types of this data instance can be both XDTO value types and XDTO object types.

The `XDTOValueType` and `XDTOObjectType` has the same properties:

- `Name` - type name;
- `NamespaceURI` - URI of the namespace, where this type is defined.

Value of these properties correspond to the similar parameters identified for the type in the XML schema. The type name and namespace URI create a unique type identifier. The type name must be set. The `NamespaceURI` property may contain an emty string, although this is undesirable.

16.1.3.2. XDTO value type

The type of XDTO value in accordance with the rules for simple type from XML schema can be defined in the following ways:

- by limiting while setting base type (BaseType property) and a set of limits for the set of possible values (Facets property);
- by merging when the type is the result of combining several types of values (the combined types are listed in the MemberTypes property);
- by the list when the value is the list of values (the type of item values constituting the values list is set in the ListItemType property).

Apart from Name and NamespaceURI the XDTO value type contains the following properties:

- BaseType - the base type for this XDTO value type. Base type can be inherited but only from other types of XDTO values. The acceptable set of values of the inherited type is the subset of possible base value set. The upper level of the simple types hierarchy is the predefined anySimpleType type from http://www.w3.org/2001/XMLSchema namespace. All value types are directly or indirectly inherited from this type. The types created by merging or by the list are always directly inherited from anySimpleType.
- Facets - the list of facets limiting the set of possible values against the base type. The list of facets is only set for XDTO value types that are defined by limiting the base type. Each separate facet if the pair: facet name and value. The list of possible facet names is defined. Not all available facets can be aplied to the type. The list of facets and their applicability to some type are defined by the XML Schema rules (http://www.w3.org/TR/xmlschema-2/).
- MemberTypes - list of types constituting the merging. Only XDTO value types can be combined. If the type is created by merging, the MemberTypes list includes at least one type. The Facets list must be empty and ListItemType property must retrieve the undefined value.
- ListItemType - in case the XDTO value type is defined by the list, this property shows the type of list item. In this case the Facets and MemberTypes lists must be empty.
- The list of available facets names (defined by the XDTOFacetType system enumeration):
  - Length - length facet. Contains a number of length units. The length unit has different meaning for different types. For string and anyURI types, the length includes the number of characters. For

`hexBinary` and `base64Binary` types, the length includes the number of binary data bytes. For the types defined by the list, the length includes the number of list items.

- `MaxInclusive` - the facet of the maximum including the limit. Limits the value space of this type with the maximum value. Any value of this type is less or equal to the specified value.
- `MaxLength` - the facet of maximum length. Includes the maximum number of length units. The length unit has different meaning for different types. For `string` type the maximum length includes the maximum number of characters. For `hexBinary` and `base64Binary` types, the maximum length includes the maximum number of binary data bytes. For the types defined by the list, the maximum length includes the maximum number of list items.
- `MaxExclusive` - the facet of maximum excluding the limit. Limits the value space of this type with the maximum value. Any value of this type is less than the specified value.
- `MinInclusive` - the facet of minimum including the limit Limits the value space of this type with the minimum value. Any value of this type is larger or equal to the specified value.
- `MinLength` - the facet of minimal length. Includes the minimum number of length units. The length unit has different meaning for different types. For `string` type the minimum length includes the minimum number of characters. For `hexBinary` and `base64Binary` types, the minimum length includes the minimum number of binary data bytes. For the types defined by the list, the minimum length includes the minimum number of list items.
- `MinExclusive` - the facet of minimum excluding the limit. Limits the value space of this type with the minimum value. Any value of this type is larger than the specified value.
- `Pattern` - the sample facet. Includes regular expression that defines the value space of this type.
- `Enumeration` - the enumeration facet. Defines the set of acceptable values for this type.
- `SpaceCharacters` - the space characters facet. It can take the following three values:
  - `Save` - the string may contain any space characters.
  - `Replace` - the string must not contain #x9 (tab characters), #xA (feed characters) и #xD (carriage return). If they are present, they must be replaced with #x20 character (space).

    - Collapse - in addition to requirement for the replace value, the string must not contain the #x20 (space) double characters, as well as leading and closing #x20 (space) characters.
  - TotalDigits - the facet of total number of digits. Includes the total number of digits (both integer and fractional part).
  - FractionDigits - the facet of fraction digits number. Includes the number of digits in the fractional part of the number.

The XDTO infrastructure defines a set of predefined XDTO value types. This set corresponds to the set of primitive types defined in XML Schema Part 2: Datatypes. The predefined types create a hierarchy in accordance with the XML Schema Part 2: Datatypes. The type names correspond to the type names in XML Schema and belong to namespace URI - http://www.w3.org/2001/XMLSchema. The predefined types are automatically registered in any XDTO factory.

16.1.3.3. XDTO object type

Apart from Name and NamespaceURI properties the type of XDTO object includes the following properties:

- BaseType - base type for this type. This may be XDTO object type only. Th base type in XDTO object hierarchy is the predefined anyType type from http://www.w3.org/2001/XMLSchema namespace. All XDTO object types are inherited from this type directly or indirectly.
- Open - flag, showing whether the XDTO object type is open. This property indicates whether an instance of an XDTO object can contain additional properties that are not defined in its type, meaning it implements the open content model. It corresponds to appearance of the following in the XML schema of this description type: <anyAttribute>, <any>.
- Abstract - flag, showing if the XDTO object type is abstract. It corresponds to appearance of the abstract="true" attribute item in the schema.
- Ordered - flag of whether the sequence of the items representing property values is strictly the order with the XDTO object type properties. If the xsd:all content model i set then the order of XML elements can be arbitrary. The acceptable order corresponds to the order of properties in the type. If the Ordered property is False, then the sequence of the XML elements is not controlled at the input, and at the output it is determined by the sequence of the properties, unless the Serial flag is True.
- Serial - this property indicates whether the instance of the corresponding XDTO object contains the XDTO sequence. This flag is True when the order of nested XML elements can't be unambiguously defined by order of properties in the type (for example, in XML schema the content is set as

<sequence ... maxOccurs=10 ... >) or the mixed="true" attribute is set for the XML type in the schema. The XDTO sequence allows to explicitly define the sequence of the items as they appear in the XML document. For objects of types where the Serial property is False, the sequence of nested elements corresponds to the sequence of properties.

- Mixed - the property indicates, if the mixed content for this type is defined in the XML schema. If the Mixed property is True, then the Serial value must be True, since the mixed content can't be modeled without using the XDTO sequence.
- Properties - the list of properties defined for this XDTO object type. Each property is presented as an instance of the XDTOProperty object. The list includes all properties, including properties defined in the base type.

There is a predefined XDTO object type called anyType and http://www.w3.org/2001/XMLSchema namespace URI. This is the base type for each XDTO object type, but it lacks the base type. It's open, not abstract. It implies that there is a sequence and has an blank property list.
This XDTO object type corresponds to anyType, type defined in XML Schema Part 2: Datatypes.

16.1.3.4. XDTO property

A specific property of the specific XDTO object is described using the XDTOProperty object. This means that the same instance of XDTOProperty object can't be used to describe properties in different XDTO object types and two different properties of one XDTO object type.
The XDTOProperty object has the follwing properties:

- Name - property name. Within the XDTO object of the same type, property names must be unique.
  When generating the XDTO data model based on the XSD schema, the names of XDTO properties are based on the attributes and items names described in the schema. The model is built sequentially: First, the list of values is created based on the attributes - then based on the items, in the order of presentation in the schema.
  The name is aligned to the 1C:Enterprise language naming rules. Characters, acceptable in XML name (for example: ".", "-«'), but are not acceptable for the 1C:Enterprise language are replaced with the character - "_". In case of attribute and item name doubling, the duplicate is assigned with a name extended by the numeric suffix (starting with 1).
- Type - property type. It can be an instance of the XDTOValueType object of the instance of XDTOObjectType object.

- `UpperRange` - property of an XDTO object type can be defined as containing one or many values. A property is considered to contain one value if it equals 1. If `UpperRange` property is bigger than 1, then it is considered that it may contain multiple values. Such property is modeled as a list in the object structure (don't miss it for the list in XDTO object description). The `UpperRange` property indicates the maximum quantity of the property values. The value larger than 1 can be set only for properties that are represented as an XML item. The `UpperRange` property corresponds to the `xsd:maxOccurs` attribute in XML Schema. The -1 value corresponds to `unbounded`.
- `LowerBoundary` - 1 can be set only for properties that are represented as an XML item. The minimum quantity of property values can be less or equal to 0. Naturally, the `LowerBoundary` value must be less than or equal to the `UpperRange` value (unless, of course, the `UpperRange` is -1);
- `Nillable` - indicates whether the property can have an undefined value. The undefined value of the property in the XML is presented as the element that looks as follows: `<elem xsi:nil="true" />`. Thus, the `Nillable` property, when it's `True`, can only be defined for properties with the `Element` presentation form. The `Nillable` property corresponds to the `xsd:nillable` attribute in the XML Schema. If the value of `UpperRange` property is larger than 1, the undefined value is valid for the property values list item.
- `DefaultValue` - the default value. The only possible default value is `XDTODataValue`. This property must be of the same type as the type of the property or of the inherited type. When creating XDTO object, the property, if it allows the single value, is set to default value (`Set()` method of the XDTO object returns `False` for this property). For properties with multiple values, the list of values is initially empty, regardless of whether the default value is defined or not.
- `Fixed` - indicates whether the property value is fixed. If set to `True`, the fixed value itself can be retrieved through the `DefaultValue` property.
- `Form` - property presentation form in XML. This may `Text`, `Item` or `Attribute`. If the presentation form is `Attribute` or `Text`, the value of `UpperRange` property can't be larger than 1. If the property is set to `Text` then the value of the `LowerBoundary` property must also be 1. One type has the only one property that can be in the `Text` presentation form, while the other properties must be have the `Attribute` presentation form.

- `LocalName` - local name of the attribute or item used to present the property. For the properties with `Text` represnetation form - it is a blank string.
- `NamespaceURI` - URI of the namespace for attribute or item used to present the property. Blank string if there is no namespace.

16.1.4. XDTO data instances

The XDTO data instances can be the XDTO values (`XDTODataValue`) or XDTO objects (`XDTODataObject`).

16.1.4.1. XDTO Value

XDTO values are the simple indivisible values, in which the parts can't be identified. These are, for example, various strings, numbers, date and so on. The instances of simple values are unmutable.

The new XDTO value can be created using the `Create()` method of the XDTO factory:

- based on the XDTO value type and value;
- based on the XDTO value type and lexical presentation of the value.

Below are the examples of XDTO value creation.

Example:

```
GlobalXDTOFactory = XDTOFactory;

// Create XDTO value from the link
CatalogItemReference = Catalogs.Products. FindByCode("0000001");
XDTOValueTypeOfValueCreated = GlobalXDTOFactory.Type("urn:schemas v8
1c ru:config data", "CatalogRef.Products");
NewXDTOValue = GlobalXDTOFactory.Create(XDTOValueTypeOfValueCreated,
CatalogItemReference);

// Create the XDTO value from the lexical presentation
XDTOValueTypeOfValueCreated =
GlobalXDTOFactory.Type("http://www.w3.org/2001/XMLSchema",
"dateTime");
NewXDTOValue = GlobalXDTOFactory.Create(XDTOValueTypeOfValueCreated,
"2006 04 20T12:00:30");
```

The new XDTO value can also be created by reading the XML file.

Example:

```
GlobalXDTOFactory = XDTOFactory;,

// Read data of the XDTO value from XML file
New XMLReader = New ReadXML;
New XMLReader.OpenFile("D:/Exchange.xml");
…

NewXDTOValue = GlobalXDTOFactory. ReadXML(New XMLReader);
XDTO value can be written to the XML file.
GlobalXDTOFactory = XDTOFactory;

// Write XDTO value data to the XML file
NewXMLWriter = New WriteXML;
NewXMLWriter.OpenFile("D:/Exchange.xml");
…

GlobalXDTOFactory.WriteXML(NewXMLWriter, NewXDTOValue);
```

16.1.4.2. XDTO object

In contrast to the simple value, the state of XDTO object is the set of its properties values. Instances of XDTO object are mutable, i.e. during the lifetime the state of XDTO object can be changed by changing values of some of its properties. Any XDTO data instances can be stated as the property values. It can be both XDTO value and XDTO object. When the property value if the XDTO object, it's said that the property value is the link to the object.

New XDTO object can be created using the `Create()` method of the XDTO factory, based on the XDTO object type. After this you should assign the values to XDTO object properties. Below is an example is XDTO object creation and property filling.

Example:

```
GlobalXDTOFactory = XDTOFactory;

// Create an empty XDTO object
XDTOObjectTypeOfObjectCreated =
GlobalXDTOFactory.Type("http://www.1c.ru/demos/products", "Products");
NewObjectXDTO =
GlobalXDTOFactory.Create(XDTOObjectTypeOfObjectCreated);

// Fill the XDTO object property values
CatalogItem = CatalogItemReference.GetObject();

NewObjectXDTO.Name = CatalogItem.Name;
NewObjectXDTO.FullDescr = CatalogItem.FullDescr;
NewObjectXDTO.PurchasePrice = CatalogItem.PurchasePrice;
NewObjectXDTO.Barcode = CatalogItem.Barcode;
```

Just like the XDTO value, the XDTO object data can be read from XML file or written the the XML file.

Example:

```
GlobalXDTOFactory = XDTOFactory;,

// Read the XDTO object data from XML file
New XMLReader = New ReadXML;
New XMLReader.OpenFile("D:/Exchange.xml");
…

NewObjectXDTO = GlobalXDTOFactory.ReadXML(New XMLReader);
…

// Write the XDTO object data to the XML file
NewXMLWriter = New WriteXML;
NewXMLWriter.OpenFile("D:/Exchange.xml");
…

GlobalXDTOFactory.WriteXML(NewXMLWriter,NewObjectXDTO);
```

When reading the untypified data, the item (in case there are attributes or subordinate items) is read into the `XDTODataObject` of `xsd:anyType` type. This is an open type with the mixed content, so the text in the item is not interpreted as the `__content` property value, but as a text and it's placed in the object sequence.

For example, when reading the `<element attr="attr_value">element value</element>` item, the `element value` text can be retrieved in the following way: `XDTODataObject.Sequence().GetText(0)`.

The `XDTODataObject` object contains the following methods:

- `Type()` - retrieves type of this XDTO object (`XDTOObjectType`);

- `SetUp(<Expression>), SetUp(<Property>, <Value>)` - allows to set the property value;
- `Expression` - expression on the XPath, specifying the property;
- `Property` - property name;
- `Value` - the property value set.

If `Expression` is set incorrectly or `Value` can't be assigned to the property (for example, the type is incompatible with the property type), the exception is called. If the property is assigned with undefined value, and the `Nillable` property is `False`, the exception is displayed. If the property is assigned with the link to XDTO object and the link to his XDTO object is already a value of some other property, then this link is no longer a value of this other property.
The chains of links to XDTO objects in the object properties can't form cycles. That's why when assigning links to XDTO object that calls cycle formation, the exceptions is called. If the property allows multiple values, then applying `SetUp()` method is prohibited and results in exception. When assigning value to the property, it's verified whether this type of value can be assigned to the property. The value can be assigned if its type is the same as the property type, is inherited from the property type or is one of the types from merging. When assigning, if the property value presentation form in XML is `Text` or `Attribute`, the value is aligned with the property type. If the presentation form is `Item`, the value is assigned as is.

- `Get(<Property>), Get(<Expression>)` - getting the property value:
  - `Property` - property name.
  - `Expression` - expression on XPath, specifying the property. For properties with multiple values this method returns the - `XDTOList` list of property values. All property values modification operations must be performed via this list.
- `Clear(<Property>), Clear (<Expression>)` - clear property value:
  - `Property` - property name.
  - `Expression` - expression on XPath, specifying the property. The `Clear()` method acts differently for different properties. For properties allowing multiple values (`UpperRange > 1`), executing `Clear()` method results in clearing the list of values.
- `Set()` - verifies, whether the property value is set. After creating an object, the result of `Set()` method will be `False` for all properties.
- `Sequence()` - return sequence object (`XDTOSequence`) owned by this XDTO object. The XDTO sequence is used to modify the object state. This

method returns XDTO sequence only if the `Sequential` property is set for the object type.

- `Verify()` - this method allows to verify validity of XDTO object properties filling. The objects that are the property values are also verified. What is verified: a match between a number of property value and `LowerBoundary` and `UpperRange` properties, validity of property value order in XDTO sequence, if the `Ordered` property is `True`. The verification is stopped after the first error. In this case the exception is displayed.

16.1.4.3. XDTO sequence

The `XDTOSequence` object is used to model the order of items and text fragments as they look in XML presentation of the object. The sequence consists of the "property - value" pairs. Only the properties with the `Item` presentation form can be the properties because attributes order matters. The property in "property - value" pair can have the undefined value. In this case it's considered that this sequence item is a text fragment. Appearance of the sequence items presenting the text fragments is only acceptable for the type objects with `Mixed` property set to `True`.

When generating the XDTO object content by assigning values to properties, the order of assignment is shown in XDTO sequence.

The XDTO sequence includes the following methods:

- `Number()` - returns a number of sequence items.
- `GetProperty(<Index>)` - returns property, corresponding to the value located by `Index` index. If `Index` is outside the allowed values, the exception is displayed. The method can return the undefined value, if the sequence item corresponds to the text fragment from the mixed content (text and items).
- `GetValue(<Index>)` - return the value located by `Index` index. If `Index` is outside the allowed values, the exception is displayed.
- `SetValue(<Index>, <Item>)` - set the `Item` value by `Index` index. `Index` must have the value from within the acceptable indexes range. Item must have the value that's acceptable for the property, for which it's set, or for the text.
- `Add(<Property>, <Item>)` - adds the "property - value" pair to the property. Value must be acceptable for the property.
- `Add(<Text>)` - adds text fragment to the sequence. If the `Mixed` object type property is `False`, the exception is displayed.

- `Insert(<Index>, <Property>, <Item>)` - inserts the "property - value" pair into the `Index` position of the sequence. `Index` must have the value from within the index range. The item in `Index` position and all items with the value that's larger than the index are shifted one position to the right.
- `Insert(<Index>, <Text>)` - inserts `Text` text into `Index` sequence position. `Index` must have the value from within the index range. The item in `Index` position and all items with the value that's larger than the index are shifted one position to the right.
- `Delete(<Index>)` - deletes sequence item in `Index` position. `Index` must have the value from within the acceptable range.

16.1.4.4. XDTO list

The `XDTOList` object is used to model the list of values for properties with multiple values (`UpperRange > 1`). The list is the ordered set of objects, which can be XDTO values or XDTO objects. Among them, there can be undefined values, if the `Nillable` property is `True`. The concept of "setting" is not defined for the list item.

The `XDTOList` object contains the following methods:

- `Number()` - returns list size.
- `Get(<Index>)` - gets the value located by `Index` index. `Index` must be within the allowed values. Otherwise, the exception is displayed.
- `SetUp(<Index>, <Item>)` - sets the `Item` value to `Index` position. The set value replaces the previous value. `Index` must be within the allowed values and `Item` must be allowed for the property. Otherwise, the exception is displayed.
- `Add(<Item>)` - add the value to the end of the list. `Item` must be acceptable for the property. Otherwise, the exception is displayed.
- `Insert(<Index>, <Item>)` - inserting the `Item` value into the `Index` position. `Index` must be within the allowed values and `Item` must be allowed for the property. Otherwise, the exception is displayed. Value in `Index` position and values with the larger positions are shifted one position to the right.
- `Delete(<Index>)` - deletes values in `Index` position. `Index` must be within the allowed values. Values with larger positions are shifted to the free space.

16.1.4.5. XPath

To navigate the object tree, expressions on the XPath can be used. Strictly speaking, this is not exactly the XPath, but rather modified XPath subset.

The main construction of this language is the path to the value that consists of the separate steps. The steps in the path are divided with slashes (/ character). The path step is the property name or predefined constructions: . (dot) and .. (two dots).

The expression below denotes the `PropertyName` property of the current object, particularly an object with `Get()` or `Set()` method called.

```
PropertyName
```

The expression below means that for the current object the `PropertyName1` property value is received, and for the object with the link that is the value of `PropertyName1` property the `PropertyName2` property is received.

```
PropertyName1/PropertyName2
```

The step denoted as a dot means the current object and two dots - the object that owns the current object.

If the search path starts with slash, it means searching from the object tree root. If some property on the path is not found, this calls an exception. If there is a property with multiple values on the path, the result is the whole list of values of this property.

For example, if in the path below, the `List` property has multiple values - the result of this expression is the whole list (`XDTOList`) of this property's values.

```
Property/List
```

To get one value from this list, specify the 0-based index of the value in the list in the property name and separate it with the dot, as shown below.

```
Property/List.0
```

The index must be set as an integer within the range of acceptable indexes. Otherwise, an exception is thrown.

You can get one value form the list using the 1-based index. To do this, the following construction is used:

```
Property/List[1]
```

Index can have the value from 1 to the number of list items.

You can also search in the list (only for objects). The expression for searching looks as follows:

```
Property/List[PropertyName='SearchString']
```

Here, the `List` - a property with multiple values. The value of the list are the objects with `PropertyName` property. The result is the first object in the list with the `Property` property set to 'SearchString' string value. If no objects are found, the result is undefined value. Some objects in the list might not have the `PropertyName` property. Also, none of the objects may have it. The value, compared to the property value, can be set as a number, logical value (`True` or `False`) or string literal.

```
Property/List[PropertyName!='SearchString ']
```

The expression above is the same as the previous one, except the result is the first object in the list with `PropertyName` property value not set to 'SearchString ' string value.
Below is the definition of the XPath set described.

<Path>

```
[/] <List of steps>
```

<List of steps>

```
<Step> [/<List of steps>] |
```

<Step>

```
<Property name> [<Clarification>] | .. | . |
```

<Property name>

```
[<Letter> | _]<Rest of the name>
```

<Rest of the name>

```
{<Letter> | <Digit> | _} <Rest of the name> |
```

<Clarification>

```
.<0 based index> |
[<Property name>=<Value>] |
[<Property name>!=<Value>] |
[<1 based index>]
```

<0-based index>

```
<Unsigned integer>
```

<1-based index>

```
<Unsigned integer>
```

<Unsigned integer>

```
<Digit> <Digits>
```

<Digit>

```
0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

<Digits>

```
<Digit> <Digits> |
```

<Value>

```
<Number> | <String> | <Boolean>
```

<Number>

```
[+| ]<Unsigned integer>[.<Unsigned integer>]
```

<String>

```
"<Characters>" | '<Characters>'
```

<Boolean>

```
true | false
```

NOTE. The string with the " delimiters can't include " characters. Similarly, the string with ' delimiters can't include' characters.

When comparing the property value with the value set as a literal, the value set as a literal is converted by the behavior rules, and then the comparison is performed.

16.1.5. XDTO-based XML serializer

The values of 1C:Enterprise configuration types can be serialized directly into(from) XML file based on the XDTO.
To do this, the `XDTOSerializer` object is used that can be retrieved using the wizard and based on the existing XDTO factory. `XDTOSerializer` object operation is the as with global procedures and XML functions.
For example, serialization of the link to `Products` catalog into XML file can be performed using the program code.

Example:

```
// Retrieve the link to Products catalog item
CatalogItemReference = Catalogs.Products.FindByCode("0000001");

// Create XDTO serializer for global XDTO factory
NewXDTOSerializer = New XDTOSerializer(XDTOFactory);

// Create XML record object and open file
NewXMLWriter = New WriteXML;
NewXMLWriter.OpenFile("D:/Exchange.xml");

// ...

// Serialize the link into XML
NewXDTOSerializer.WriteXML(NewXMLWriter, CatalogItemReference,
XMLTypeAssignment.Explicit);
```

Below is the example of serialization of `Products` catalog item link from XML file.

Example:

```
// Create XDTO serializer for global XDTO factory
NewXDTOSerializer = New XDTOSerializer(XDTOFactory);

// Read the XDTO object data from XML file
New XMLReader = New ReadXML;
New XMLReader.OpenFile("D:/Exchange.xml");

…

// Serialize link from XML
NewLinkToCatalog = NewXDTOSerializer.ReadXML(New XMLReader);
```

16.1.6. Recommendation on XML schema appearance

In general, the `XMLSchemaSet` → `XDTOFactory` → `XMLSchemaSet` conversion doesn't result in a XML schema set that is equivalent to the source set. However, following recommendations on the XML schema appearance will ensure equivalence of the source set and the resulting set. Particularly:

- To convert `XMLSchemaSet` → `XDTOFactory` → `XMLSchemaSet` the resulting schemas set will be similar to the source set;
- Successful verification of `XDTODataObject` object properties filling validity (`Verify()` method) ensures that presentation of the object in XML will correspond to XML schema;
- The maximum flexibility and no distortion when using the polymorphism is ensured.

The XML schema based on the XDTO factory with types, for which the values of the default parameters responsible for XML presentation of data, of course, corresponds to the recommendation above.

In general, recommendations below are the set of rules that ensure the best result from different perspectives.

The XML schema must not contain the anonymous types

Constructions of the following type are not allowed:

```
<element name="Person">
  <complexType>
    <sequence>
      <element name="FirstName" type="string" />
      <element name="FamilyName" type="string" />
    </sequence>
  </complexType>
</element>
```

This fragment must look as follows:

```
<element name="Person" type="tns:PersonType">
  <complexType name="PersonType">
    <sequence>
      <element name="FirstName" type="string" />
      <element name="FamilyName" type="string" />
    </sequence>
  </complexType>
</element>
```

Only the sequence model must be used for complex type content

To model the complex types content (`complexType`), use the single `sequence` block without predefined values for `minOccurs` and `maxOccurs` attributes by default.

```
<complexType name="PersonType">
  <sequence>
    <element name="FirstName" type="string" />
    <element name="FamilyName" type="string" />
  </sequence>
</complexType>
```

The `all` model doesn't breach the equivalence of the source and resulting schemas during `XMLSchemaSet` → `XDTOFactory` → `XMLSchemaSet` conversion . However, there are restrictions. In particular, the `maxOccurs` attribute value for items in `all` model can't be larger than 1.

The `choice` model breaches the equivalence of the source and resulting schemas and doesn't allow the `Verify()` method to verify, whether the object data filling correspond to the XML schema.

For the same reasons, you shouldn't use a combination of several content models or set `minOccurs` and `maxOccurs` attributes of `sequence` to custom values within the single complex type.

It's recommended to present object properties as elements in XML

Presenting properties as XML attributes doesn't affect the equivalence of the source and resulting schemas. However, there are some restrictions:

- The values of object type can't be presented in the attribute - only the value types.
- Properties with multiple values can't be presented in the attribute.
- In case of polymorphous types, the value type of property presented as the XML attribute can be corrupted. Precisely: when assigning value to the property, the value type is aligned with the property type, because the xsi:type attribute, which allows to specify the property value type, can only be specified for the property value in the XML value.

As a result, you don't have to use `simpleContent` for `complexType`, this value saving model implies using attribute properties and the text that includes XML element.
It's not recommended to use the same names for attribute and element declaration..

Don't use mixed content model

Using construction similar to the one below in XML schema means that in the XML element that corresponds to the described type the text can be mixed with the XML elements.

```
<complexType name="FormLetter" mixed="true">

</complexType>
```

To support such information content of the `XDTOObjectType` object, the values of `Sequential` and `Mixed` properties are `True`, and XDTO sequence (`XDTOSequence` object) appears for each instance of this `XDTODataObject` object. Managing information content of such objects is much more complex then content of objects with the state presented by the set of properties only.
Luckily, in most cases, there's no need to use `mixed content`.
The `ElementFormDefault` property of the XML schema must be set to `Qualified`.
This recommendation is the part of good style and XDTO maintains this style.

16.1.7. Rules for XDTO factory verification

The general scheme for coding XDTO factory verification error message identifiers is:

```
xdto <area> <section>[ <rule>]: <error description>
```

- <area> - verification area (XDTO factory, XDTO package, XDTO value type, XDTO facet, XDTO object type);

- <section> - number of section where rules verificatoin failed;
- <rule> - section rule;
- <error description> - description of the error.

16.1.7.1. XDTO factory verification rules

When verifying the XDTO factory, errors are coded with `model` prefix. The common prefix will look as follows:

```
xdto model <section>[ <rule>]: <error description>
```

Packages in the model must have the unique namespace URI - duplicating the packages in the model is prohibited.
Import directive must define not-empty URI of the imported package namespace.
The package import directives must define the existing package types.
Packages of types defined in the models must comply with the package verification rules.

16.1.7.2. Rules for XDTO package verification

When verifying the XDTO package, errors are coded with `package` prefix. The common prefix will look as follows:

```
xdto package <section>[ <rule>]: <error description>
```

The XDTO package must have the `NamespaceURI` property set.
The types defined in the XDTO package may have links to the types specified in the list of the imported (`Dependencies` property).
Import directives must comply with the following rules:

- import directives must define XDTO packages that can't include the import directives pointing at this XDTO package - cycling the imported directives is prohibited;
- import directive must define the filled `NamespaceURI` property of the imported XDTO package;
- import directives must define the existing XDTO package types.

Package properties must comply with the following rules:

- property names must be specified and must not be empty;
- package property names must be unique within the package;
- package property type must be set or defined;
- package property types must be set in the package or its dependencies;
- setting the global property name and anonymous definition of global property type is prohibited;
- package property must not refer to the definition of another package property;
- package property must not define the range of property values number;

- the only possible presentation forms for the package property are `Attribute` and `Item`.

16.1.7.3. Rules for verifying XDTO value type

When verifying the XDTO value type, errors are coded with `valueType` prefix. The common prefix will look as follows:

```
xdto valueType <section>[ <rule>]: <error description>
```

- General rules for verifying the XDTO value type:
- if the type is defined within the types package, the following conditions must be met:
- the XDTO value type must have the `Name` property set to some name;
- the XDTO value type name must be unique within the XDTO package (among all types of XDTO package);
- if the type is defined within another value type definition or within the object type property definition, the following condition must be met:
- the `Name` property of the type definition must not be set;
- the XDTO value type must not contain links to itself neither in base type nor in the list item type or any member type within the hierarchy.
- Rules for base type verification - `BaseType` property:
  - if the `BaseType` property is not set:
    - if the `Option` property is not set and `MemberTypes` property is not set or `Option` property is set to `Atomic`:
    - if the `TypeDefinition` contains the single value, then this type definition is the anonymous base type definition;
  - otherwise, the `anySimpleType` type of the XML schema namespace (http://www.w3.org/2001/XMLSchema) is the base type;
  - if the `BaseType` type definition property is set the following conditions must be met:
    - the base type must comply to the second rule of the XDTO package verification;
    - base type must be XDTO value type;
  - base type can't be this XDTO value type.
- Rules for list item type verification - `ListItemType` property:
  - if the `ListItemType` type definition property is set, the following conditions must be met:
    - the list item type must comply to the second rule of the XDTO package verification;
    - the list item type must be XDTO value type;
    - list item type can't be this XDTO value type;

        - XDTO value type, which is the list item, must be either atomic or combination of XDTO value types consisting only of atomic XDTO value types;
    - of the `Option` property is not set, and `BaseType` property is set or `Option` property is set and has `List` value:
        - if the `TypeDefinition` property contains the single value, then this type definition is the definition of anonymous list item type;
        - otherwise the value of `ListItemType` property is defined from the properties of basic type of XDTO value.
- Rules for member type verification - `MemberTypes` property:
    - if the `MemberTypes` of the XDTO type definition is set the following conditions must be met:
        - member type must comply to the second rule of the XDTO package verification;
        - member type must be XDTO value type;
        - member type can't be this XDTO value type;
        - XDTO value type that is a member, must be atomic or it must be a list;
    - of the `Option` property is not set, and `BaseType` property is set or `Option` property is set and has `Member` value:
        - `TypeDefinition` property contains values that are anonymous definitions of member types;
        - otherwise the value of `MemberTypes` is defined by the value of `MemberTypes` property of the XDTO value base type.
- XDTO value type inheritance is considered correct if the following conditions are met:
    - for atomic types of XDTO values (after verifying rules for list item type and member type, `ListItemType` and `MemberTypes` properties are not set) the following conditions must be met:
        - the type of XDTO value that is a base type must be atomic, i.e. `ListItemType` and `MemberTypes` properties of the basic type of XDTO value shouldn't be set;
        - parent of XDTO value type must be one of the primitive types of XML schema namespace (http://www.w3.org/2001/XMLSchema);
        - the content of facets set in the XDTO value type description must correspond to the list of acceptable facets for the primitive type that is the parent of this XDTO type;

- the value of each facet set in the XDTO value type must comply with the rules of limiting the effective value of the same XDTO basic type facet;
- for list item types (after verifying rules for list item type and member type the `ListItemType` property is set):
  - XDTO basic type can't be the member type, i.e `MemberTypes` property of the XDTO basic type must not be set;
  - if the `ListItemType` property of the XDTO basic type is set, the value of this property must define the basic type for list item type defined in the `ListItemType` property of this XDTO value type;
- if the basic type is `anySimpleType` type of XML schema namespace (http://www.w3.org/2001/XMLSchema), then only one `Space` facet is acceptable in the facet list;
- otherwise, only facets can be defined: `Pattern`, `Enumeration`, `Length`, `MinLength`, `MaxLength` and `Space`;
- the value of each facet set in the type must comply with the rules of limiting the effective value of the same basic type facet;

- for member types (after verifying rules for list item type and member type the `MemberTypes` property is set):
  - the base type can be the `anySimpleType` type of the XML schema namespace (http://www.w3.org/2001/XMLSchema) or member type, i.e. the `MemberTypes` property must be set for the base type;
  - if the member type is the base type, the following conditions must be met:
    - number of member type of the base type must not be larger than the number of member types of this type;
    - member types must be descendants of the corresponding basic type member types in order of member types list;
    - if the basic type is `anySimpleType` type of the XML schema namespace (http://www.w3.org/2001/XMLSchema), the facet definition is prohibited;
  - otherwise, only `Patter` and `Enumeration` facets can be defined;

- the value of each facet set in the type must comply with the rules of limiting the effective value of the same basic type facet.
- If `Option` property (content model) of the type definition is set, it must not interfere with the type definition:

- if the property is set to `Atomic` the type has atomic content model and must comply with the xdto-valueType-5.1 rules;
- if the property is set to `List` the type has `List` content model and must comply with the xdto-valueType-5.2 rules;
- if the property is set to `Member` the type has `Member` content model and must comply with the xdto-valueType-5.3 rules.

16.1.7.4. Rules for verifying XDTO object type

When verifying the XDTO object type, errors are coded with `objectType` prefix. The common prefix will look as follows:

```
xdto objectType <section>[ <rule>]: <error description>
```

- General rules for verifying the XDTO object type:
  - if the type is defined within the types package, the following conditions must be met:
    - the XDTO object type must have the `Name` property set to some name;
    - the XDTO object type name must be unique within the package (among all types of package);
  - if the type is defined within the object type property, the following conditions must be met:
    - the `Name` property of the type definition must not be set.
- Rules for base type verification - `BaseType` property:
  - if `BaseType` property is not set, the `anyType` type of the XML schema namespace (http://www.w3.org/2001/XMLSchema) is considered the base type;
  - if the `BaseType` XDTO object property is set the following conditions must be met:
    - the base type must comply to the second rule of the XDTO package verification;
    - base type must be XDTO object type;
    - base type can't be this XDTO object type.
- Each property of the XDTO object must comply with the following rules:
  - property name must be defined;
  - property name must not be emtpy;
  - property name must be unique for XDTO object type;
  - if the `Type` property is set the following conditions must be met:
    - type name must define the existing XDTO object type or XDTO value type;

    - property type must comply to the second rule of the XDTO package verification;
- property can't contain definition of the anonymous type;
- if the `Type` property is not set the following conditions must be met:
    - if the property has the definition of anonymous type, the property type is the type that corresponds to this definition;
    - otherwise the `anyType` property of the XML schema namespace (http://www.w3.org/2001/XMLSchema) is considered the property type;;
- if the default value is defined for the property, the following conditions must be met:
    - type of `DefaultValue` property must be the XDTO value type;
    - lexical presentation of the default value must correspond to the XDTO property type value space;
- meet the following requirements:
    - if the `LocalName` property of the XDTO property is not set, then the `Name` property of the XDTO property is used as the local name of the XML presentation of the XDTO property;
    - if the `NamespaceURI` property of the XDTO property is not set, the URI of XML namespace of the XDTO property presentation is defined in the following way:
        - if the form of XML presentation of XDTO property (`Form` property) is `Item`, then the namespace URI of the type of this property is used;
        - otherwise, the namespace URI of XDTO property XML presentation is considered absent;
- XDTO property must be unique by XML presentation within the XDTO object type;
- if the XDTO property has the form of `Text` XML presentation, the following rules must be met:
    - name and URI of the namespace must be unset or empty;
    - XDTO object type properties may include XDTO properties with `Attribute` XML presentation form;
    - if the XDTO property has an `Item` XML presentation form, the presence of XDTO property with the `Text` XML presentation form among the XDTO object type properties is prohibited;
- the lower threshold value of the number of `LowerBoundary` value appearance can take the values of non-negative numbers. The `LowerBoundary` value of the lower threshold must be less or equal

to the `UpperRange` threshold value of property value appearance on condition that this value is not -1;
  - the upper threshold value of the number of `UpperRange` value can take the values of non-negative numbers or -1. If this value is -1, it means unlimited number of property values;
  - if the `Fixed` property value is set, then `DefaultValue` must be set by default. It must must comply with the XDTO property type value space;
  - if the link is set to the specific global property, the following rules must be met:
    - property definition can't redefine global definition property values;
    - global property with the link set in this property definition, must be defined withing the package or dependence packages.
- If among the XDTO object type properties there is an XDTO property that corresponds by name or XML presentation with the base type property, the inheritance by limitation is applied. The following conditions must be met with this type if inheritance:
  - the following conditions must be met for each XDTO property:
    - XDTO property with the same name - redefined property - must be defined in the base type;
    - if the base type defines property order (`Ordered` property), the position of the redefined property must be identical to the position in the descendant type;
    - XML presentation form of the redefined property and property if this type must correspond;
    - local name of XML presentation of the redefined property and properties of this type must correspond;
    - Namespace URI of XML presentation of the redefined property and properties of this type must correspond;
  - if the redefined property defines the fixed value, then:
    - presence of fixed value can't be canceled in the descendant type;
    - fixed value in base type and descendant type must correspond;
    - lower threshold of property values number must be less or equal to the lower threshold of redefined property values number;
    - upper threshold of property values number must be more or equal to the upper threshold of redefined property values number;
    - property type must be descendant of the redefined property type;
    - if the base type doesn't have the mixed content (`Mixed` property), the mixed model can't be set in the descendant type;
    - if the base type property order is fixed (`Ordered` property), the order can't be changed in the descendant type;

    - if the base type doesn't define sequence presence (`Sequential` property), the presence of sequences can't be set in the descendant type;
    - if the base type doesn't define the open content mode (`Open` property), the open content model can't be set in the descendant type;
- Otherwise the inheritance type is inheritance by extension. The following rules must be met for this type of inheritance:
    - if the content model of the base type is mixed (`Mixed` property), it can't be changed in the descendant type;
    - if the base type property order is not fixed (`Ordered` property), the order can't be changed in the descendant type;
    - if the base type defines sequence presence (`Sequential` property), the sequence can't be prohibited in the descendant type;
    - if the base type defines the open content mode (`Open` property), the content model can't be changed in the descendant type;
- When you use any type of inheritance, the following conditions must be met:
    - if the content model is mixed (`Mixed` property), the presence of sequence (`Sequential` property) can't be denied;
    - if the content model is open (`Open` property), the presence of sequence (`Sequential` property) can't be denied.

16.1.7.5. Facet limitation rules

When verifying the XDTO facet, errors are coded with `facet` prefix. The common prefix will look as follows:

```
xdto facet <section>[ <rule>]: <error description>
```

Rules for `Length` facet:

- facet value must correspond to the effective value of the base type facet;
- if the `MinLength` facet value is set, it must meet the following conditions:
    - the `MinLength` facet value must be equal or less than `Length` facet value;
    - the value of `Length` facet is not set in the base type or `MinLength` facet value is the same as effective value of `MinLength` base type facet;
- if the `MaxLength` facet value is set, it must meet the following conditions:
    - the `MaxLength` facet value must be equal or less than `Length` facet value;

- the value of `Length` facet is not set in the base type or `MaxLength` facet value is the same as effective value of `MaxLength` base type facet.

Rules for `MinLength` facet:

- the facet value must be equal or more than the effective base class facet value;
- the facet value must be equal or less than the effective `MaxLength` value;

Rules for `MaxLength` facet:

- the facet value must be equal or less than the effective base class facet value;
- the facet value must be equal or more than the effective `MinLength` value;

Rules for `Space` facet:

- if the effective value of the base type facet is collapse, the facet value can't take another value;
- if the effective value of the base type facet is preserve, the facet value can't take replace value.

Rules for `MinInclusive` facet:

- facet value must be less than the `MaxExclusive` facet effective value of this type;
- the facet value must be equal or more than the effective base type facet value;
- if the `MaxInclusive` facet value is set, it must be less or equal to the effective value of the `MaxInclusive` base type facet.
- if the `MinExclusive` facet value is set, it must be larger than the effective value of the `MinExclusive` base type facet.
- if the `MaxExclusive`, facet value is set, it must be less than the effective value of the `MaxExclusive` base type facet.

Rules for `MinExclusive` facet:

- setting the `MinExclusive` and `MinInclusive` facets in the type definition is prohibited;
- if the `MaxInclusive` facet value is set, the value of `MinExclusive` facet must be less than the `MaxInclusive` facet value of this type;
- the facet value must be equal or more than the effective base type facet value;
- if the `MaxInclusive` facet value is set, it must be less or equal to the effective value of the `MaxInclusive` base type facet;
- if the `MinInclusive` facet value is set, it must be more or equal to the effective value of the `MinInclusive` base type facet;

- if the `MaxExclusive`, facet value is set, it must be less than the effective value of the `MinInclusive` base type facet.

Rules for `MaxInclusive` facet:

- the facet value must be more or equal to the `MinInclusive` facet value of this type;
- if the `MaxExclusive`, facet value is set, it must be less than the effective value of the base type;
- if the `MinInclusive` facet value is set, it must be more or equal to the effective value of the base type;
- if the `MinExclusive` facet value is set, it must be larger than the effective value of the base type.

Rules for `MaxExclusive` facet:

- setting the `MaxExclusive` and `MaxInclusive` facets in the type definition is prohibited;
- if the `MinExclusive` facet value is set the `MaxExclusive` facet value must be less than the `MinExclusive` facet value of this type;
- the facet value must be equal or less than the effective base type facet value;
- if the `MaxInclusive` facet value is set, it must be less or equal to the effective value of the base type.
- if the `MinExclusive` facet value is set, it must be larger than the effective value of the base type;
- if the `MinInclusive` facet value is set, it must be larger than the effective value of the base type.

A rule for `TotalDigits` facet:

- the facet value must be equal or less than the effective base type facet value;

Rules for `FractionDigits` facet:

- the facet value must be equal or less than the effective `TotalDigits` value;
- the facet value must be equal or less than the effective base type facet value;

16.2. JSON operations

16.2.1. General information

The JSON format is the universal way of presentation during data exchange (RFC 7159, https://tools.ietf.org/html/rfc7159). The data in JSON format (when encoded):

- **Object** - unordered set of `key:exception` pairs enclosed in braces (`{}`). The `key:value` pair are separated by commas (`,`).
- **Array** - a set of values. The array is enclosed in square brackets (`[]`). The values are separated by commas (`,`).
- **Value** - can be a string, a number, an object, an array or `true`, `false` or `null` literal.
  - **String** - a set of characters enclosed in double quotation marks (`""`).
  - **Number** - serializes with the dot as a separator (`.`). The number accuracy is unlimited.

Thus, the above elements can be used to desribe objects of any complexity for presentation in JSON format. For example, some code in the 1C:Enterprise language generates some data set (in the structure):

```
Data = New Structure;
Data.Insert("LastName", "Ivanov");
Data.Insert("Name", "Ivan");
Data.Insert("Patronimic", "Ivanovich");
Data.Insert("Birthday", BirthDate);

Phones = New Array;
Phones.Add("+7 987 123 45 67");
Phones.Add("+7 986 987 65 43");

Data.Insert("Phones", Phones);
```

By serializing this object in JSON, the following document can be retrieved:

```
{
  "LastName": "Ivanov",
  "Name": "Ivan",
  "Partonimic": "Ivanovich",
  "BirthDate": "2009 02 15T00:00:00Z",
  "Phones": [
    "+7 987 123 45 67",
    "+7 986 987 65 43"
  ]
}
```

JSON doesn't standardize the date presentation format. This is why date presentation in JSON document is defined by preferences of application developer who generates the document and requirements of the system, which will process the JSON document. 1C:Enterprise supports several date presentation formats (set using `JSONDateFormat` system enumeration):

1. The ISO format (`JSONDateFormat.ISO` value). In this case the date is seralized in the following way: "2009-02-15T00:00:00+0400".
2. The JavaScript format (`JSONDateFormat.JavaScript` value). In this case the date is seralized in the following way: "new Date(1234656000000)".
3. The Microsoft format (`JSONDateFormat.Microsoft` value). In this case the date is seralized in the following way: "\/Date(1234656000000)\/" or "/Date(1234656000000)/" (depending on the character screening mode).

There are several option for writing the date (example: May 10, 2014, 13:14:15, UTC+4):

- as a local date: 2014-05-10T13:14:15.
- as a local date with shifting: 2014-05-10T13:14:15+04:00.
- as a UTC date: 2014-05-10T09:14:15Z.

To manage this, the `JSONDateWritingVariant` system enumeration is used. The date in UTC option can be written in any format (ISO, JavaScript and Microsoft). Other options of data presentation are possible only when the serialization is performed in ISO format.

When writing JSON document there are ways to manage the generated data: set the line break, date serialization format, "/" character processing and so on. This settings can be changed using `JSONSerializerSettings` and `JSONWriterSettings` objects.

There are several methods for data operation in JSON:

- Object method - allow to format simple and small JSON documents for sharing with external applications or web services. The JSON document structure is automatically generated by 1C:Enterprise during document

writing. The special option of this method is an ability to serialize in JSON format for XDTO objects in 1C:Enterprise.

- Flow method - allows to operate data of large volumes without uploading it to application memory. The navigation in JSON document is completely on the application developer (both for writing and for reading).
- Mixed method - allows to combine the flexibility of flow method and simplicity of object method.

16.2.2. Object method

16.2.2.1. General information

1C:Enterprise supports serialization of the following data in JSON format:

- `String` - serializaes in string;
- `Number` - serializes in number;
- `Boolean` - serializes in `true` and `false` literals;
- `Undefined` - serializes in `null`
- `Array`, `FixedArray` - serializes in JSON array in case any array item can be serialized in JSON.
- `Structure`, `FixedStructure` - serializes in JSON object:
  - Key - structure item key.
  - Value - structure item value in case the value can be serialized in JSON.
- `Map`, `FixedMap` - serializes in JSON object:
  - Key - map item key. Key can only be the value of `String` type. Otherwise, the exception will be generated.
  - Value - map item value in case the value can be serialized in JSON.
- `Date` - serialization format is defined by settings.
- If there is an attempts to serialize a type that's not from this list - the exception will be called.

When using the object method, you can read (and write) data in the map or structure. The main difference between these objects is that the structure item key complies with the rules for generating variable in 1C:Enterprise language, while there can be any map item key. Considering the fact that JSON doesn't limit the key value, not all JSON documents can be written into the structure. Another difference between the structure and the map is that the structure items can be called "via the dot" and the map items can't be accessed this way. That's why sometimes it's more convenient to retrieve data as a structure, if the keys from JSON document meet the requirements for the keys of 1C:Enterprise structures.

The object method is a simple method of data operation, however, the memory consumption levels are high because the whole JSON document is processed in RAM.

16.2.2.2. Record

To write a record in JSON format, use (in the simplest case) the following objects:

1. The written object, for example the object of `Structure` type.
2. The object that enable low-level data writing in JSON format - `JSONWriter`.
3. `JSONSerializerSettings` serialization settings object.

The `WriteJSON()` global context method operates the objects above. Let's look at the example of structure writing with three items of different types (primitive types):

```
Record = New JSONWriter;
Write.OpenFile("c:\temp\simpleWrite.json");
Data = New Structure;
Data.Insert("RecordLength", 20);
Data.Insert("RecordKey", "abcdefgh");
Data.Insert("ChangeDate", CurrentDate());
WriteJSON(Record, Data, New JSONSerializerSettings);
Write.Close();
```

As a result, this example will generate the following JSON document:

```
{
"RecordLength": 20,
"RecordKey": "abcdefgh",
"ChangeDate": "2014 09 24T17:32:11"
}
```

If the `Date`-type data are included in the serialized data, the serialization format management might be required. To enable this, set the `JSONSerializerSettings` object parameters. In the next example the date will be serialized in `JavaScript` format:

```
SerializationSettings = New JSONSerializerSettings;
SerializationSettings.DateSerializationFormat =
JSONDateFormat.JavaScript;
SerializationSettings.DateWritingVariant =
JSONDateWritingVariant.UniversalDate;
Record = New JSONWriter;
Write.OpenFile("c:\temp\simpleWrite.json");
Data = New Structure;
Data.Insert("RecordLength", 20);
Data.Insert("RecordKey", "abcdefgh");
Data.Insert("ChangeDate", CurrentDate());
WriteJSON(Record, Данные, SerializationSettings);
Write.Close();
```

As a result, this example will generate the following JSON document:

```
{
"RecordLength": 20,
"RecordKey": "abcdefgh",
"ChangeDate": "new Date(1411565832000)"
}
```

If we expand the set of data written, for example, add `UUID`-type value, the writing will not be performed. Result of code execution:

```
Record = New JSONWriter;
Write.OpenFile("c:\temp\simpleWrite.json");
Data = New Structure;
Data.Insert("RecordLength", 20);
Data.Insert("RecordKey", "abcdefgh");
Data.Insert("ChangeDate", CurrentDate());
Data.Insert("UUID", New UniqueID);
WriteJSON(Record, Data, New JSONSerializerSettings);
Write.Close();
```

The exception will be called:

```
Error when calling context method (WriteJSON)
WriteJSON(Record, Data, New JSONSerializerSettings);
reason:
Unacceptable value type sent
```

The reason of such behavior is that the `UUID` type is not included in JSON serialized data type of 1C:Enterprise. However, the system offers ways to "bypass" this: pass the name of callback function, which will deal with JSON serialization of unsupported object, to the `WriteJSON()` function. This function will be called **convertion function**. The format of such serialization will be developed by the developer. Note that such serialization is not universal, i.e. the receiving side that doesn't know the serialization format will not be able to reat the data sent. In other words, the serialization format must be developed by all parties in the exchange of this sort of data.

Considering the above statements, the more complex exchange option will look as follows:

- The function for serialization of the object into JSON format is called (`WriteJSON()`).
- 1C:Enterprise will call conversion function for each structure item with the value type not serialized in JSON format.
- The conversion function analyzes the object passed and makes decision - deny writing or return the value that can be serialized in JSON to the platform.

The updated writing code will look as follows:

```
&OnServer
Procedure WriteAtServer()

  Record = New JSONWriter;
  Write.OpenFile("c:\temp\compositeWrite.json");
  Data = New Structure;
  Data.Insert("RecordLength", 20);
  Data.Insert("RecordKey", "abcdefgh");
  Data.Insert("ChangeDate", CurrentDate());
  Data.Insert("UUID", New UniqueID);
  WriteJSON(Writing, Data, New JSONSerializerSettings,
"RecordConversionFunction", ThisObject);
  Write.Close();

EndProcedure

&OnServer
Function RecordConversionFunction(Property, Value,
AdditionalParameters, Denial) Export
  If TypeOf(Value) = Type("UUID") Then
    Return String(Value);
  EndIf;
  Rejection = True;
EndFunction
```

Note that the conversion function must be declared with `Export` keyword. Also remember that conversion function (in managed form module) can only be described in "context" part of the module, i.e. using `&OnClient` and `&OnServer` compilation directives.

As a result of the example execution, the following JSON document will be generated:

```
{
"RecordLength": 20,
"RecordKey": "abcdefgh",
"ChangeDate": "2014 09 24T18:09:13",
"UID": "5a80e5dc 252b 416f b487 d9ddcebc523e"
}
```

When creating a conversion function, it is necessary to remember the following features:

- The conversion function is called for all types of values, that don't support serialization to JSON.
- If the conversion function return value, which can't be serialized to JSON - the exception is called.
- If the passed value is a structure, the conversion function calling for item of this structure will be called until the property name of this item is written to the stream. As a result, the conversion function may deny writing the value and the structure of the generated JSON document will not be corrupted.
- The property name will be passed to the conversion function only for structure items and maps.

16.2.2.3. Reading

16.2.2.3.1. Overall scheme

Data reading in object method is the same as writing. Let's look at the reading of files, which were generated when we reviewed object writing (previous section).

```
Reading = New JSONReader;
Reading.OpenFile("c:\temp\simpleWrite.json");
Data = ReadJSON(Reading, False, "ChangeDate",
JSONDateFormat.JavaScript);
Reading.Close();
```

The document will have the following conten (accurate to date value):

```
{
"RecordLength": 20,
"RecordKey": "abcdefgh",
"ChangeDate": "new Date(1411565832000)"
}
```

When reading the JSON document to the `Data` variable, the following structure will be generated:

```
Key = RecordLength, value = 20
Key = RecordKey, value = abcdefgh
Key = ChangeDate, value = <date and time value>
```

This reading option fits when the read data can be converted into the structure or map and all the read data can be unambiguously deserialized without loosing the information on the type. If the rad data has complex structure or require additional conversions when reading, there are two ways:

1. Retrieve the map (or structure), to which the JSON document will be uploaded in full and then complete the conversion by bypassing the resulting object.
2. Perform the required conversion directly during data uploading. To do this, sent name of callback function, which will deserialize JSON data into the required 1C:Enterprise objects, to the `ReadJSON()` function. This function will be called **restoration function**.

In both cases the application developer must know, which data are in JSON document and their type. Both data uploading options are described below. The sample task is getting the information on weather in Moscow using some online service. The online service returns the following JSON document as a response:

```
{
  "id":524901,"name":"Moscow","dt":1411464940000,
  "coord": { "lon":37.62,"lat":55.75 },
  "sys": {
"country":"RU","sunrise":1411442400000,"sunset":1411485660000 },
  "weather": [
    { "main":"Clouds","description":"cloudy" }
  ],
  "main": { "temp":282.93,"pressure":1014,"humidity":93 },
  "wind": { "speed":4,"deg":350 },
  "clouds": { "all":90 }
}
```

A brief description of format:

- `id` - city identifier;
- `name` - city name;
- `dt` - date and time of getting a weather, in Unix, GMT format;
- `coord` - city location:
  - `lon` - longitude;
  - `lan` - latitude.
- `sys` - additional information:
  - `country` - country, where the city is located;
  - `sunrise` - sunrise time in Unix, GMT format;
  - `sunset` - sunset time in Unix, GMT format.
- `weather` - additional weather information:

    - `main` - general weather characteristic;
    - `description` - weather description.
- `main` - weather description itself:
    - `temp` - temperature, Kelvins degree. To get the value in Celsius degree, extract 273.15;
    - `pressure` - pressure in hectopascals. To convert to millimeters of mercury, multiply the pressure value by 0.75.
    - `humidity` - humidity in %.
- `wind` - wind parameters:
    - `speed` - speed in miles per hour. To convert to kilometers per hour, multiply by 1.61.
    - `deg` - wind direction in degrees.
- `clouds` - information on precipitation:
    - `all` - probability of precipitation in %.

Uploading this data will result in the structure, where all time is presented with the standard `Date` type, the temperature - in Celsius degrees, speed - in kilometers per hour, and pressure - in millimeters of mercury.
Let's look at both ways to upload this information. The data are written in c:\temp\weather.json file.

16.2.2.3.2. Reading with post processing

The reading process itself is simple:

```
Reading = New JSONReader;
Read.OpenFile("c:\temp\weather.json");
Data = ReadJSON(Reading, False);
Reading.Close();
```

If not consider conversion requirement, everything is expected. However, the date and time did't convert automatically. We can try to point the systen at the fact that `dt` field (for example) is the field with date and time:

```
Reading = New JSONReader;
Read.OpenFile("c:\temp\weather.json");
Data = ReadJSON(Read, False, "dt", JSONDateFormat.JavaScript);
Reading.Close();
```

This will result in the following error message:

```
Error when calling context method (ReadJSON)
  Data = ReadJSON(Read, False, "dt", JSONDateFormat.JavaScript);
reason:
Data presenation has wrong format
```

The reason for this error is that 1C:Enterprise doesn't get the date and time format expressed as a prime number. The program interface for JSON operations offer the function that helps to convert the `Date` - `ReadJSONDate()` fields. To use this function, align deserialized string with the JavaScript format, for example. The example will result in the following program text:

```
Data.dt = ReadJSONDate("new Date("+Format(Data.dt, "NG=")+")",
JSONDateFormat.JavaScript);
```

As a result, the value of `Data.dt` property will be equal to `23.09.2014 13:35:40` (`Date` type). The rest of conversion is performed in similar way.

16.2.2.3.3. Reading with restoration function

The option with post-processing doesn't look well, if the JSON document needs to be read in different places within an application. In this case, the situation might happen, where the conversion code will be located in several places throughout the application.
The good replacement of post-processing in this case is the restoration function. To apply restoration function, describe the function and modify the document reading for a little bit:

```
&OnServer
Function ReadingRestorationFunction(Property, Value,
AdditionalParameters) Export

  If Property = "dt" Or Property = "sunrise" Or Property = "sunset"
Then
    Return ReadJSONDate("new Date("+Format(Value, "ЧГ=")+")",
JSONDateFormat.JavaScript);
  EndIf;
  If Property = "pressure" Then
    Return Value*0.75;
  EndIf;
  If Property = "temp" Then
    Return Value 273.15;
  EndIf;
  If Property = "speed" Then
    Return Value*1.61;
  EndIf;

EndFunction

&OnServer
Procedure ReadAtServer()

  Reading = New JSONReader;
  Read.OpenFile("c:\temp\weather.json");
  Data = ReadJSON(Read, False, , , "ReadingRestorationFunction",
ThisObject);
  Reading.Close();

EndProcedure
```

The example above has one aspect - restoration function will be called for all properties identified in JSON document. This is not always convenient and reduces the JSON document reading performance (because of calling restoration function). For example, when reading weather data we need to perform a special conversion only for the properties containing date and time. We are not going to convert other properties. To avoid analyzing the attribute name in the restoration function (as in the example above), you can do the following: explicitly send the list of attributes, which contain date and time to the JSON document reading function and write restoration function considering that this function will only be called for certain properties. The result is the following code:

```
&OnServer
Procedure ReadAtServer()

  Reading = New JSONReader;
  Read.OpenFile("c:\temp\weather.json");
  Attributes = New Array;
  Attributes.Add("dt");
  Attributes.Add("sunrise");
  Attributes.Add("sunset");
  Data = ReadJSON(Read, False, , , "ReadingRestorationFunction",
ThisObject, , Attributes);
  Reading.Close();

EndProcedure

&OnServer
Function ReadingRestorationFunction(Property, Value,
AdditionalParameters) Export

  Return ReadJSONDate("new Date("+Format(Value, "ЧГ=")+")",
JSONDateFormat.JavaScript);

EndFunction
```

Note that the restoration function must be declared with `Export` keyword. Also remember that restoration function (in managed form module) can only be described in "context" part of the module, i.e. using `&OnClient` and `&OnServer` compilation directives. When developing the restoration function consider that document properties are read in an order that's different from the file.

Let's look at the sequence, where JSON document properties get into the restoration function. To do this, we will place each file property in the table in the order of reading:

JSON document data	**Call sequence of restoration function**
{ "id":524901,	1
"name":"Moscow",	2
"dt":1411464940000,	3
"coord":	6 (structure)
{ "lon":37.62,	4
"lat":55.75 },	5
"sys":	10 (structure)
{ "country":"RU",	7
"sunrise":1411442400000,	8
"sunset":1411485660000 },	9
"weather":	14 (array)

JSON document data	Call sequence of restoration function
[	13 (structure)
{ "main":"Clouds",	11
"description":"cloudy" }	12
],	
"main":	18 (structure)
{ "temp":282.93,	15
"pressure":1014,	16
"humidity":93 },	17
"wind":	21 (structure)
{ "speed":4,	19
"deg":350 },	20
"clouds":	23 (structure)
{ "all":90 }	22
	24 (structure)

In general, the following bypassing rule can be formulated: the first property to be read is the one with no subordinate properties. For example, there are no properties subordinate to `id` property and it will be read first. However, the `coord` property has `lon` and `lat` subordinate properties, so these properties will be read first, and only then - the `coord` property, whichwill get the structure (or map) from the subordinate properties of the document, will be read.

16.2.3. Operating XDTO objects

16.2.3.1. General information

Operations with XDTO objects are usully focused on information exchange between systems, written on the 1C:Enterprise platform. However, the mechanism itself aplies no restrictions for its usage and sharing with other systems.
JSON serialization of XDTO object is performed right into the JSON document without generating the full structure of serialized objects in memory. You should also consider that JSON serializaton "emulates" XML serialization and because of this the resulting JSON document looks similar to XML document.
The JSON document may include any 1C:Enterprise objects, for which it's stated that they can be serialized in XDTO. When attempting to serialize values of the unsupported type, the exception will be called.

16.2.3.2. Record

To write a XDTO object in JSON format, use (in the simplest case) the following objects:

1. The written object itself, supporting conversion to/from XDTO, for example, catalog item.
2. XDTO object serializer- `XDTOSerializer`;
3. The object that enable low-level data writing in JSON format - `JSONWriter`.
4. `JSONSerializerSettings` serialization settings object.

The serialization itself is performed using `WriteJSON()` method of `XDTOSerializer` object. Let's review the example of `CatalogObject`-type data serialization. The `Currencies` catalog will be used as an example. In contains the following fields: `Rate` (`Number` type) and `RateDate` (`Date` type):

```
Data = Catalogs.Currencies.FindByCode("978").GetObject();
Record = New JSONWriter;
Write.OpenFile("c:\temp\simpleXDTOWrite.json");
XDTOSerializer.WriteJSON(Writing, Data);
Write.Close();
```

As a result, this example will generate the following JSON document:

```
{
"#value": {
"Ref": "5b65bd8e ea70 11e4 af93 e03f49b16069",
"DeletionMark": false,
"Code": "978",
"Description": "Euro",
"Rate": 54.659,
"RateDate": "2015 04 25T00:00:00"
}
}
```

The `Date`-type values are serialized in ISO format (identified by XDTO mechanism). The serialization is not managed when the data is being written. The conversion functions are also not supported during serialization, as opposed to stream method and object method.

Also remember about the following aspect: when the object is written its type is not generated, so after JSON serialization of the XDTO object you can't deserialize without specifying the type of object read. The previous example of `Currencies` catalog item serialization will be impossible to deserialize without specifying the explicit value type. To make it simpler, you can use `XMLTypeAssignment` parameter of `WriteJSON()` method of `XDTOSerializer` object. If you set `XMLTypeAssignment.Explicit` as

this parameter value, you will be able to deserialize without explicit type specification and the generated file will look as follows:

```
{
"#type": "jcfg:CatalogObject.Currencies",
"#value": {
"Ref": "5b65bd8e ea70 11e4 af93 e03f49b16069",
"DeletionMark": false,
"Code": "978",
"Description": "Euro",
"Rate": 54.659,
"RateDate": "2015 04 25T00:00:00"
}
}
```

In this example, pay attention to `#type` item, which described the current item type.

General principles of JSON serialization of XDTO objects are the same as with XML serialization:

- The data structure corresponds to the XML document structure.
- Ther are insignificatn differences associated with the aspect of type storage and presentation of arrays in JSON:
  - Order and content of properties are defined in XDTO model and can't be changed.
  - If the type of the written value is defined unambiguously from the type, defined in the XDTO model property, then only the value without type identification is written.
  - If the type of the written value is defined ambiguously from the type, defined in the XDTO model property (for example, complex type value), the value will be written as a complex JSON object with dedicated properties for type (`#type` property) and value (`#value` property). The next example includes writing equivalent options of `String`-type value presentation. The first record is used in ambiguous definition of the type and the second - in unambiguous definition:

```
The first record
{
"#type": "jxs:string",
"#value": "String with example"
}

The second record
"String with example"
```

JSON serialization of XDTO objects has some aspects (as opposed to XML serialization):

- Collection of values (for example, value of `Array`, `Structure` and other types) are always written as a JSON array.
- The attribute value type is written in a dedicated JSON property named `#type`.
- The type namespace in a dedicated JSON property named `#ns`:
  - For namespaces integrated into the platform, the types are written without `#ns` property but with a prefix:

```
jxc:ChartOfCharacteristicTypesObject.DemoChartOfCharacteristicTypes
jxs:string
jxs:decimal
```

  - If there is a schema without prefix, the schema name is written unambiguously using `#ns` property:

```
{
"#ns": "http://mycompany.com/data/enterprise/myscheme",
"#type": "SpecialObject",
"#value": …
}
```

The `Undefined` and `NULL` values are serialized in a special way:

1. `Undefined`:

```
{
"#type": "jv8:Null",
"#value": ""
}
```

2. `NULL`:

```
{
"AttributeName": null
}
```

The JSON allows to serialize XDTO objects, which don't correspond to any schema. In this case the explicit description of attribute types is used. The properties of the same name will not be connected into the array, but they will be included into the JSON document by element in accordance with the XDTO object.

Example:

```
{
  "#type": "jxs:anyType",
  "#value": {
    "Filters": {
      "Filter": {
        "name": "Recorder",
        "Value": {
          "#type": "jxs:anySimpleType",
          "#value": "acc2d259 c8f3 11e2 b5da 5404a6a68c42"
        }
      }
    },
    "Record": {
      "Recorder": {
        "#type": "jxs:anySimpleType",
        "#value": "acc2d259 c8f3 11e2 b5da 5404a6a68c42"
      },
      "Period": {
        "#type": "jxs:anySimpleType",
        "#value": "2012 09 08T00:00:00"
      },
      "Active": {
        "#type": "jxs:anySimpleType",
        "#value": "true"
      },
    },
    "Record": {
      "Recorder": {
        "#type": "jxs:anySimpleType",
        "#value": "acc2d259 c8f3 11e2 b5da 5404a6a68c42"
      },
      "Period": {
        "#type": "jxs:anySimpleType",
        "#value": "2012 09 08T00:00:00"
      },
      "Active": {
        "#type": "jxs:anySimpleType",
        "#value": "true"
      },
    },
  }
}
```

16.2.3.3. Reading

In general, reading XDTO object from JSON document is similar to writing. The reading is performed using the mechanism for reading XDTO objects from XML file, so there are the following restrictions:

- Only the objects with XDTO serialization can be read.
- Properties in JSON document must follow the order of XDTO object.

- When the read object doesn't comply with the schema - the exception is called.
- You can read arbitrary JSON document into XDTO object (`XDTO-DataObject`) using XDTO factory (`XDTOFactory`). Such reading is possible if:
  - the XDTO factory used for reading "knows" about types in JSON document, from which the data is read.
  - all JSON items are specified without explicitly specifying types and elements that are JSON document specific and generated during XDTO objects serialization.

  Reading a JSON document - if it has types that are unknown to the XDTO factory used to read the document - is impossible.

Let's look at the example of reading a JSON document retrieved when operating the `Currencies` catalog from the previous section.

```
Reading = New JSONReader;
Reading.OpenFile("c:\temp\simpleXDTOWrite.json");
Data = XDTOSerializer.ReadJSON(Reading);
Reading.Close();
```

As a result of this example execution, the `Data` variable will include an object of `CatalogObject.Currencies` type for the currency with `978` code. However, such behavior will only be seen if the value of `XMLTypeAssignment` parameter was set to `Explicit` when performing JSON serialization. Otherwise, in an attempt to read the data (as stated above) the exception will be called. When reading an object with implicit specification of object type, the read type can be passed as a `ReadJSON()` method parameter. In this case the example will look as follows:

```
Reading = New JSONReader;
Reading.OpenFile("c:\temp\simpleXDTOWrite.json");
Data = XDTOSerializer.ReadJSON(Read,
Type("CatalogObject.Currencies"));
Reading.Close();
```

When exchanging data between applications using XDTO objects you might have a need to preprocess some data from exchange package. For example, change any value of some type to the fixed value of the same type. To simplify such preprocessing, there is a restoration function. The restoration function name is passed into the reading function from JSON document for serializer of XDTO factory. When reading the object, the restoration function will be called when serializing value of the following types:

- `Boolean`;
- `Number`;

- String;
- Date;
- UUID;
- BinaryData;
- ValueStorage;
- Enumeration values;
- System enumeration values;
- Links to database objects.

The restoration function is not called for Undefined values.

When developing the restoration function, remember, that in this function you can replace a value read from JSON document, but you can't change the type of object read. The restoration function will also not be called for system properties, which are created and used by 1C:Enterprise, for example, #type, #ns and so on.

16.2.4. Srteams operation method

16.2.4.1. General information

Stream method for document operation is designed to partially upload the document in the memory and process it sequentially, item by item. For example, if you need to read some object from JSON document, the srteam method can bring a significant benefit, especially if the required item is located at the beginning of the processed list.

16.2.4.2. Record

To write the JSON document in stream, you need the data written and JSONWriter object. Keep in mind that the generation of the correct JSON document structure depends on the application developer, who generates the document. To simplify such control, the JSONWriter object has the ValidateStructure property.

Let's look at the simple example of document writing:

```
JSONParameters = New JSONWriterSettings(JSONLineBreak.Auto, " ",
True);
Record = New JSONWriter;
Write.ValidateStructure = True;
Write.OpenFile("c:\temp\streamWrite.json", , , JSONParameters);
Write.WriteStartObject();
Write.WritePropertyName("RecordLength");
Write.WriteValue(20);
Write.WritePropertyName("Product");
Write.WriteStartObject();
Write.WritePropertyName("Code");
Write.WriteValue("0020");
Write.WritePropertyName("Name");
Write.WriteValue("Product");
Write.WriteEndObject();
Write.WritePropertyName("NumbersArray");
Write.WriteStartArray();
Write.WriteValue(3.141592654, True);
Write.WriteValue(2.718281828, False);
Write.WriteEndArray();
Write.WriteEndObject();
Write.Close();
```

As a result of this code execution the following document will be generated:

```
{
 "RecordLength": 20,
 "Product": {
  "Code": "0020",
  "Name": "Product"
 },
 "NumbersArray": [
  3.141592654E0,
  2.718281828
 ]
}
```

Such document format is convenient for visual viewing but take up more space. You can change the value of the first parameter of the `JSONWriterSettings` wizard to `JSONLineBreak.No` and the resulting document will look as follows (the difference is about 20%):

```
{"RecordLength":20,"Product":{"Code":"0020","Name":"Product"},"Numbers
Array":[3.141592654E0,2.718281828]}
```

If you need to retrieve the JSON document without file generation, use `SetString()` method of the `JSONWriter` object. After calling this method, to retrieve the string with the generated JSON document, you only need to complete the document writing with `Close()` method of the `JSONWriter` object:

```
JSONParameters = New JSONWriterSettings(JSONLineBreak.No, " ", True);
Record = New JSONWriter;
Write.SetString(JSONParameters);
…
Document = Write.Close();
```

Now this document (from `Document` variable) can be passed to, for example, the HTTP request body.

16.2.4.3. Reading

Stream reading of JSON document is similar to writing: application developer reads the next item in cycle, and decides what is read and processes the data read. The primitive case of stream reading may look as follows:

```
Reading = New JSONReader;
Reading.OpenFile("c:\temp\streamWrite.json");
While Reading.Read() Do
  Message("Current item type " + Reading.CurrentValueType);
  IfReading.CurrentValueType = JSONValueType.PropertyName Then
    Message("Name = " + Reading.CurrentValue);
  EndIf;
  If Reading.CurrentValueType = JSONValueType.Boolean Or
     Reading.CurrentValueType = JSONValueType.String Or
     Reading.CurrentValueType = JSONValueType.Number Or
     Reading.CurrentValueType = JSONValueType.Comment Then
    Message("Value = " + Reading.CurrentValue);
  EndIf;
EndDo;
Reading.Close();
```

Then after reading document generated when we looked at stream writing of JSON document, the following result will be generated:

```
Current item type Object start
Current item type Property name
Name = RecordLength
Current item type Number
Value = 20
Current item type Property name
Name = Product
Current item type Object start
Current item type Property name
Name = Code
Current item type String
Value = 0020
Current item type Property name
Name = Name
Current item type String
Value = Product
Current item type Object end
Current item type Property name
Name = NumbersArray
Current item type Array start
Current item type Number
Value = 3,141592654
Current item type Number
Value = 2,718281828
Current item type Array end
Current item type Object end
```

16.2.5. Combining methods

To simplify operations with JSON, you can combine methods when generating one document. For example, you need to generate a document that includes some structure set and an array. In this case you can generate the whole document with the stream method, and write generated structures and an array with the object method. It's important to ensure the correct document structure before writing with object method.

Let's look at the example of methods combination:

- This system has to return list of orders as a response to external system request.
- The list must contain generation date and a set of orders (that correspond to some criteria).
- Each order is described by the following parameters:
  1. Order number;
  2. Order generation date;
  3. Contractor of the order;
  4. The unique order ID, which can be used to retrieve all order information.

A code fragment in 1C:Enterprise language, which generates JSON document will look as follows:

```
SerializationSettings = New JSONSerializerSettings;
SerializationSettings.DateWritingVariant =
JSONDateWritingVariant.UniversalDate;
SerializationSettings.DateSerializationFormat = JSONDateFormat.ISO;
SerializationSettings.SerializeArraysAsObjects = True;
JSONParameters = New JSONWriterSettings(JSONLineBreak.Auto, " ",
True);
Record = New JSONWriter;
Write.ValidateStructure = True;
Write.OpenFile("c:\temp\combinedWrite.json", , , JSONParameters);
Write.WriteStartObject();
Write.WritePropertyName("GenerationDate");
Write.WriteValue(WriteJSONDate(WhenGenerated, JSONDateFormat.ISO,
JSONDateWritingVariant.UniversalDate));
Write.WritePropertyName("Orders");
Orders = New Array;
For each Order From OrderList Do
  OrderDescription = New Structure("Link, Number, Date, Contractor");
  OrderDescription.Link = String(Order.Link);
  OrderDescription.Number = Order.Number;
  OrderDescription.Date = Order.Date;
  OrderDescription.Contractor = String(Order.Contractor);
  Orders.Add(OrderDescription);
EndDo;
WriteJSON(Write, Orders, SerializationSettings);
Write.WriteEndObject();
Write.Close();
```

When executing such code, it's supposed that:

- The `WhenGenerated` variable contains value of the `Date` type. Contains date and time of JSON document generation.
- The `OrderList` variable is an array of links to order documents.

Executing this code leads to generation of the next JSON document:

```
{
  "GenerationDate": "2014 10 06T12:57:35Z",
  "Orders": {
    "0": {
    "Link": "f4d1495a 02b5 4d56 92c6 840c11dfb592",
    "Number": 234,
    "Date": "2014 09 30T20:00:00Z",
    "Contractor": "Ivanov I.I."
    },
  "1": {
    "Link": "ee821799 2d57 475e a330 f414e53b8bda",
    "Number": 436,
    "Date": "2014 09 24T20:00:00Z",
    "Contractor": "Petrov A.P."
    },
  "2": {
    "Link": "e058a5a8 3c0d 453b 8b1c 963a35fe2b7a",
    "Number": 118,
    "Date": "2014 08 31T20:00:00Z",
    "Contractor": "Ivanov I.I."
    }
  }
}
```


By changing parameter values of `SerializationSettings` and `JSON-Parameters` objects and by manipulating parameters of `WriteJSONDate()` method you can change the resulting JSON document for maximum compliance with the "expectations" of the accepting system.

16.3. Binary data operations

16.3.1. General information

When implementing application, you might need to analyze various binary data. For example, you need to identify the file type by signature or manipulate the picture. To operate the binary data, 1C:Enterprise offers dedicated program interfaces. The binary data operation abilities are described below.

The storage of binary data during system operation is ensured by the object of the same name - `BinaryData`. This object can be used to read data of any file irregardless of this file type. However, the data modification is not supported for this object. The only exception is an ability to split binary data in multiple parts of the specified size (`SplitBinaryData()` method) or vice versa (`ConcatBinaryData()` method).

All further binary data operations are based on the concept of **stream**. **Stream** - is a logical summarization of arbitrary (in general) data source (`Stream` object). The 1C:Enterprise doesn't support creation of an independent `Stream` object not associated with some source. However, there are arbitrary objects that can be created - it's a stream connected with the file on the disk (`FileStream`) or a stream created in memory (`MemoryStream` object). Streams offer data reading and writing abilities. To verify the ability to perform operations, the stream (and derived objects) has special methods that allow to verify that operations are available for this stream (`CanWrite()`, `CanRead()`, `CanSeek()` methods).

If you need to operate the stream on a higher level, in particular, read/write such data as number (with various bitness) or string there are `DataReader`/`DataWriter` objects. This objects enable more structured approach to binary data in the stream. For example, if you know the format of the file you can read this file quite easy by retrieving the required data from headers (which, as a rule, are presented by number and string types) passing the redundant data blocks and downloading the required ones.

A general scheme of binary data operation can be represented as follows:

1. The stream is retrieved.
2. The `DataReader` or `DataWriter` object is created.
3. The object created on Step 2 is used to perform the required actions.
4. The object created on Step 2 is closed.
5. If no more operations are required the stream - retrieved on Step 1 - is closed.
6. If you need to continue operations with the stream, you can set new position in the stream (if this operation is supported) and continue from Step 2.

It should be noted that steps 1 and 2 can be combined. In other words, 1C:Enterprise provides the ability to create `DataReader`/`DataWriter` objects directly from, for example, `BinaryData` object.
To perform various operations with binary data, 1C:Enterprise provides an ability to retrieve a portion of the stream as a dedicated fragment with arbitrary (by byte) access (`BinaryDataBuffer` object). The buffer size is set during creation and can't be changed later. When using the binary data buffer, you can operate numbers with different bitness as a whole. You can also set the order of bytes using the words: "young-old" (little endian) or "old-young" (big endian). You can also split one buffer combine several binary data buffers into one resulting buffer.
It's important to note that binary data buffer significantly simplifies implementation in case the binary data operation is performed on the client application side in asynchronous mode. In this case the data will be read by asynchronous operation and the buffer data will be operated using the synchronous operation.
The binary data can be operated on the client application side (including the web client) and on the 1C:Enterprise server side as well as in synchronous and asynchronous operation mode. Examples below will demonstrate synchronous operation scheme.

16.3.2. Reading binary data

As an example of binary data reading we will review the task of defining the specific size of the file chosen in the system for further use. The .wav file with audio data will be used as a verified file. To store .wav files the Resource Interchange File Format (RIFF) is used. It's decription is available at: https://msdn.microsoft.com/en-us/library/windows/desktop/ee415713.aspx. The following data on the format will be used for reading example:

1. The first 4 bytes of the file contain format ID: `RIFF`.
2. The next 4 bytes contain the size of audio data with the little-endian byte order.
3. The next 4 bytes contain the text type of the data used: `WAVE`.

To perform these action, the following code in 1C:Enterprise language will be required:

```
Reading = New DataReader(FileName, , ByteOrder.LittleEndian);
FileFormat = Reading.ReadChars(4);
DataSize = Reading.ReadInt32();
FileType = Reading.ReadChars(4);
If FileFormat <> "RIFF" Then
  Message("This is not RIFF file");
  Return;
EndIf;
If FileType = "WAVE" Then
  Message("This is a WAV fil data,size: " + DataSize + " bytes");
Else
  Message("This is not WAV file");
  Return;
EndIf;
```

Let's consider this example in detail.

At first, the file is opened with the name from `FileName` variable. The file is opened for reading (`FileOpenMode.Open`). The file will only be used for reading (`FileAccess.Read`) and for this the 16 bytes buffer will be used.

Then the stream for data reading is created, which will have the "younger-older" order for data of `Number`-type. Then 4 ASCII characters, 32-bit integer and 4 more ASCII characters are read from the resulting stream. The resulting data are analyzed and the analysis results are use to decide, whether the chosen file is a .wav file or not.

16.3.3. Writing binary data

In the simplest case, the data are written to the file in the following way:

```
Record = New DataWriter(FileName);
For Index=0 По 255 Do
  Write.WriteByte(Index);
EndDo;
Write.Close();
```

This example writes the sequence of bytes from 0 to 255 (0xFF in hexadecimal notation) to the file. This is the simplest way to write.

You can also use the way that is similar to reading, reviewed in the previous example, when the file stream is retrieved and the data are written in this file stream.

16.3.4. Operating binary data buffer

As mentioned above, the binary data buffer is a convenient way to manipulate binary data gragments. Both reading and writing are supported.

As an example, we will review the RIFF file header parsing from data reading example. To build the example, the same information o the file format will be used. Thus, the buffer with the size of file header will need to be read. The header consists of 4 byte fields. A total of 12 bytes needs to be read.

```
Buffer = New BinaryDataBuffer(12);
File = FileStreams.Open(TempFilesDir() + "Windows Logon.wav",
FileOpenMode.Open, FileAccess.Read);
File.Read(Buffer, 0, 12);
Size = Buffer.ReadInt32(4);

StringStream = New MemoryStream(Buffer);
StringStream.GoTo(0, PositionInStream.Beginning);
ReadLine = New DataReader(StringStream);
FileFormat = ReadLine.ReadChars(4, "windows 1251");
ReadLine.Close();

StringStream.GoTo(8, PositionInStream.Beginning);
ReadLine = New DataReader(StringStream);
FileType = ReadLine.ReadChars(4, "windows 1251");
ReadLine.Close();
```

The process of retrieving data to the binary data buffer has nothing special to it. Further operations require a comment. Reading number of any supported bitness is supported from any buffer position. In this example, `Buffer.ReadInt32(4);` means reading 32-bit integer starting from 4 byte of a buffer. Thus, if you need to read several numbers, located in different parts of a buffer, this can be done without direct positioning in this buffer.

Reading the string, however, is not supported by the binary data buffer. This is why, you must use the object that allows this: `DataReader`. The `DataReader` object can't be created based on the binary data buffer. However, the binary data buffer can be used as a basis to create a stream that is a universal intermediary between physical storage location (file, binary data buffer) and high-level object, enabling operations with these data.

When the `DataReader` object is created based on some stream, it starts reading data from the positions currently set in the stream. That's why in the example the position is set in the stream first, and then the `DataReader` object is created and the required characters are read.

16.3.5. Values in binary and hexadecimal notation and logical operations

When operating various binary formats, the so called "magical numbers" - constants or bit masks - are used quite frequently. Such magical number can be converted into decimal numbers, but such conversion increases risk of errors and is less obvious. To simplify operation with such magical number, there are global context functions: `NumberFromHexString()` and `NumberFromBinaryString()`.

Example:

```
OlderBitMask = NumberFromBinaryString("0b10000000");
ParameterSet = NumberFromHexString("0x79E");
```

The system also enables logical operations with binary data buffer. The following procedures are available:

- logical "And" (`WriteBitwiseAnd()` method);
- logical "Or" (`WriteBitwiseOr()` method);
- logical "excluding Or" (`WriteBitwiseExcludingOr()` method);
- logical "and not" (reverse operation to logical "And", `WriteBitwiseAndNot()` method);
- inversion (`Invert()` method).

Using this methods you can perform various logical operations with data, including verification of specific bits setting. These methods operate with binary data buffers. Thus, the bitness of operation is limited by buffer size. Example of these methods usage are described in the next section.

However, logical operations are increasingly frequently required to operate numbers with limited bitness. The platform provides a set of methods that simplifies manipulation of specific bits in numbers with limited bitness:

- functions for bits and masks operation: `CheckBit()`, `CheckByBitMask()`, `SetBit()`.
- bitwise logical operations: `BitwiseAnd()`, `BitwiseOr()`, `BitwiseNot()`, `BitwiseAndNot()`, `BitwiseXor()`.
- shifting operations: `BitwiseShiftRight()`, `BitwiseShiftLeft()`.

When operating the above mentioned functions, remember:

- All operands are considered 32-bit unsigned integers. An attempts to other value as a parameter is considered invalid value of the parameter with exception generation.
- The bit number is an integer from 0 (the youngest bit) to 31 (the oldest bit).
- When shifting, the free digits are filled with 0.

16.3.6. Specifications

Operating string

When using the binary data, remember about aspects of `String`-type data operation. The aspect is that the length of the string returned by the `StrLen()` global context function is calculated in characters. The sizes of data read/written in string writing/reading methods in binary data operation objects (`ReadChars()`, `ReadLine()`, `WriteChars()`, `WriteLine()`) should be specified in characters too. There is no single option for string length conversion to the similar parameter in bytes. Depending on the string content and coding, the ratio will be different. For example, in UTF-8 coding one character can take from one to six bytes. That's why when operating some data structures with strings of variable length, you should realize what are the length units specified.

In case you need to convert strings to byte set and vice versa, use `GetBinaryDataFromString()`/`GetBinaryDataBufferFromString()` methods and opposite methods: `GetStringFromBinaryData()`/`GetStringFromBinaryDataBuffer()`.

These methods allow specifying the string coding and the need to use BOM (Byte Order Mark for Unicode). When converting strings to byte set, the size of resulting string can be retrieved via the standard interface for retrieving resulting object size (`BinaryData.Size()` or `BinaryDataBuffer.Size`).

Thus, the binary data can be retrieved from the string in the following way:

```
Data = GetBinaryDataFromString("String sample for conversion", "UTF
8", False);
StringSizeInBytes = Data.Size();
```

The opposite operation is performed in the following way:

1. The binary data buffer of the required size is retrieved from the reading stream.
2. The resulting binary data buffer is converted into the string.

In 1C:Enterprise language, this operation will look as follows:

```
// ReadingStream is an existing reading stream
Buffer = ReadingStream.ReadIntoBinaryDataBuffer(StringLengthInBytes);
```

StringRead = GetStringFromBinaryDataBuffer(Buffer);

Cooperative usage of objects

It's not recommended to use DataReader/DataWriter objects and stream objects simultaneously. If between two sequential DataReader reading operations or two sequential DataWriter writing operations the position in stream, which the DataReader/DataWriter objects operate, is changed- the exception is generated. The next example shows the correct change of position in the stream when writing data to the stream:

```
Stream = New MemoryStream();
DataWriter = New DataWriter(Stream);
DataWriter.WriteLine("Hello world!");
DataWriter.Close();

Stream.GoTo(0, PositionInStream.Beginning);

DataWriter = New DataWriter(Stream);
DataWriter.WriteLine("Bye!");
DataWriter.Close();
```

The next example results in exception:

```
Stream = New MemoryStream();
DataWriter = New DataWriter(Stream);
DataWriter.WriteLine("Hello world!");
Stream.GoTo(0, PositionInStream.Beginning);

// Exception will be generated in the next string
DataWriter.WriteLine("Bye!");
```

At the same time, there might be situations, when the system behavior is incorrect, but the errors are not generated:

```
Stream = GetStream();
DataReader = New DataReader(Stream);
TestString = DataReader.Read();

OriginalPosition = Stream.CurrentPosition();
DataWriter = New DataWriter(Stream);
DataWriter.WriteLine("Unexpected stream");
DataWriter.Close();

Stream.GoTo(OriginalPosition, PositionInStream.Beginning);

// In general, it's impossible to define, which value will be placed
to the TestString2
TestString2 = DataReader.ReadLine();
```

The reason for the behavior described in this section is that `DataReader`/`DataWriter` objects use their own buffers when operating the stream. As a result the actual position of the stream differs from the logical position, which is generated as a result of performed operations.
Also, simultaneous use of `DataReader` and `DataWriter` objects, which use one stream, is not supported.

Bitwise operations without buffer usage

Verifying and setting digits in data set, which, for example stores a set of Boolean flags, is used quite frequently. If dataset is presented by the unsigned integers with no more than 32 digits in "length", then to verify the section you can use dedicated functions. Below is an example of their (functions) usage. If the operand bitness is more than 32, it's recommended to use the next example in this section.
To verify state of some number bit, two functions are available: `CheckBit()` and `CheckByBitMask()`. The difference is that using the second function you can verify several check boxes in one value.
Using this function (`LogicalAnd()`) can be illustrated with the following example:

```
READ    = NumberFromBinaryString("0b00000001"); // 0 bit
CREATE  = NumberFromBinaryString("0b00000010"); // 1 st bit
CHANGE = NumberFromBinaryString("0b00000100"); // 2 nd bit
DELETE  = NumberFromBinaryString("0b00001000"); // 3 rd bit
AccessFlags = NumberFromBinaryString("0b00001110");

// verifying the bit
CanRead =  CheckBit(AccessFlags, 0);
// verification by mask (several flags)
CanWrite = CheckByBitMask(AccessFlags, CREATE+CHANGE);
```

In this example we review some byte, which contains flags, defining abilities to operate the supposed file. Each flag is defined by its mask (`READ`, `CREATE`,

CHANGE, DELETE variables). In the example above, the AccessFlags value describes the following set of flags:

- Reading is unavailable. 0 bit is cleared.
- Creating is possible. 1st bit is cleared.
- Change is possible. 2nd bit is cleared.
- Deleting is possible. 3rd bit is cleared.

Verification of the specific bit is performed during verification of ability to read by the presented mask. If you need to verify some other digit of AccessFlags, then when calling the CheckBit() function you have tp change the value of the second parameter. However, if you need to verify several digits at once, then instead of CheckBit() function use CheckByBitMask() method. The second parameter of this function sends verifiation mask, which includes 1 values in the verified digits. If all the required digits in the mask also include 1 - the result of this function will be True. In the example, two digits are verified: CREATE and CHANGE. Combining these two values results in the required mask, verified by CheckByBitMask() method.

Simplfying bitwise operations

Verifying and setting digits in data set, which, for example stores a set of Boolean flags, is used quite frequently. If values don't exceed 32 digits in "length", such verifications are described in the previous example of this section. If the more "lengthy" numbers are used (bitness is more than 32), the required functions can be implemented using the 1C:Enterprise language. Note that because 1C:Enterprise types system doesn't imply specialized unsigned types of the limited bitness, all further methods are based on the suggestion that unsigned integers are passed. In spite of what is said above (about the bitness of more than 32 digits) the examples operate 8-bit numbers. This is done to simplify the example. This example also doesn't consider bites order in the buffer. The youngest value is always located in the youngest bite of the generated binary data buffer. This suggestion is true for both axillary method below.

To implements this operation, two auxiliary functions will be required. The first one will convert the passed number into the binary data buffer of the required size, and the second one will perform the opposite operation - convert binary data buffer to the Number-type value.

```
Function SplitNumberByBytes(Val Number, BytesNumber = 0, Filler = 0)
  // split the number sent by bites
  Number = Int(Number);
  ArrayByBytes = New Array;
  While Number > 0 Do
    ArrayByBytes.Add(Number % 256);
    Number = Int(Number / 256);
  EndDo;
  // if the specific number bitness is required, do it here
  If BytesNumber <> 0 Then
    If BytesNumber < ArrayByBytes.Number() Then
      // cut the result off to fit the set size
      While ArrayByBytes.Number() > BytesNumber Do
        ArrayByBytes.Delete(BytesNumber);
      EndDo;
    Else
      // adding older bites from the filler
      Difference = BytesNumber - ArrayByBytes.Number();
      While Difference <> 0 Do
        ArrayByBytes.Add(Filler);
        Difference = Difference - 1;
      EndDo;
    EndIf;
  EndIf;
  BinaryDataBuffer = New BinaryDataBuffer(ArrayByBytes.Number());
  For Counter = 0 To ArrayByBytes.Number() 1 Do
    BinaryDataBuffer[Счетчик] = ArrayByBytes[Счетчик];
  EndDo;
  Return BinaryDataBuffer;
EndFunction

Function CollectNumberFromBytes(BinaryDataBuffer)
  Result = 0;
  Position = 0;
  For each Byte From BinaryDataBuffer Do
    Result = Result + ?(Position = 0, BinaryDataBuffer[Позиция],
BinaryDataBuffer[Позиция] * Pow(256, Position));
    Position = Position + 1;
  EndDo;
  Return Result;
EndFunction
```

Then the operation of the logical "And" for two numbers will look as follows:

```
Function LogicalAnd(Val Operand1, Val Operand2, Val OperandLength = 0)
  Buffer1 = SplitNumberByBytes(Operand1);
  OperandLength = ?(OperandLength = 0, Buffer1.Size, OperandLength);
  Buffer2 = SplitNumberByBytes(Operand2, OperandLength);
  Buffer1.WriteBitwiseAnd(0, Buffer2, OperandLength);
  Result = CollectNumberFromBytes(Buffer1);
  Return Result;
EndFunction
```

In this case, it's supposed that the size of the second operand in operation is always equal to the size of the first operand.

Using this function (`LogicalAnd()`) can be illustrated with the following example:

```
READ    = NumberFromBinaryString("0b00000001");
CREATE  = NumberFromBinaryString("0b00000010");
CHANGE = NumberFromBinaryString("0b00000100");
DELETE  = NumberFromBinaryString("0b00001000");

AccessFlags = NumberFromBinaryString("0b00001110");
CanRead = LogicalAnd(AccessFlags, READ);
CanWrite = LogicalAnd(AccessFlags, CREATE+CHANGE);
FullAccess = LogicalAnd(AccessFlags, CREATE+CHANGE+READ+DELETE);
```

In this example we review some byte, which contains flags, defining abilities to operate the supposed file. Each flag is defined by its mask (`READ`, `CREATE`, `CHANGE`, `DELETE` variables). The flag is easily verified: if the result of `LogicalAnd()` function is not equal to the sum of the verified flags - it means that not all verified flags are set. If the function result equals to the sum of of the verified flags - it means that all verified flags are set. In the simplified case, verification of one flag returns 0 value, if the flag is not set and the flag value - if it's set.

In the example for `AccessFlags` flags above:

- Reading is unavailable. `CanRead` variable value is 0.
- Writing is not possible. Value переменной `CanWrite` равно 6 (сумма флагов `CREATE` и `CHANGE`).
- Full access is not possible. The value of `FullAccess` variable is 15 (sum of all described flags, except the unavailable `READ` flag).

You can perform other functions for bitwise data operation similar to `LogicalAnd()` method.

16.4. PDF files

16.4.1. General information

1C:Enterprise enables you to save spreadsheet documents and representable document batches in PDF format. Moreover, both spreadsheet documents and representable document batches can be saved in PDF format (it is described by ISO 32000-1). So far as spreadsheet documents are concerned, you can as well save them in any of long-term storage formats, i.e. PDF/A-1 (ISO 19005-1), PDF/A-2 (ISO 19005-2) and PDF/A-3 (ISO 19005-3).

When you save a document in PDF/A format, note the following:

- A document is saved in PDF 1.4 format (for PDF/A-1 only).
- Duplex printing is not supported.
- The source of paper in accordance with PDF-document size cannot be chosen during printing.
- No transparency is supported (for PDF/A-1 only).
- The version history of the platform used for the spreadsheet writing and the document creation dates are saved when a document is written.

To save a document in PDF format, it is sufficient to specify correctly the following:

- `TableFileKind` of `SpreadsheetDocument.Write()` method.
- `FileKind` of `DisplayedDocumentsBatch.Write()` method.

In addition to saving in PDF format, the platform as well enables you to manage attachments which may be available in a PDF document. Such a need may arise when you need to create a document containing both human and machine readable sections. By way of example only, let's take a waybill which is saved in PDF and nevertheless can be viewed and printed (to the extent of its human readable section). Moreover, a XML-file is attached thereto which describes content of this waybill and can be read by dedicated software only.

PDF application interface enables you to manage digital signatures available in a document. Moreover, you can verify and apply restrictions to a document, add a further signature, validate existing signatures and place a digital signature representation on the document page. Note that a document in PDF format executed with a digital signature cannot be edited with the said signature remaining valid and in full force and effect. This rules applies both to the document and all attachments.

The existing PDF document handling model fails to support tools enabling users to programmatically modify a human readable section.

16.4.2. Attachments

16.4.2.1. Attachment writing

Use `WritePDF` to write attachments to PDF documents. The said object is available both on the client and on the server side in either asynchronous or synchronous mode. In the documentation, examples where operations are performed in the synchronous mode will be given only (to reduce their scope). You can use attachments in PDF documents saved in 1.5 or any later version.
Generally, attachments are added as follows:

1. Create `WritePDF` on the basis of a file or thread. If this object is created without reference to its source - you need to open a PDF document using `WritePDF.Open()` method.
2. An attachment collection is now generated for a PDF document. To get an attachment collection, use `PDFAttachmentsCollection`. To add attachments, use `PDFAttachmentsCollection.Add()`.
3. An attachment collection is added to an opened document using `WritePDF.WriteAttachments()`.
4. To close a PDF document, use `WritePDF.Close()`.

When you write it, attachments with their names being similar to those available in the collection are replaced. As such, "name" means `PDFAttachment.FileName` property value (`PDFAttachmentsCollection` collection element). If any object is deleted from an attachment collection, it means that it is as well deleted at the file level. 1C:Enterprise cannot compare attachment lists on element-by-element basis (i.e. a list of existing attachments and that of attachments being written).
If you need to delete an attachment from a file - use `WritePDF.WriteDeleteAttachments()`. In this scenario, objects are as well compared by names: each array element represents a value of `PDFAttachment.FileName` property. Note that physically no attachment is deleted. They are just marked as deleted. PDF viewer allows you to return to any attachment so deleted.

16.4.2.2. Attachment reading

Use `ReadPDF` to read attachments to PDF documents. The said object is available both on the client and on the server side in either asynchronous or synchronous mode. In the documentation, examples where operations are performed in the synchronous mode will be given only (to reduce their scope).
Generally, attachments are read as follows:

1. Create `ReadPDF` on the basis of a file or thread. If this object is created without reference to its source - you need to open a PDF document using `ReadPDF.Open()` method.
2. An attachment collection is retrieved from a PDF-document using `ReadPDF.GetAttachments()` method.
3. To close a PDF document, use `ReadPDF.Close()`.

An attachment collection can exist independently from `ReadPDF`. Therefore, if you need to delete any attachment from an existing PDF file, this can be done as follows:

1. Open an existing PDF document: create an object using a dedicated wizard or `ReadPDF.Open()`.
2. Get an attachment using `ReadPDF.GetAttachments()`.
3. Close `ReadPDF` using `ReadPDF.Close()`.
4. Use `PDFAttchmentsCollection` so retrieved to update a PDF document. For that purpose, use `WritePDF`.

16.4.3. Digital signature

16.4.3.1. General information

Whenever deemed necessary, these terms will be adjusted and updated. Specifically, in this section "digital signature" means a signature which consists of the following two parts: digital signature in cryptographic terms (i.e. data sequence generated as a result of cryptographic conversion of source data) and graphic representation of a digital signature.

There are two kinds of digital signatures (so far as PDF file attachments are concerned): approving and certifying signatures. A single document can have several approving digital signatures. if a document is signed with an approving digital signature, you can disable any further editing. In each document there is only one certifying digital signature. However, document editing can be allowed, provided that modifications so made have no impact on its content (for instance, comments can be added).

An approving digital signature can be used as follows:

- A document (for instance, commercial proposal) is created.
- Its author signs the said commercial proposal with an approving signature and thus cetifies its content and his or her authorship
- Further, a commercial proposal is delivered to the company's marketing director who verifies the same and signs it with his or her approving digital signature and moreover locks the said document.
- Therefore, a document is approved by all responsible persons (using digital signatures) and locked from editing.

A certifying digital signature can be used as follows:

- There is a document which needs to be delivered for reviewed and approval through open communication channels.
- An author creats a document and signs it with his or her certifying digital signature, at the same time allowing it to be reviewed and commented, if necessary.
- Further, it is handled by a reviewer who makes his or her comments and redelivers the same to the sender. However, both parties know that the document remains unmodified (it is guaranteed by the said signature).

It should be noted that classification of digital signatures described in this section applies exclusively to and is defined by PDF format. Moreover, legal definition of a digital signature and its classification (see applicable regulations) generally is not in any way associated with the definitions used in this section.

16.4.3.2. Adding a signature

When you add a digital signature, the following mechanisms are used: PDF document handling tools and cryptographic means supported by the platform.

To sign a PDF document, do the following. Create an object describing a digital signature: `Signature = New PDFSignatureDescription();`. Let's see when properties of this object are used.

The following properties are of material importance for each signature:

- `Certificate` - make sure that this property is always filled in with a link to a public key certificate which is used to **verify** a digital signature so generated.
- `SignatureKind` - defines a digital signature to be used to sign a document. Depending on the signature kind, you need to define one of the following properties.
- `LockDocument` - whenever `SignatureKind` is set to `Approving`, use this property (if set to `True`) to restrict any further modification of a document, as soon as it is duly signed. So far as an approving digital signature is concerned, `AllowAccessEdit` makes no sense.
- `AllowAccessEdit` -whenever `SignatureKind` is set to `Certifying`, use this property to specify actions which can be performed by a user when handling a document so signed. Anyway, no action so performed can modify the content of a duly signed document. So far as a certifying digital signature is concerned, `LockDocument` makes no sense.

If graphic representation of a digital signature is used when you execute a document, its location in a document is defined by the following properties:

- `PageNumber` - defines a page of a PDF document where graphic representation of a digital signature is made available. Page numbering starts from 1.
- `Left` and `Top` - these properties define a distance from the left and right page boundaries (accordingly), where graphic representation of a digital signature is made available. The zero point is in the top left corner of a page. The said distance is set in mm.
- `Width` and `Height` - these properties are used to define the size of graphic representation of a digital signature horizontally and vertically (accordingly).

As soon as an object which describes a digital signature is created and filled with data, you can add a digital signature to a PDF document.

To add a digital signature, do the following:

- Create an object for access to cryptographic functionality.
- Get a certificate to verify a digital signature as a separate object (`CryptographicCertificate`).
- Create and fill in `PDFSignatureDescription`.
- Create an object allowing it to add a digital signature to a PDF document (`WritePDF`).
- Sign a PDF document using `WritePDF.WriteSign()`.

Example of signed document:

```
SpreadsheetDocumentAgreement.Write(DocumentName,SpreadsheetDocumentFil
eKind.PDF);
CryptoManager = New CryptoManager("", "", 75);
CryptoManager.PrivateKeyAccesssPassword = Password;
Certificate = GetPublicCertificate();
Stamp = GetLayout ("SignPDF");
SignPDF = New PDFSignatureDescription;
SignPDF.Name = "SignAgreement";
SignPDF.Certificate = Certificate;
SignPDF.SignatureKind = PDFSignatureKind.Approving;
SignPDF.LockDocument = True;
WritePDF = New WritePDF(DocumentName);
WritePDF.WriteSign(CryptoManager, SignPDF, Stamp);
```

16.4.3.3. Signature verification

The following two options are available for a signed PDF document:

- Get a list of all digital signatures used to sign a document for further review and analysis.
- Verify all digital signatures or a single signature only selected by a user.

No digital signature used to sign a document - can be modified.

To get a list of signatures used to sign a document, select `ReadPDF.GetSignaturesDescription()`. The function returns an array of objects which describe a signature.

```
CryptoManager = New CryptoManager("", "", 75);
CryptoManager.PrivateKeyAccesssPassword = Password;
ReadPDF = New ReadPDF(DocumentName);
Signatures = ReadPDF.GetSignaturesDescription(CryptoManager);
```

However, the primary task pursued by a digital signature, - is that you need to be sure that a document is not modified, as soon as you sign it. The following two methods are used for this purpose:

1. Verify all digital signatures used to sign a document. For this purpose, you can use `ReadPDF.VerifySignatures()`.

For examples of signature verification, see below.

```
CryptoManager = New CryptoManager("", "", 75);
CryptoManager.PrivateKeyAccesssPassword = Password;
ReadPDF = New ReadPDF(DocumentName);
Try
  ReadPDF.VerifySignatures(CryptoManager);
Exception
  Notify("Signatures are not valid!");
EndTry;
```

2. Verify a single digital signature used to sign a document. For this purpose, you can use `ReadPDF.VerifySignature()`.

For an example of verification of a single signature only, see below:

```
CryptoManager = New CryptoManager("", "", 75);
CryptoManager.PrivateKeyAccesssPassword = Password;
ReadPDF = New ReadPDF(DocumentName);
Signatures = ReadPDF.GetSignaturesDescription(CryptoManager);
Per each CurrentSignature from Signatures Cycle
  Try
    ReadPDF.VerifySignature(CurrentSignature.Name, CryptoManager);
  Exception
    Notify("Signature" + CurrentSignature.Name +" is not valid!");
  EndTry;
EndDo;
```

Chapter 17. Online services mechanisms

17.1. Web services

17.1.1. General information

Web service mechanism in 1C:Enterprise is designed to support Service-Oriented Architectures (SOA).
Service-oriented architecture is an applied architecture, where all functions are defined as a independent services with the called interfaces. Calling these services in a sequence allows to implement buisness processes.
Service-oriented architecture is the new approach to creation of distributed information systems, where program resources are viewed as services provided via network. This approach ensures fast consolidation of distributed components (services) into a single solution that supports certain business processes.
The web services mechanism allows to use 1C:Enterprise as a set of services in complex distributed and heterogeneous systems. It also allows to integrate it with other enterprise systems using service-oriented architectures.
The 1C:Enterprise configuration can export its functions via web services. Definitions of web services are set in the configuration tree and become available to arbitrary information systems thanks to their publication on a web server.

Besides, 1C:Enterprise can call third party web services via static links defined in the configuration tree or using dynamic links created with the 1C:Enterprise language means.

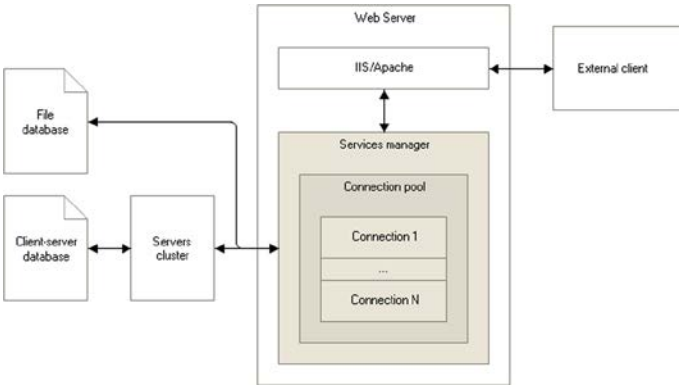


Fig. 280. Web services

Web service clients interact with 1C:Enterprise using web server plug-ins. A web server plug-in operates in the process of the host, which receives/sends messages to/from Service manager. IIS or Apache web server can be used as a service host. A web server plug-in contains a pool of connections and other mechanisms intended to interact with 1C:Enterprise database.

The web services mechanism, implemented in 1C:Enterprise, supports the following standards:

- SOAP 1.1,
- SOAP 1.2,
- WSDL 1.1,
- WS-I Basic Profile 1.1,
- HTTP 1.1,
- TLS 1.x (TLS 1.1 and 1.2 are supported if sending client certificate to the server is not required), including cryptographic algorithms that comply with GOST R 34.10-94, R 34.10-2001, R 34.10-2012, R 34.11-94, R 34.11-2012, and 28147-89;
- MTOM;
- Authentication: Basic, NTLM/Negotiate.

Web services mechanism implemented in 1C:Enterprise doesn't support:

- WS-Policy mechanism;
- WS-Addressing mechanism;
- WS-Security protocol;

- Setting the namespace that is already used by XML as a target namespace (targetNamespace).

To access the Web service use the address that generates as follows: http://host/base/ws/WebServiceName or http://host/base/ws/WebServiceAddress. Let's consider the address components in more detail:

- http://host/base - ordinary URL, for example, used for access to the infobase via a web client. When separators are available, specification of separator values with the command-line parameter Z of the client application startup is not supported.
- ws - the flag of addressing the web service (as opposed to hs, which defines access to HTTP service).
- WebServicename - web service name. Set in the Web service object property.
- WebServiceAddress - describes alternative name for accessing the web service. Set in the Publication file name property of Web service object. Can be changed when publishing web service.
- Calling the web service name is the same as calling its address.

17.1.2. Providing functions via web services

To make the 1C:Enterprise functions available to the external web service consumers, do the following:

- create the required number of web services in configuration,
- publish web services using the dedicated Designer tool.

Creating the web service is:

- adding web service configuration object to the metadata tree,
- desribing operations that the created web service can perform,
- describing parameters of web service operations.

A Web service configuration object has a module, where 1C:Enterprise language procedures are created that are performed when calling web service operations: Web service operation parameter types are described using the XDTO types and can be either XDTO values or XDTO objects.

The web service is called in the following way:

- the appropriate connection with the infobase is chosen from the pool; when there is no appropriate connection, it's established;
- the new session is created and `SessionParametersSetting` event is called for the created session (in session module);

- the requested web service method is called and `SessionParametersSetting()` handler is called (in session module) every time when the non-initiated session parameter is called.

TIP. It's not recommended to perform resource-demanding operations in `SessionParametersSetting` event handler.

The `SessionParametersSetting` event of the session module is called on the server in a privileged mode. The module of the called service is performed on the server in normal mode.
The session module initializes session parameters and performs some set of commands when calling any 1C:Enterprise web service.

17.1.3. Example of web service implementation

For example, you need to create the 1C:Enterprise web service, which has to return the content and tabular section of the invoice using the sent invoice number.
To describe the returned value, let's created the `InvoiceData` XDTO package with http://www.MyCompany.ru/shipment namespace containing three types of XDTO packages:

- `Products` - to send data of `Products` catalog item. This type of XDTO object will contain the following properties:
  - `Name` - `string` type from http://www.w3.org/2001/XMLSchema namespace;
  - `FullDescr` - `string` type from http://www.w3.org/2001/XMLSchema namespace;
  - `Barcode` - `string` type from http://www.w3.org/2001/XMLSchema namespace;
  - `PurchasePrice` - `int` type from http://www.w3.org/2001/XMLSchema namespace.
- `InvoiceRow` - to send the data with one invoice row. This type of XDTO object will contain the following properties:
  - `Products` - `Products` type from http://www.MyCompany.ru/shipment namespace; it's a link to XDTO object defined above;
  - `Quantity` - `int` type from http://www.w3.org/2001/XMLSchema namespace;

    - Price - int type from http://www.w3.org/2001/XMLSchema namespace;
    - Total - int type from http://www.w3.org/2001/XMLSchema namespace.
- Invoice - to send data of all invoice rows. This type of XDTO object will contain the only value:
    - Content - InvoiceRow type from http://www.MyCompany.ru/shipment namespace. It's a link to XDTO object defined above. To ensure that this property may have the unlimited number of values, set its Upper bound property to -1.

After the required XDTO types are created, add new InvoiceData web service with the following properties to the configuration:

- Namespace URI - http://www.MyCompany.ru/shipment;
- XDTO packages - InvoiceData;
- Publication file name - shipment.1cws.

Define the Get operation for the created web service with the following property values:

- Return value type - Invoice from http://www.MyCompany.ru/shipment namespace;
- Possible empty value - set;
- Procedure name - Get.

The DocumentNumber parameter must be defined for the Get operation. It should have the following property values:

- Value type - string type from http://www.w3.org/2001/XMLSchema namespace;
- Transfer direction - Input.

After this, open the module of the created web service and place Get() function, which is performed when calling this web service, to the module.

```
Function Get(DocumentNumber) Export

  // Get invoice object by the sent number
  DocumentRef = Documents.Invoice. FindByNumber(DocumentNumber,
CurrentDate());

  If DocumentRef.Empty() Then

    Return Undefined;

  EndIf;

  Document = DocumentRef.GetObject();

  // Get XDTO object types
  ProductsType = XDTOFactory.Type("http://www.MyCompany.ru/shipment",
"Products");
  InvoiceType = XDTOFactory.Type("http://www.MyCompany.ru/shipment",
"Invoice");
  InvoiceRowType =
XDTOFactory.Type("http://www.MyCompany.ru/shipment", "InvoiceRow");

  // Create XDTO object of the invoice
  Invoice = XDTOFactory.Create(InvoiceType);

  For Each DocumentLine From Document.Content Do

    // Create XDTO object of invoice row
    // and products
    DeliveryNoteString = XDTOFactory.Create(DeliveryNoteStringType);
    Products = XDTOFactory.Create(ProductsType);

    // Fill product properties
    Products.Name = DocumentString.Products.Name;
    Products.FullDescr = DocumentString.Products.FullDescr;
    Products.Barcode = DocumentString.Products.Barcode;
    Products.PurchasePrice = DocumentString.Products.PurchasePrice;

    // Fill properties of the delivery note string
    DeliveryNoteString.Products= Products;
    DeliveryNoteString.Quantity = DocumentLine.Quantity;
    DeliveryNoteString.Price = DocumentLine.Price;
    DeliveryNoteString.Amount = DocumentLine.Amount;

    // Add the delivery note string
    DeliveryNote.Components.Add(DeliveryNoteString);

  EndDo;

  // Return the delivery note
  Return DeliveryNote;

EndFunction
```

At the end you should publish the generated Web service on the web server, for example, on http://www.MyCompany.ru, in the directory shipment.

17.1.4. Operation of other providers' web services

17.1.4.1. General information

1C:Enterprise 8 may use the web services provided by other providers using several methods:

- using static links generated in the configuration tree;
- using dynamic links generated by means of 1C:Enterprise language;
- by combination of previous methods.

The advantage of using static links is faster operation, because description of the provider's web service is retrieved once when the link is generated. In the future, when you call this web service, the existing description of the web service is used.

When you use dynamic links, the description of the provider's web service will be acquired by 1C:Enterprise 8 system every time when the web service is called that will naturally slow down operation of this web service. However, the advantage of this approach is the option to acquire the relevant description of the provider's web service. When static links are used to acquire the relevant description of the web service, you should repeat import of WSDL description by means of the Designer and save the modified configuration.

When applications are operated, the situation may arise when the same web service is displayed at different addresses (URL) but has the identical description (WSDL). At this case you need to download the description of the web service in configuration (to create the object in the configuration object tree) but you should give the specific address of the web service while using it. The combined method allows operation in such manner. As an example we may look at the following situation: we have the replicated web service that performs some function. The application written on 1C:Enterprise 8 uses services of this web service, and the address of the service may be varied (as the service is replicated) and the description - is docked. Then you may download the description of the web service in the application and provide to set up the address of the service's specific copy in settings of the application that (the address) will be used for operation.

The other method is also available: the dynamic link is used but the address of the web service is acquired not from the file of description (WSDL) but is directly specified when the object is generated.

When it is attempted to download the description of the web service in the Designer (the static link is generated) or when the dynamic link is used (using the

object WSDefinitions). If the description of the web service contains the error (according to the "opinion" of 1C:Enterprise 8 system) the description will not be downloaded and the exception will be generated. The text of the exception will contain the detailed diagnostics of reasons for denied downloading. WSDL errors are rated according to their disclosure. Each WSDL error contains the entry with the detailed description of the following type:

```
<Type of the item with the error>.<Name>
  [<Type of the item with the error>.<Name>[…]]
    <error description>
    <Type of the item with the error>
    <Name>
```

In this description:

- Type of the item with the error -type of the WSDL item. It most often matches the name of the tag in the xml file describing the web service.
- Error description - description of the certain error.
- Name - name of the WSDL object where the error has occurred.

The following table gives the error that may occur when the description of the web service is checked and recommendations of how to correct them (errors):

Error description	Correction recommendations
No type is found	Check that name spaces and type and item names are correct
No message is found	Check that name spaces and message names are correct
Binding is not found	Check that name spaces and binding names are correct
No port type is found	Check that name spaces and port type names are correct
No abstract procedure is found	Check that the abstract operation name is correct
No procedure components are found	Specify components in the procedure. It may be procedure input or output
No error description is found	Specify the name of the error description
No abstract operation is set up	Specify the abstract operation
No name is set up	Specify the name of the WSDL object according to the plan: http://schemas.xmlsoap.org/wsdl/2003-02-11.xsd

Error description	Correction recommendations
Procedure input does not match the abstract operation input	The procedure and the abstract operation should either have the input (both) or have not it (both). The abstract operation may not have the input and the procedure none and vice versa.
Procedure output does not match the abstract operation output	The same as the message about the procedure input
No port type is set up	Specify the port type, check name spaces
Object is not unique	Check uniqueness of the object according to the name space and the name
No location is set up	Specify the location R2007 rule - for the item import R2711 rule - for the item port
Name spaces are different	Check and correct differences between import name spaces and the name space of the imported design
No type of the message part is set up	Set up the type for the message part (the type or the item according to the WSDL style), check name spaces
No message is set up	Specify or correct the message name, check the name space for the WSDL object
Abstract operation requires input	Specify input for the abstract operation
Abstract operation requires no input	The abstract operation should not have input
Abstract operation requires output	Specify output for the abstract operation
Abstract operation requires no output	The abstract operation should not have output
Abstract operation error is not set up	Specify or check that the abstract error name and the name space are correct
No binding is set up	The port binding is not set up, the binding name is not correct, the name space is not correct
No abstract operation description is set up	The abstract operation description is not set up, the name or the name space is not correctly specified

Error description	Correction recommendations
The procedure name space should be set up in the rpc-literal style	Specify the item namespace for the procedure. R2717 rule
The procedure name space should not be set up in the document-literal style	Delete the item namespace for the procedure. R2216 rule
The part of the message in the rpc-literal style should refer to definition of the type in the design	To specify the type of the part the relevant object message in parts of messages should use the type, not the item (the tag type, not element). R2203 rule
The part of the message in the document-literal style should refer to the announcement of the item in the design	To specify the type of the part, the relevant object message in parts of messages should use the item, not the type (the tag element, not type). R2204 rule
Unknown binding style	Specify the binding type and check that it is correctly written. Specification: http://www.w3.org/TR/wsdl#_bindings. R2705, R2706 rules
The transport in binding should be set up as http://schemas.xmlsoap.org/soap/http	Check that the transport is exactly http://schemas.xmlsoap.org/soap/http. R2702 rule
No message part is set up	Set up the part of the message for the error description. R2205 rule
In the design the header error should be described as the item announcement	Use the type set up as the item (use the tag element) in the part of the message used for the error description. R2205 rule
The header in the design should be described as the item announcement	Use the type set up as the item (use the tag element) in the part of the message used for the header description. R2205 rule

Error description	Correction recommendations
The design should contain the attribute targetNamespace	You should add the attribute target-Namespaceand fill it. This requirement is related with features of the platform implementation
Error of imported design download	Check that name spaces and the location of the imported design are correct

When you specify the expression of the type Rule RXXXX in the table it means that - you should find the rule RXXXX in the specification of Basic Profile Version 1.1 (http://www.ws-i.org/profiles/basicprofile-1.1-2004-08-24.html, in English).

17.1.4.2. The example of using the static WS reference

As an example of using the third-party provider's web services, let's look at the call to the web service that have been generated earlier in the example.

First of all, add the new configuration object, that is the WS reference to the configuration tree under the name `DeliveryNoteData` referring to the published service. To do it, import the WSDL description of the published service and give http://www.MyCompany.ru/shipment/ws/Shipment.1cws?wsdl as an URL.

After that you may generate the procedure given below, for example, in the goods receipt module. It fills the tabular section of the document with the data of the provider's goods receipt acquired using the provider's web service.

```
The procedure AcquireGoodsReceiptData(ProviderReceiptNumber)

  // generate the WS proxy based on the link
  Proxy = WSReferences.GoodsReceiptData.
CreateWSProxy("http://www.MyCompany.ru/shipment", "GoodsReceiptData",
"GoodsReceiptDataSoap");

  GoodsReceiptData = Proxy.Get();

  If GoodsReceiptData = Indefinite Then

    Return;

  EndIf;

  // Fill the goods receipt with the acquired data
  Fore Each StringReceipt From ReceiptData.Components Loop

    NewRow = DocumentObject.Components.Add();

    NewRow.Quantity = ReceiptString.Quantity;
    NewRow.Price = ReceiptString.Price;
    NewRow.Amount = ReceiptString.Amount;

    // Find the product item according to the passed data
    // (for example, according to a bar code)
    NewRow.Products = Catalogs.Products. FindByAttribute("Barcode",
ReceiptString.Products.Barcode);

  EndDo;

EndProcedure
```

If the address of the service's actual location is different from the address used when the description of the web service is being downloaded into configuration, the new address should be expressly specified when the object `WSProxy` is generated:

```
// generate the WS proxy based on the link
Proxy = WSReferences.GoodsReceiptData.
CreateWSProxy("http://www.MyCompany.ru/shipment", "GoodsReceiptData",
"GoodsReceiptDataSoap", , , , "http://www.realURL/realPath");
```

17.1.4.3. Example of using the dynamic WS reference

Using the dynamic link is different from using the static link only by the method of WS proxy generation and that generation of the WS reference in the configuration tree is not required.

If we compare it in the example given in the previous section, so, unlike generation of the proxy based on the static link, the WS proxy is generated using the constructor as follows when the dynamic link is used:

```
// Generate the WS proxy based on the WS definition
Definition = New
WSDefinitions("http://www.MyCompany.ru/shipment/ws/Shipment.1cws?wsdl"
);

Proxy = New WSProxy (Definition, "http://www.MyCompany.ru/shipment",
"GoodsReceiptData", "GoodsReceiptDataSoap");
```

While the WS proxy based on the static link looks as follows:

```
// generate the WS proxy based on the link
Proxy = WSReferences.GoodsReceiptData.
CreateWSProxy("http://www.MyCompany.ru/shipment", "GoodsReceiptData",
"GoodsReceiptDataSoap");
```

If the address of the service's actual location is different from the address specified in the WSDL file used to generate the definition of web services, the new address should be specified when the object `WSProxy` is generated based on the definition of web services:

```
// Generate the WS proxy based on the WS definition
Definition = New
WSDefinitions("http://www.MyCompany.ru/shipment/ws/Shipment.1cws?wsdl"
);

Proxy = New WSProxy(Definition, "http://www.MyCompany.ru/shipment",
"GoodsReceiptData", "GoodsReceiptDataSoap", , , ,
"http://www.realURL/realPath");
```

17.1.5. Editing properties of the web service

The name, the synonym and the comment of the object are inserted into the bookmark Main.

The subordinate objects Procedures are generated in the bookmark Procedures that in their turn may have subordinate objects Parameters needed to operate objects of this type. Subordinate objects are described in the property palette.

17.1.5.1. Properties "Procedures"

Apart from general properties of configuration objects, the procedure of the web service contains the following properties:

- Return value type - type of value, returned by the Web service operation. It may be either XDTO value type or XDTO object type.

- Possibly empty value - indicates whether the return value can be undefined.
- In transaction - indicates, if the Web service module code will be executed in transaction. If the property is set up, the transaction will automatically start when the web service is called and the transaction will be either docked or retrieved (depending on results of the transaction) when operation of the web service is completed. If the property is not set up, the transaction will not start when the web service module begins execution.
- Method name - is the name of the exported procedure of the web service module that will be completed when this property is called.

The bookmark Subsystems specifies the subsystems objects of this type belong to.

The bookmark Other determines the following properties:

- Namespace URI - contains Web service namespace URI. Each Web service can be uniquely identified by its name and the URI of the namespace which it belongs to. The name space of the service should not coincide with known name spaces that are already used or reserved by other companies. It is recommended to include the fragment unique for the company that develops the web service into the name space of the service. For example, it makes sense to begin all name spaces with the prefix for the company named Promresurs, for example,http://promresurs.com. Then the URI of the web service's name space will look as http://promresurs.com/public/services/OurService.
- XDTO packages - a list of XDTO packages which types can be used as return value types for operations and Web service operation parameter types.
- Publishing file name - name of the web service description file located on the web server.

The editor of the web service module is opened using the button Module.

17.1.5.2. Properties "Parameter"

Apart from general properties of configuration objects, the parameter of the web service procedure contains the following properties:

- Value type - the value type of the Web service operation parameter. It may be either XDTO value type or XDTO object type.
- Possibly empty value - indicates whether the operation parameter value can be undefined.
- Transfer direction - determines the data transfer direction using this parameter. Values:

- Input - means that the parameter is used to transfer the data to the web service;
- Output - means that the parameter is used to get the data from the web service;
- Input-Output - means that the parameter may be used both to transfer the data and to get them from the web service.

17.2. HTTP services

17.2.1. Standard OData interface

17.2.1.1. General information

The standard OData interface in 1C:Enterprise 8 is used to get access to the data of the system from the external application without modifying the code of the application (for example, if the application is set up on support). To get such access, you should specifically publish the application on the web server and specify what configuration objects will be used in such manner.

The 3rd version of the OData protocol (http://www.odata.org/, in English) (http://www.odata.org/documentation/odata-version-3-0/odata-version-3-0-core-protocol, in English) is used to get access to the data. The following data representation formats are supported: **atom-xml** and **json**. The standard OData interface that allows reading the 1C:Enterprise 8 data, modifying them, generating new data objects and to delete existing ones is automatically generated for access to the data during publishing. The application based on 1C:Enterprise 8 may be both the client and the server during operation of the standard OData interface. You don't need to take any additional measures for operation of the server (this option is automatically provided). To become the client of the standard OData interface, you should implement the software layer in the application that will use the data of the server using standard 1C:Enterprise 8 interfaces, for example, the object `HTTPConnection`.

One system may contain both SOAP services (implemented in 1C:Enterprise 8 as web services) and the standard OData interface and the HTTP service.

The standard OData interface may be used to solve the following tasks:

- Integration of the application with various websites (for example, based on Microsoft SharePoint);
- Implementation by third-party means of additional application functionality without modifying its configuration;
- Downloading the data to the application and uploading the data from it;
- Integration of the application with corporate systems, probably without additional programming.

Typical procedures performed using the standard OData interface:

- Acquisition of the list of system objects with preset filter;
- Acquisition of the data of the certain system object;
- Generation of the new system object and recording modifications of the existing one;
- Posting one document, start of the business process.

17.2.1.2. General principles of operation

Specific terms are used for operation of the OData protocol. The following terms are used when the protocol is applied:

- **Entity** - something that has identity. The entity also has the set of properties. Some properties describe its identity which combination determines the key of the entity. This key may be used to get the certain entity.
- **The set of entities** - the collection of entities with the specific type.
- **The complex type** - the set of properties without identity.
- **The function** - the set of some procedures completed on behalf of the server returning the data (they don't need to be the entity or the set of entities) and not resulting into observed side effects (data modifications). The function is always related to the entity or the set of entities.
- **The action** - the function that may modify the data.

Terms of the protocol somehow match terms taken in the 1C:Enterprise 8:

- **The entity** - one of 4 1C:Enterprise 8 object groups may be used as the entity: object types, entry record sets, register records and strings of tabular sections in object types. Register records and strings of tabular object types (as individual entities) are available for read only.
- Attributes of 1C:Enterprise 8 objects are displayed as **properties** of entries. In some cases (for example, the attribute of the configuration object of the complex type), the attribute may be represented by several properties, one of which will be **navigational**. Such property contains the link (URL) to the entity describing the 1C:Enterprise 8 object as a value. The property describing the type of such attribute is called **dispatcher**. The name of such property ends with the suffix _Type. More details of the dispatcher property are described below.
- If the attribute of the object has the reference type, the object that such attribute refers to will be called **the bound entity**.

The standard OData interface is called using the HTTP query, performed at the certain URL. URL is generated in a specific manner and consists of the following sections:

1. The infobase address;
2. The flag of calling the standard OData interface;
3. The name of the resource being called;

4. Parameters of the query for the call of the resource.

The call is made using HTTP query of the certain type. When operation of the standard OData interface is described in more details, the query type will be individually specified.

Let's look at URL sections in more details.

The infobase address

It is the standard URL, for example, used to get access to the infobase using the web client. For example, http://host/base or http://host.server.zone/database. You should also remember that when you use the infobase where separation is set up, you may use only one method to specify values of separators, such as: values of separators may be specified only to the URL of the infobase. Specification of separator using the parameter Z is not supported.

The flag of calling the standard OData interface

The sequence /odata/standard.odata is used as such flag.

The name of the resource being called

The specifically generated resource ID (probably with the parameter) or predetermined resources. For example, $metadata, Catalog_Контрагент(guid'value').

Parameters of the call to the resource

Parameters of the type taken for HTTP queries are used as parameters of the call: ?key=value&key2=value2.

When the resource is called, the specific keywords with the specific designation may be used:

- $format - specifies the format where you should get the data. If the keyword is not specified, the data are acquired in the atom-xml format.
  - $format=atom - returns the data in the format **atom-xml**.
  - $format=json - returns the data in the format **json**. To specify the fact that the data should be returned in the format **json**, you may specify the MIME type application/json in the header Accept of the HTTP query to get the data.
- $metadata - specifies that you need to get the description of the standard OData interface.
- $filter - describes the filter applied to data acquisition.
- $select - describes the list of the entity's properties acquired when the standard OData interface is called.

Once the URL of the needed resource is generated, you should complete the HTTP query of the needed type. Depending on the procedure being performed, the relevant HTTP method is used:

- Data acquisition - method `GET`;
- Object generation - method `POST`;
- Data update:
  - method `PATCH` - in this case you may specify only the properties to be updated;
  - method `PUT` - in this case you should specify all properties of the entity;
- Data deletion - method `DELETE`.

As a result of the query completion, the client application gets the response of the server that may contain various data given by the server as a XML document apart from the status code.

Addressing the standard interface OData, take into account that the following notation shall be observed for the specification of some types:

- Type `UniqueID` (type `Edm.Guid`) is specified as `guid'value'`. Example: `guid'8d04ce3c-6360-11e9-dd92-fa163e8ba11d'`.
- Type `Date` (type `Edm.DateTime`) is specified as `datetime'yyyy-mm-ddThh:mm:ss'`. Example: `datetime'1945-05-09T06:00:00'`.

17.2.1.3. Data Presentation

The data returned by the standard OData interface may be represented as a XML document or a JSON document. It depends on the format of the data to be requested.

The following correlation between the 1C:Enterprise 8 type and the OData type is used for various data types:

The 1C:Enterprise 8 type	The type of the OData property
`String`	`Edm.String`
`Date`	`Edm.DateTime`
`Number` (even)	`Edm.Int16`, `Edm.Int32`, `Edm.Int64`
`Number` (fraction)	`Edm.Double`
`Boolean`	`Edm.Boolean`
`Ref`,`UniquID`	`Edm.Guid`
`Enum`	`Edm.String`
`ValueStorage`	Consists of three properties *****: 1. Navigational property;

The 1C:Enterprise 8 type	The type of the OData property
	2. <Property name>_Base64Data; 3. <Property name>_Type.
Reference type	Two properties: • The value of the link to the object of the type `Edm.Guid`; • Navigational property.
Complex types	Two properties of the type `Edm.String`: • The value of the actual type expressed as a string; • Name of the actual type.

The symbol * in the table means that:

- You may get the data stored in the attribute using the navigational property:
  - The image with the relevant value of theHTTP header content-type will be acquired for the type `Image`-;
  - The byte-oriented stream will be acquired for the type `BinaryData`-;
  - The- XDTO-serialized value of the stored data will be acquired for other types.
- The property <Property name>_Base64Data stores the data that may be acquired using the navigational property, but only encoded in Base64. You may only edit the data of the attribute with the type `ValueStorage` using this property.
- The property <Property name>_Type describes the type of the data stored in the attribute. It may take one of three values:
  - application/octet-stream - binary data;
  - application/xml+xdto - XDTO-serialized object;
  - the value of the HTTP header content-type matching the image stored in the attribute, for example, image/jpeg for the image of the JPEG format.

Other types are not supported and the attempt to read them will result into the error with the code 501.

Names of properties may end with various suffixes. Such properties have specific meaning. Possible suffixes will be described in more details below:

- Key;
- Type;
- Base64Data;
- ____Presentation.

Key

The property with such suffix contains the value of the key in the relevant reference attribute of the object (without such suffix) or the independent information register without dimensions. The name with such suffix is used to set up the filter as the name of the attribute for which the filter is made. For example, to set up the filter for the reference field `Counterparty`, the condition will look as follows: Counterparty_Key=guid'value'. At this the property Counterparty will contain representation of the counterparty with the specified value of the link.

Type

This suffix is used to describe the attribute of the complex type. So, if the data has the field of the complex type `Counterparty`, то в the document that returns the standard OData interface, this field will match two properties:

- Counterparty_Type - will contain the description of the attribute value type as the string (type `Edm.String`, the dispatcher property);
- Counterparty - will contain the value of the attribute (the relevant type).

The list of permitted types that may be used in the field with such suffix is determined by the design of the service that you may acquire once you request the full description of the standard OData interface. Thus, when you need to set up the type `Document.GoodsExpenditure`, the item with the suffix _Type should contain the recorded value StandardODATA.Document_GoodsExpenditure.

If the value of the attribute of the complex type in the 1C:Enterprise 8 infobase has the value `Indefinite`, the dispatcher property will have the value StandardODATA.Undefined, and the value of the property should be ignored.

The example of the representation of the attribute of the complex type:

```
<d:AttributeComplex/>
<d:AttributeComplex_Type>StandardODATA.Undefined</d:AttributeComplex_T
ype>
```

Base64Data

Thus suffix is used to specify the name of the property containing the data located in the attribute of the type `ValueStorage`, as the string `Base64`. So, if the configuration object has the attribute `File` that has the type `ValueStorage`, two properties will match this field in the document that returns the standard OData interface:

- File_Type - contains the name of the data type stored by the attribute;
- File_Base64Data - contains the string `Base64` containing the data.

____Presentation

The property <Property name>____Presentation contains the representation of the attribute (4 symbols "_" are located between the name of the property and Presentation). The value of the field is equal to the value resulting from the operation of the query language function `Presentation()`.
Fields of representations are not included into results of the query with the blank parameter $select and when values * and ** are located in the parameter $select. Thus, if you need to acquire all attributes of the object and to represent some attributes, you should specify the value *, and then individually list representation fields of the requested attributes while dividing them with commas: $select=*, AttributeLink____Presentation.
To acquire representations of all object attributes you should specify the value *____Presentation in the parameter $select: $select=*____Presentation.
You should use the following expression to acquire values of all fields and representations for all fields: $select=*, *____Presentation.
To acquire the representation of the current entity, you may use the option to specify the field Presentation in the list of requested fields: $select=Ref_Key, Presentation. In this case, the value of the link and the representation of this link will be acquired.
Representation fields are not included into the meta data description that is returned in the description of the standard OData interface (параметр $metadata).
Acquisition of the representation is not supported for tabular sections and attributes of the type `ValueStorage`. Modification of values for representation fields is not supported.

17.2.1.4. Rules of resource name generation

When you call any resource, its ID is generated according to the following principle: NamePrefip_ConfigurationObjectName_NameSuffix. Using the standard OData interface, you can get access to the following objects (NamePrefix):

Configuration objects	Prefix name to specify in URL
Catalog	Catalog
Document	Document
Documents journals	DocumentJournal
Constant	Constant
Exchange plan	ExchangePlan
Chart of accounts	ChartOfAccounts
Chart of calculation types	ChartOfCalculationTypes
Chart of characteristic types	ChartOfCharacteristicTypes

Configuration objects	Prefix name to specify in URL
Information register	InformationRegister
Accumulation register	AccumulationRegister
Calculation register	CalculationRegister
Accounting register	AccountingRegister
Business process	BusinessProcess
Task	Task

ConfigurationObjectName - property Name of the configuration object as it is set up during development of the application in the Designer.

NameSuffix - is used to specify the resource name and is not the required part of the name. The following expressions may be used as the name suffix:

- Name of the object's tabular section;
- Name of the tabular section's attribute or the record set;
- Name of the register's virtual table;
- RowType;
- RecordType.

More details of the resource name specification described above will be described in more details:

Name of the object's tabular section

If the object has the tabular section, to get access to all records of this tabular sections, you should add the name of the tabular section after the name of the object, for example, to acquire all strings of tabular sections `Goods` of all documents `GoodsExpenditure` you should complete the GET query at the following address: http:// host/base/odata/standard.odata/Document_GoodsExpence_Goods.

Name of the tabular section's attribute or the record set

If the object has the tabular section, there is the option to specify that you need to get not all attributes of the tabular section but some list of these attributes. For this, you should specify the list of required attributes in the parameter $select as follows: <Tabular section name>/<Field name>. The same option is provided for record sets of registers where RecordSet is used as the name of the tabular section: RecordSet/<Field name>.

Name of the register's virtual table

The function related to the resource returning the set of register entities is used as the register's virtual table. The name of the function is the dame as the English version of the query language's used virtual table name. Parameters of the function match parameters of the virtual table. For example, to get the slice of last

information register ExchangeRates you should complete the GET query at the following address: http://localhost/demo/odata/standard.odata/InformationRegister_ExchangeRates/SliceLast().

RowType

The entity with such suffix describes the type of the tabular section string in any object.

RecordType

The entity with such suffix describes the individual register record.

17.2.1.5. Rules of filter condition generation

17.2.1.5.1. General information

This section gives the information about various method used for generation of the filter of the acquired data used in the standard OData interface of 1C:Enterprise 8.
You may get the description of filter options in the OData protocol in the protocol documents (in English):

- http://www.odata.org/documentation/odata-version-3-0/odata-version-3-0-core-protocol/ раздел 10.2.3.1;
- http://www.odata.org/documentation/odata-version-3-0/odata-version-3-0-core-protocol/ раздел 5.1.2.

17.2.1.5.2. $filter

You may filter the data when you acquire the data. For this the specific language is used that allows to describe the conditions that the data returned by the standard OData interface must comply with. The description of the filter starts with the keyword $filter that is followed by the condition. The following procedures are available:

- Logical procedures:

Description	Name	Example
Equal to	eq	/Catalog_Города?$filter=Description eq 'Main'
Not equal	ne	/Catalog_Cities?$filter=Description ne 'Perm'
Greater than	gt	/Catalog_Goods?$filter=Price gt 10
Greater or equal	ge	/Catalog_Goods?$filter=Price ge 10
Less than	lt	/Catalog_Goods?$filter=Price lt 10

Description	Name	Example
Less or equal	le	/Catalog_Goods?$filter=Price le 10
Logical Or	or	/Catalog_Goods?$filter=Price lt 10 or Price gt 100
Logical And	and	/Catalog_Goods?$filter=Price gt 10 and Price lt 100
Negation	not	/Catalog_Goods?$filter=not (Price eq 10)

- Arithmetic procedures:

Description	Name	Example
Addition	add	/Catalog_Goods?$filter=Price add 5 gt 10
Subtraction	sub	/Catalog_Goods?$filter=Price sub 5 gt 10
Multiplication	mul	/Catalog_Goods?$filter=Price mul 5 gt 1000
Division	div	/Catalog_Goods?$filter=Price div 4 gt 2

- Grouping commands:

Description	Name	Example
Procedural priority	()	/Catalog_Goods?$filter=(Price add 5) gt 10

Example of the filter:

```
http://host/odata/standard.odata/Catalog_Goods?$filter=Name eq 'Milk'
and Price lt 2500
```

When the filter condition is generated you should remember priority of procedures. The following table shows the list of the filter expression language procedure in order of descending priority. Procedures with the same priority are calculated from left to right:

Operator	Description
( )	Increased procedural priority
/	Navigation
-	Arithmetical negation
Not	Logical negation
Mul	Multiplication
Div	Division
Add	Addition
Sub	Subtraction
Gt	Greater than
Ge	Greater or equal
Lt	Less than
Le	Less or equal

Operator	Description
Eq	Equal to
Ne	Not equal
And	Logical "And"
OR	Logical "OR"

When query conditions are generated (parameter filter) or the attribute by which ordering is carried out is generated (parameter orderby) the following functions may apply:

- String functions:

Function	Description	Example
substringof(Str1, Str2)	Returns true if Str1 is the substring Str2.	/Catalog_Goods?$filter=substringof('Red October', Manufacturer) eq true
endswith(Str1, Str2)	Returns true if Str1 ends with Str2.	/Catalog_Goods?$filter=endswith(Manufacturer, 'OOO') eq true
startswith(Str1, Str2)	Returns true if Str1 ends with Str2.	/Catalog_Goods?$filter=startswith(Manufacturer, 'OOO') eq true
substring(Str, Int1) substring(Str, Int1, Int2)	Returns the substring from Str1. In the option with two parameters, the string is returned from the position Int up to the end of the string.	/Catalog_Suppliers?$filter=substring(INN, 1, 2) eq '77'

Function	Description	Example
	In the option with three parameters the substring is returned starting from the position Int1 and with the length of Int2.	
concat(Str1, Str2)	Returns the string resulting from concatenation of Str1 and Str2.	/Catalog_Supplier?$filter=concat(concat(City, ', '), Country) eq 'Moscow, Russia'
like(Str, Template)	Returns true if the value Str1 meets the template Template. The template syntax is similar to the function LIKE() of the query language.	/Catalog_Goods?$filter= like(Name, '[^к]%')

- Functions for operation of dates:

Function	Description	Example
year(DateTime)	Returns the year from the value of the type `Edm.DateTime` or `Edm.DateTimeOffset`.	/Catalog_Goods?$filter=year(Made) eq 2013
quarter(DateTime)	Number of the year quarter where the specified value of the type `Edm.DateTime` is located.	/Catalog_Goods?$filter=quarter(ManufactureDate) eq 1
month(DateTime)	Returns the month from the value of the type `Edm.DateTime` or `Edm.DateTimeOffset`.	/Catalog_Goods?$filter=month(Made) eq 12
day(DateTime)	Returns the day from the value of the type `Edm.DateTime` or `Edm.DateTimeOffset`.	/Catalog_Goods?$filter=day(Made) eq 1
hour(DateTime)	Returns the value of hours from the value of the type `Edm.DateTime` or `Edm.DateTimeOffset`.	/Catalog_Goods?$filter=hour(Made) eq 23
minute(DateTime)	Returns the value of minutes from the value of the type `Edm.DateTime` or `Edm.DateTimeOffset`.	/Catalog_Goods?$filter=minute(Made) eq 59
second(DateTime)	Returns the value of seconds from the value of the type `Edm.DateTime` or `Edm.DateTimeOffset`.	/Catalog_Goods?$filter=second(Made) eq 59
datedifference(DateTime1, DateTime2, Type)	Returns difference of dates DateTime2 and DateTime1 in units specified by the parameter Type: • `second` - seconds; • `minute` - minutes; • `hour` - hours; • `day` - days;	/Catalog_Goods?$filter=datedifference(Made, ValidBefore, 'day') gt 10

Function	Description	Example
	• month - months; • quarter - quarters; • year - years.	
dateadd(DateTime1, Type, Int1)	Returns the date acquired by addition of the value Int1 to the value DateTime1 expressed in units Type: • second - seconds; • minute - minutes; • hour - hours; • day - days; • month - months; • quarter - quarters; • year - years.	/Catalog_Goods?$filter=dateadd(Made, 'month', 1) eq ValidBefore
dayofweek(DateTime)	Returns the day of the week by the value of the typeEdm.DateTime.	/Catalog_Goods?$filter=dayofweek(ProductionDate) eq 7
dayofyear(DateTime)	Returns the day of the year by the valueEdm.DateTime.	/Catalog_Goods?$filter=dayofyear(ProductionDate) eq 1

- Other functions:

Function	Description	Example
round(Number)	Returns the parameter rounded to the closest even number.	/Catalog_Goods?$filter=round(Bec) gt 31

Function	Description	Example
isof(expr, type)	Returns true if the object which the parameter expr specifies has the type that type refers to. The string designating the name of this type is taken as the type of the value, for example: String, Number, Boolean, Date, Catalog_Goods an so on.	/Catalog_Products?$filter=isof(Price, 'Number')
cast(expr, type)	Returns the object that the parameter expr specifies corrected to the type specified by the parameter type. The string designating the name of this type is taken as the type of the value, for example: String, Number, Boolean, Date, Catalog_Goods an so on.	/Catalog_Goods?$filter=cast(ComplexAttribute, 'Number') le 12

NOTE. Procedures of comparison with the attribute of the type ValueStorage are not supported.

There is the option to filter entries using check for equality of the field of the complex type and the link. To do this, you should use the function cast().

Example:

```
$filter=ReceiptDocument eq cast(guid'0d4a79cb 9843 4147 bcd9
80ac3ca2b9c7', 'Document_ReceiptOfGoods')
```

In this example, entity is assigned a combined attribute `ReceiptDocument`. The query will filter all records of the entries where this attribute is filled by the link to the document `GoodsReceipt` with the specified ID (0d4a79cb-9843-4147-bcd9-80ac3ca2b9c7).
If you need to filter by the attribute of the type `UUID` it is done using the simple comparison procedure:

```
$filter=AttrubuteName eq guid'0d4a79cb 9843 4147 bcd9 80ac3ca2b9c7'
```

To filter by the type of any attribute of the complex type, you should use the function isof():

```
$filter=isof(ReceiptDocuments, 'Document_GoodsReceipt')
```

The next expression will always return the blank result of the query, regardless the actual presence of the used ID in the relevant column of entities:

```
$filter=ReceiptDocuments eq guid'0d4a79cb 9843 4147 bcd9 80ac3ca2b9c7'
```

If you need to filter by the collection item, you should use lambda functions. As arguments of the lambda function the name of the lambda variable is used followed by the symbol of colon and the logical expression using the lambda variable to specify properties of the collection item. The system supports use of the following lambda functions:

- any - applied the logical expression to each collection item and returns the value true if at least one collection item meets this condition. The lambda function any returns true without arguments, if the collection is not blank.

```
http://host/base/odata/standard.odata/Document_Sales?$filter=Goods/any
(d: d/Цена gt 10000)
```

In this example, the list of documents `Sales` is generated that have the tabular section `Goods` with the attribute `Price`. At this, `Price` should be more than `10000`. The resultant list will include the document containing at least one string that meets the condition.

- all - applies the logical expression to each collection item and returns the value true, if all collection items meet it.

```
http://host/base/odata/standard.odata/Document_Sales?$filter=Goods/all
(d: d/Price lt 10000)
```

In this example, the list of documents `Sales` is generated that have the tabular section `Goods` with the attribute `Price`. At this `Price` should be

less than 10,000. The resultant list will include the documents where all strings meet the preset condition.

If you need to specify the value of the attribute of other attribute of the complex type with reference values in such manner that both the value of the checked attribute and the type of the complex attribute are checked, you should use the function cast().

For example, there is the document RouteSheetthat contains the attribute of the complex type ShippingReason. This attribute may take values of the type DocumentRef.Consignment and DocumentRef.Internal-Traffic. At this the document Consignment has the attribute ShippingPlace that may take values of the type CatalogRef.Warehouse andCatalogRef.ShippingAddresses.

If you need to filter documents RoutingSheet with the condition for the name of the warehouse specified in the consignment (i.e., by the attribute Name of the object of the type CatalogRef.Warehouse recorded in the attribute ShippingPlace of the document Consignment) the condition should look as follows:

```
cast(cast(ShippingPlace, 'Document_Consignment')/ShippingPlace,
'Catalog_Warehouse')/Name eq 'Main in Moscow'
```

The following standard functions are not supported: lenght, indexof, replace, tolower, toupper, trim, years, days, hours, seconds, floor, ceiling;

17.2.1.5.3. $top

There is the option to limit the number of entries returned when the resource is called. For this the parameter $top is used.

Example:

```
http://host/odata/standard.odata/Catalog_Goods?$filter=Price lt
1000&$top=10
```

17.2.1.5.4. allowedOnly

If you need to get only the data objects that are not under data access restrictions when the query is completed, you should add the parameter allowedOnly in the URL of data retrieval.

Example:

```
http://host/base/odata/standard.odata/Catalog_Goods?$allowedOnly=true
```

If the parameter is not specified or specified with the value false the error with the code 401 (if the result of the query contains the data with the denied access)

may occur when the query is completed. The error may not occur if the additional conditions have been specified that limited the filter by only the authorized data. For example, the data access restriction is set up for the directory `Data` that does not permit to acquire items of the directory that have theattribute `PrivacyValue` equal to 1 but allows to acquire items of the directory that have the attribute `PrivacyValue` equal to 2. In this case the next query will not result into the error because the condition that leaves only the authorized data for selection is expressly applied:

```
http://host/base/odata/standard.odata/Catalog_Data?allowedOnly=false&$
filter=PrivacyValue eq 2
```

The parameter allowedOnly may be used only for GET request to sets of entities.

17.2.1.5.5. $skip

Allows to exclude first few records from the result of the query. If parameters $top and $skip are specified at the same time, the parameter $skip will be applied earlier than the parameter $top. Priority for application of parameters does not depend on the order of their specification in the body of the query.

Example:

```
http://host/base/odata/standard.odata/Catalog_Товары?$skip=2
```

17.2.1.6. Parameters of the query

17.2.1.6.1. $count

This parameter allows to get not selection but its (selection's) size as a result of the query. If successful, the body of the response should contain only the number of collections items formatted as an ordinary number.

Example:

```
// returns the number of items in the directory Goods,
// for which the condition is true: the value of the attribute Price
is more than 500
http://host/base/odata/standard.odata/Catalog_Goods/$count?$filter=Pri
ce gt 500
```

17.2.1.6.2. $inlinecount

The parameter allows to specify that the system should include the result of the query not only into acquired records but also into the number of these records. To do this, you should specify the value of the parameter $inlinecountequal to allpages. Use of the parameter $inlinecount with the value none suppresses

returning the number of records together with the result of the query. If the parameter $inlinecount is not specified in the query body, the number of records along with the query result is not returned. Using the parameter $inlinecountwill result in ignoring the parameters $skipand $top. Shared usage of the parameters $filter and $inlinecount is allowed.

Example:

```
http://host/base/odata/standard.odata/Catalog_goods?$inlinecount=allpa
ges
```

Example of a query result, with the parameter $inlinecount=allpages:

Format atom-xml:

```
<?xml version="1.0" encoding="UTF 8"?>
<feed xmlns=http://www.w3.org/2005/Atom
xmlns:at=http://purl.org/atompub/tombstones/1.0
xmlns:d=http://schemas.microsoft.com/ado/2007/08/dataservices
xmlns:m=http://schemas.microsoft.com/ado/2007/08/dataservices/metadata
xml:base="http://localhost/ODataTests/odata/standard.odata/">
  <entry>
    …
  </entry>
    …
  <entry>
    …
  </entry>
  <m:count>19</m:count>
</feed>
```

Format json:

```
{
  "odata.metadata":
"http://localhost/ODataTests/odata/standard.odata/$metadata#Document_D
emoDocument",
  "odata.count": "19",
  "value": [{...}, ..., {...}]
}
```

17.2.1.6.3. $orderby

This parameter allows to set the order of the query result. The value of the parameter $orderby contains a comma-separated list of attributes. The attribute value may include the ordering direction:

- asc - for grouping in ascending order.
- desc - for grouping in descending order.

The ordering direction must be separated from the attribute name with 1 or more spaces. If asc or desc is not specified, grouping in ascending order is considered to be required.

Example:

```
http://host/base/odata/standard.odata/Catalog_Goods?$orderby=Name asc,
Manufacturer desc
```

Ordering by subordinate attributes properties is allowed. In this case, the full attribute name, according to which ordering is performed, is formed with usage of the separator "/".

Example:

```
http://host/base/odata/standard.odata/Document_Expense?$orderby=Counte
rparty/TIN asc
```

In this example, grouping will be performed in ascending order (asc) by the `TIN` attribute value of the entity, to which the `Expense` document `counterparty` refer.

17.2.1.6.4. $expand

This parameter allows to get the values of related entities together with the main query results, which allows to avoid requesting each entity separately.
The parameter value will be presented with a list of links or comma-separated composite attributes (if the composite attributes can take the reference value). For expansion of attributes of composite attributes, the type name, to which the composite attribute can keep the reference, must be put after the composite attribute name, and the expandable attribute is put yet after that. In this case, if the composite attribute of the entity has a link to an object of the type specified, then its specified attribute will be expanded. If the value is presented by a link to an object of another type or is not a link, then expansion will not be performed.
The option can be used for expanding entities from the results of functions `SliceLast()` and `SliceFirst()`.

Example:

```
http://host/base/odata/standard.odata/Catalog_Clients?$expand=Certific
ate
```

For each item of the catalog `Clients`, an object referenced by the attribute `Certificate` must be put in the query result.

Example:

```
http://host/base/odata/standard.odata/Catalog_Clients?$expand=Contract
/Document_LabourContract/Application
```

For each item from the catalog `Clients`, an object referenced by a composite type attribute `Contract` must be added to the query result (if it contains any reference value). If the attribute `Contract` keeps a link to the object type `DocumentRef.LabourContract`, then the object referenced by `Contract.Application` must be added to the query result as well.

The value `*` in the path end of the parameter $expand means a need to expand all the links inside the entity. However, it is not allowed to expand navigation fields of a composite attribute without explicit specification of the link type.

Example:

```
http://host/base/odata/standard.odata/Catalog_Clients?$expand=Contract
/Document_LabourContract/*
```

In this example, `Contract` - is a composite attribute. This option of specifying `*` is supported.

Example:

```
http://host/base/odata/standard.odata/Catalog_Clients?$expand=Contract
/*
```

In this example, `Contract` - is a composite attribute. The given option of specifying `*` is not supported.

Examples:

```
http://host/base/odata/standard.odata/Catalog_Clients?$expand=*
```

For each item of the catalog `Clients`, objects referenced by all `Client` object links must be added to the query result.

Example:

```
http://host/base/odata/standard.odata/Catalog_Clients?$expand=Contract
/*
```

For each item of the catalog `Clients`, objects referenced by all links inside the object `Contract` must be added to the query result, as well as the object `Contract` itself. At the same time, the attribute `Contract` is a reference type attribute.

Restrictions of the parameter $expand:

- Tabular section attributes extension is not supported.

- While requesting single entities, extension is not supported (entity, tabular section rows, register record).
- Extension of reference and composite types of virtual tables does not correspond to the OData version 3.0 protocol.
  The query, which will execute an extension of the virtual table dimension, will look as follows:

```
http://host/base/odata/standard.odata/AccountingRegister_AccountingReg
ister/Balance?$expand=Organization
```

Execution of this query will result in the following:

```
<d:Organization_Key>UniqueID</d:Organization_Key>
<d:Organization>
  …ObjectData…
<d:Organization>
```

With a composite type extension, the behavior is similar, except that the object data will be presented in an item with the _Expanded suffix.

17.2.1.7. The ways to get the description of the standard OData interface

If only a simplified description of the standard OData interface (only a list of entities) is needed, it is required to execute a GET-query with usage of URL http://host/base/odata/standard.odata.

If a standard description of the OData interface is needed, it is required to execute a GET-query with usage of URL http://host/base/odata/standard.odata/$metadata. As a result, the complete list of available entities, their attributes and functions will be received as an XML document. The detailed description of the document can be got at the address http://www.odata.org/documentation/odata-version-3-0/common-schema-definition-language-csdl/ (in the English language).

If data in the format **json** is retrieved, requesting for retrieving an extended meta data description of the type http://host/base/odata/standard.odata/$metadata is not supported. However, there is a capability to manage the meta data information details, which are returned when retrieving the data (for both entities and lists of entities). To do this, it is necessary to generate the parameter $format in a special way while executing the query:

- $format=application/json;odata=minimalmetadata - in this case, meta data information is passed in the minimum volume. This is a default value.
- $format=application/json;odata=nometadata - in this case, meta data information is not passed at all.
- $format=application/json;odata=fullmetadata - in this case, meta data information is passed in the full volume.

17.2.1.8. The ways of retreiving data

Using the standard OData interface it is possible to retrieve both lists of entities and entities themselves. These ways of retrieving data differ by the URL used for addressing the data.

The URL for retrieving a list of entities looks as follows: http://host/base/odata/standard.odata/Catalog_Goods. This is **the entities list URL**. To retrieve an entity, the URL will look as follows: http://host/base/odata/standard.odata/Catalog_Goods(guid'value'). In the general form, such an address will be named **the canonical URL of an entity instance**. The difference lies in the fact that in the first case the complete table of the goods catalog is retrieved, and in the second case one object is chosen, which is described with a set of key values (in the example, there is a single key value - link to the object). In a particular case, the set of key fields describing the entity depends on the entity retrieved. If the number of key fields is more than one, all of them must be specified. The key values are specified in comma-separated pairs Name=Value. If there is one key value, - there is no need to specify the key name. For example, in order to get the currency rate value for a particular date, you should execute a GET-query with the following URL:

```
http://host/base/odata/standard.odata/InformationRegister_CurrencyRate
s(Period=datetime'2008 02 05T00:00:00', Currency_Key=guid'9d5c4222
8c4c 11db a9b0 00055d49b45e')
```

When retrieving data there is a capability to specify, which properties of an entity (or of a set of entities) are required to be retrieved using the standard OData interface. For this purpose the keyword $select exists, which has a few options of usage:

- A flag to get all the properties.
  In this case, an expression of the type $select=* should be specified, or no expression at all $select.
- The list of particular properties that should be retrieved.
  In this case, it is necessary to specify a (comma-separated) list of all required fields in the operator expression $select. Property suffixes should be specified at that, if necessary. For example, in order to get the values of the catalog reference and properties Code and SKU for the catalog Goods, you should use the following URL:

```
http://host/base/odata/standard.odata/Catalog_Goods?$select=Ref_Key,
Code, SKU
```

In order to get the same fields for a particular instance of entity (with a known ID), the following URL will be used:

```
http://host/base/odata/standard.odata/Catalog_Goods(guid'value')?$sele
ct=Ref_Key, Code, SKU
```

- A flag to get all the properties except tabular sections.
  To do so, you should specify the value ** in the operator expression $select. When using the following URL all the documents ProductSales will be retrieved, with all the attributes except the tabular sections.

```
http://host/base/odata/standard.odata/Document_ProductSales?$select=**
```

- Specifying a list for retrieving related entities.
  Shared usage of the query parameters $select and $expand is accepted. In this case, you can use the parameter $expand to specify the retrieved entity properties, which should be placed in the query result. Particular values of the retrieved (or expandable) properties are specified using the parameter $select. In the parameter $select, when it is used together with the parameter $expand, using the symbols * and** is supported. Specifying the symbol * means that you need to expand all the attributes of the object retrieved. Specifying the symbol ** means that you need to expand all the attributes of the object retrieved, except the tabular sections.

```
http://host/base/odata/standard.odata/Catalog_Clients?$expand=Certific
ate&$select=Certificate/Data
http://host/base/odata/standard.odata/Catalog_Clients?$expand=Contract
/Document_LabourContract/Application&$select=Contract/Document_LabourC
ontract/Application/**
http://host/base/odata/standard.odata/InformationRegister_CurrencyRate
s?$expand=Currency&$select=Period,Currency/Code,Rate
http://host/base/odata/standard.odata/Catalog_Clients?$expand=Certific
ate&$select=Certificate/*
```

If the parameter $select contains fields of the types Object/*, Object/**, Object/AttributeName at the same time, then they are processed in the following order:

1. If the symbol * is specified, all the attributes are included;
2. Otherwise, if the symbol ** is specified, all the attributes are included except tabular sections, other directions are ingnored. Thus, if an expression of the type $select=Object/**, Object/TabularSection is specified, the tabular section will not be included.
3. Otherwise, all the specified attributes will be included.

There is a capability to retrieve data, which are available "through a dot". In this case, you need to add the property names put "through a dot" but separated by "/", to the address describing a **particular** entity. To do this, you should use English attribute names only for standard attributes. For example, in order to get the property Name of the attribute Currency of the document

ProductSales, you should use a GET-query with the following URL: http://host/base/odata/standard.odata/Document_ProductSales(guid'value')/Currency/Description.

17.2.1.9. Execution of functions and actions

Functions can be related to some entities and sets of entities. For example, functions are used to handle with register virtual tables. In this case, the URL of the resource is generated as follows: http://host/base/odata/standard.odata/<resource>/<function>(<parameters>). Let's consider in detail what the address consists of. The first part of the address (http://host/base/odata/standard.odata) represents a standard address prefix when accessing the standard OData interface. <resource> - resource name, the rules of generating of which are provided on p. 1153. The function name corresponds to the English name of the query language virtual table. The function's <parameters> are set by pairs Key=Value and separated by commas. If the filter is used as a function parameter, then the expression describing the filter should satisfy the general rules of filter descriptions. Thus, getting a section of the currency rates, which are first for the register (with parameters), will be executed at the following URL: http:// host/base/odata/standard.odata/InformationRegister_CurrencyRates/SliceFirst(Period=datetime'2008-01-01T00:00:00',Condition='Currency_Key eq guid'value'').

17.2.1.10. Erroneous situations

If an erroneous situation occurs, an answer with the HTTP-status 4XX or 5XX is returned. The status 4XX informs of errors on the client application side, the status 5XX informs of errors on the server side.

In the case of the status 4XX the server tries the clarify the reason of the error and can pass the additional internal error code and the information message (as an XML-document) in the answer body to the client application.

Example:

```
<m:error
xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadat
a">
  <m:code>9</m:code>
  <m:message>Entity instance "InvoiceShipments" is not found at the
passed key.</m:message>
</m:error>
```

Internal error codes with appearance reason description are listed further:

Code	Description
0	The capability is not supported

Code	Description
1	Parsing the row has failed
2	Invalid query format
3	The requested representation type is not supported
4	The wrong property value
5	The obligatory property value is absent.
6	Wrong URL
7	An entity key item is absent.
8	No entity type is found.
9	No entity instance is found.
10	The requested property is not found.
11	The method is not found.
12	The obligatory method argument is absent.
13	The creation of tabular section rows is not supported directly.
14	A query option parsing error
15	A entity with such a key already exists
16	Assigning a property has failed
17	The object does not support the data loading mode
18	OData interface initialization error: the object has properties with the same names
19	The applied HTTP-method is forbidden in the given context
20	Access rights error. It may occur: • When the user does not have rights for the requested action on the given object; • When an object unavailable due to data access restriction is included in the selection and the parameter allowedOnly is not used.
21	Call of an unimplemented function. Specification of a wrong number of function arguments. Try to pass a wrong type argument Specification of an unimplemented lambda-function.

17.2.1.11. Setting separator values

Separator values can be set only in the following ways:

- Using URL fragments when accessing the standard OData interface;
- Using the publication description file default.vrd.

Specification of separators with a command-line option Z is not supported.

17.2.1.12. Publishing a standard OData interface

Publishing a standard OData interface is executed by using a dialog box publication on the web-server (Administration - Publication on the web-server).
In order to make configuration objects available via the standard OData interface, you should allow it with the global context method `SetStandardODataInterfaceContent()`. If the application functions in the version 8.3.4 (and earlier) compatibility mode, this method is not used. In this case, using the standard OData interface makes all the supported configuration objects available.
Using this method provides for restricting the list of available configuration objects to those (from the overall list of supported ones), access to which is really required for external applications.
The feature of object components setting, which are available using the standard OData interface, can be executed with an external data processor. To do this, you don't need to modify the application.

17.2.1.13. Performing standard operations

This section contains examples of performing some standard operations while handling data with usage of the standard OData interface.

IMPORTANT! These examples are not complete. They are presented to illustrate usage of different constructions.

17.2.1.13.1. Handling one object

Data reading

You should use the canonical URL of the entity to get information. Reading is performed with usage of a GET-query.

Example how to read a reference object:

```
http://host/base/odata/standard.odata/Catalog_Goods(guid'value')
```

Example of reading a record set of the subordinate information register:

```
http://host/base/odata/standard.odata/InformationRegister_InitialPrice
s(Recorder_Key=guid'value')
```

In this example, guid'value' identifies the document that has moved in the information register.

Example of reading an information register record:

```
http://host/base/odata/standard.odata/InformationRegister_ReceiptPrice
s_RecordType(Product_Key=guid'value', ТипЦены="Receipt")
```

Example of reading a particular document row:

```
http://host/base/odata/standard.odata/Document_ProductSales_Goods(Ref_
Key=guid'value', LineNumber=1)
```

Reading a virtual table `SliceLast` of the information register `CurrencyRates` (with the dimension filter `MainCurrency` of the type `CatalogRef.Currencies`) looks as follows:

Example of reading the independent information register:

```
http://host/base/odata/standard.odata/InformationRegister_CurrencyRate
s/SliceLast?Condition=Currency/MainCurrency_Key eq guid'value'
```

Example of reading the subordinate information register (you should add _RecordType to the register name):

```
http://host/base/odata/standard.odata/InformationRegister_CurrencyRate
s_RecordType/SliceLast?Condition=Currency/MainCurrency_Key eq
guid'value'
```

In this example, guid'value' identifies the catalog item `Currencies`, according to the value of which the filter is executed.

Example of retrieving the account balance from the accounting register with a particular counterparty condition:

```
http://host/base/odata/standard.odata/AccountingRegister_SelfSupportin
g/Balance(AccountCondition='Account_Key eq
guid'value1'',Condition='ExtDimension1 eq cast(guid'value2',
'Catalog_Counterparties')',ExtraDimensions='value3')?$format=json
```

In this example:

- guid'value1' - the UUID of an item of the chart of accounts;
- guid'value2' -the UUID of an item of the chart of characteristic types `TypesExtDimensionSelfSupporting` with the value type `CatalogRef.Counterparties`;
- 'value3' - the UUID of an item of the catalog `Counterparties`.

Creating an object

In order to create an object you should use a POST-query with usage of the URL set of entities and passing the document (in the supported format), which contains

field values of the object created, in the query body. If the passed document contains properties, which the object created does not have, then these properties are ignored.
Further an example of creating an item of the catalog `Goods` and the answer of the standard OData interface after successful completion of the operation are provided.

Example of a POST-query (format atom):

```
POST http://host/base/odata/standard.odata/Catalog_Goods HTTP/1.1
User Agent: Fiddler
Host: host
Content Length: 981

<entry>
  <category term="StandardODATA.Catalog_Goods"
scheme="http://schemas.microsoft.com/ado/2007/08/dataservices/scheme"/
>
  <title type="text"/>
  <updated>2014 02 14T12:05:55</updated>
  <author/>
  <summary/>
  <content type="application/xml">
    <m:properties
xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices"
xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadat
a">
     <d:DeletionMark>false</d:DeletionMark>
     <d:Parent_Key>bbb079ae 8c51 11db a9b0 00055d49b45e</d:Parent_Key>
     <d:IsFolder>false</d:IsFolder>
     <d:Code>000000800</d:Code>
     <d:Description>FlipFlops</d:Description>
     <d:SKU>SL56X</d:SKU>
     <d:Supplier_Key>086715b0 f348 11db a9c5
00055d49b45e</d:Supplier_Key>
     <d:Type>Product</d:Type>
     <d:Barcode/>
     <d:Description>&lt;html&gt;Beach flip
flops&lt;/html&gt;</d:Description>
    </m:properties>
  </content>
</entry>
```

An example of an answer of the standard OData interface:

```
HTTP/1.1 201 Created
Content Length: 2705
Content Type: application/atom+xml;type=entry;charset=utf 8
Location:
http://host/base/odata/standard.odata/Catalog_Goods(guid'41aa6331 954f
11e3 814b 005056c00008')
Server: Microsoft IIS/7.5
DataServiceVersion: 3.0
X Powered By: ASP.NET
Date: Fri, 14 Feb 2014 08:18:36 GMT

<?xml version="1.0" encoding="UTF 8"?>
<entry xmlns="http://www.w3.org/2005/Atom"
```

```
  xmlns:d=http://schemas.microsoft.com/ado/2007/08/dataservices
xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadat
a">
  <id>http://host/base/odata/standard.odata/Catalog_Goods(guid'41aa633
1 954f 11e3 814b 005056c00008')</id>
  <category term="StandardODATA.Catalog_Goods"
scheme="http://schemas.microsoft.com/ado/2007/08/dataservices/scheme"/
>
  <title type="text"/>
  <updated>2014 02 14T12:18:36</updated>
  <author/>
  <summary/>
  <link
rel="http://schemas.microsoft.com/ado/2007/08/dataservices/related/Pic
tureFile"
     href="Catalog_Goods(guid'41aa6331 954f 11e3 814b
005056c00008')/PictureFile"
     type="application/atom+xml;type=entry;charset=utf 8"
     title="PictureFile"/>
  <link
rel="http://schemas.microsoft.com/ado/2007/08/dataservices/related/Par
ent"
     href="Catalog_Goods(guid'41aa6331 954f 11e3 814b
005056c00008')/Parent"
     type="application/atom+xml;type=entry;charset=utf 8"
     title="Parent"/>
  <link
rel="http://schemas.microsoft.com/ado/2007/08/dataservices/related/Sup
plier"
     href="Catalog_Goods(guid'41aa6331 954f 11e3 814b
005056c00008')/Supplier"
     type="application/atom+xml;type=entry;charset=utf 8"
     title="Supplier"/>
  <link rel="edit"
     href="Catalog_Goods(guid'41aa6331 954f 11e3 814b 005056c00008')"
     title="edit link"/>
  <content type="application/xml">
    <m:properties
xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices"
xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadat
a">
     <d:Ref_Key>41aa6331 954f 11e3 814b 005056c00008</d:Ref_Key>
     <d:DataVersion m:null="true"/>
     <d:Description>FlipFlops</d:Description>
     <d:Code>000000800</d:Code>
     <d:Parent_Key>bbb079ae 8c51 11db a9b0 00055d49b45e</d:Parent_Key>
     <d:IsFolder>false</d:IsFolder>
     <d:DeletionMark>false</d:DeletionMark>
     <d:SKU>SL56X</d:SKU>
     <d:Supplier_Key>086715b0 f348 11db a9c5
00055d49b45e</d:Supplier_Key>
     <d:PictureFile_Key>00000000 0000 0000 0000
000000000000</d:PictureFile_Key>
```

```
     <d:Type>Product</d:Type>
     <d:Barcode/>
     <d:Description>&lt;html&gt;Beach flip
flops&lt;/html&gt;</d:Description>
     <d:PictureFile_Key>00000000 0000 0000 0000
000000000000</d:PictureFile_Key>
     <d:Parent_Key>bbb079ae 8c51 11db a9b0 00055d49b45e</d:Parent_Key>
     <d:Supplier_Key>086715b0 f348 11db a9c5
00055d49b45e</d:Supplier_Key>
    </m:properties>
  </content>
</entry>
```

Example of a POST-query (format json):

```
POST http://host/base/odata/standard.odata/Catalog_Goods HTTP/1.1
Accept: application/json
Accept Charset: UTF 8
User Agent: Fiddler
Content Type: application/json
Content Length: 2426

{
"DeletionMark":false,
"Parent_Key":"bbb079ae 8c51 11db a9b0 00055d49b45e",
"IsFolder":false
"Code":"000000800",
"Description":"FlipFlops",
"SKU":"SL56X",
"Supplier_Key":"086715b0 f348 11db a9c5 00055d49b45e",
"Type":"Product",
"Barcode":null,
"Description:"Beach flip flops"
}
```

Object update

In order to update an object, you should execute a PUT-/PATCH-query with usage of the entity's canonical URL (similarly to a GET-query for retrieving an entity), by passing an XML document in the query body (in the format atom) and a JSON document (in the format json), which contains entity property values. If the passed document contains properties, which the object created does not have, then these properties are ignored.

In the case of a PATCH-query the omitted entity properties will be ignored, i. e. only the properties passed for changing in the query will be changed. You should specify values of all the properties of the entity updated for a PUT-query.

When modifying an entity with a PUT-query, when recording values of reference attributes, you should use the field `ReferenceAttributeName@odata.bind`, where you should place URI

entities (accepted in the short option- only a path fragment after standard.odata).

Example:

```
PUT http://host
/base/odata/standard.odata/Document_Document(guid'19961ec3 3c5f 11e7
8785 50465da19fe4')?$format=json
{
  "Date": "2015 01 01T12:01:07",
  "Organization@odata.bind": "Catalog_Organizations(guid'298584b5 92e4
11e3 8bc3 0050568b1678')",
  "Warehouse@odata.bind": http://host
/base/odata/standard.odata/Catalog_Warehouses(guid'081bc346 2abc 13e4
a1bd 0050568b1688')
  …
}
```

In the next example changing the attribute `Name` and the tabular section components `SalesAreas` of the catalog `Shops` is considered. The other catalog attributes are not used under this example. The data format json is used in this example.

Example of a PATCH-query:

```
PATCH
http://host/base/odata/standard.odata/Catalog_Shops(guid'value')?$form
at=json HTTP/1.1

Host: host
Connection: keep alive
Accept: application/json
Content Length: 638

{
  "odata.metadata":
"http://host/base/odata/standard.odata/$metadata#Catalog_Shops/@Elemen
t",
  "Description": "New description of the shop",
  "SalesAreas@odata.type" :
"Collection(StandardODATA.Catalog_Shops_SalesAreas_RowType)"
  "SalesAreas": [
    {
      "LineNumber": "1",
      "Name": "Blue Area",
      "Square": 56,
      "OpeningDate": "2015 01 01T00:00:00"
  },
  {
      "LineNumber": "2",
      "Name": "Red area",
      "Square": 56,
      "OpeningDate": "2015 06 13T11:45:41"
  }
  ]
}
```

IMPORTANT! You should pass the tabular section in full (all the rows), even in the case when data in only one row of this tabular section must be changed.

In this example guid'value' identifies the changeable item of the catalog Shops.

Deleting an object

In order to delete an object, you should use a DELETE-query with usage of the entity canonical URL.

IMPORTANT! The deletion mark is not performed, the object is to be deleted directly.

Optimistic data blocking

In order to perform optimistic data blocking (for verifying that the data has not changed since reading), you should use the heading `If-Match` of the HTTP-query, related to data modification (PATCH) or data deletion (DELETE). The heading value will be presented by the value of the property `DataVersion`, which was retrieved at the first reading of the entity.
The mechanism works as follows:

- When an entity instance is requested, the property `DataVersion` is returned among others. When an entity set is requested, the property `DataVersion` is returned from each entity of the set.
- The application must save this object version value for itself, and then use it in PATCH and DELETE queries by passing it in the heading `If-Match`.
- If the value of the property `DataVersion` coincides with the value of the heading `If-Match` at the query execution moment, the requested action takes place. Otherwise, the action is not performed, and the status 412 HTTP is returned to the client.

Recording an object in the data loading mode

If you need to emulate a record, which is executed during operation of the data exchange feature (property `DataExchange.Load = Truth`), while recording an object, you should use a HTTP heading 1C_OData-DataLoadMode with the value true, when you perform the applicable record operation.

Posting or cancelling posting of a document

In order to post a document, you should execute a POST-query with usage of the canonical entity URL (similarly to a GET query for retrieving an entity), by specifying a specially generated URL suffix. In this case, the suffix will consist of the command Post and a parameter specifying the document posting mode: http://host/base/odata/standard.odata/Document_ProductSales(guid'value')/Post?PostingModeOperational=false.
With the purpose of canceling a posting of a document, the suffix consists of the command Unpost without parameters.

17.2.1.13.2. Handling an object collection

In order to get an entity set of any type, you should execute a GET-query, using URL of the following type: http://host/base/odata/standard.odata/ExchangePlan_DataExchange.
If you need to set a filter for the retrieved list, it can be executed with the parameter $filter, which is put in URL:

```
http://host/base/standard.odata/Document_Receipt?$filter=Priority eq 1
```

Addressing the register virtual table looks as follows (after the example of the virtual table `BalanceAndTurnovers` of the accumulation register `Inventory`):

```
http://host/base/odata/standard.odata/AccumulationRegister_Inventory/B
alanceAndTurnovers(StartPeriod=datetime'2014 01 01',
EndPeriod=datetime'2014 02 01', Condition='Product_Key eq
guid'value'')
```

You can also use the parameters $top and $select for restriction of the entity set.

17.2.1.13.3. Handling exchange plans

Generating an exchange message

In order to generate an exchange message you need to execute a POST-query with usage of URL of the following type: http://host/base/odata/standard.odata/SelectChanges?<params>. You should specify the following as parameters:

- The parameter DataExchangePoint - must contain the entity canonical URL of the required exchange plan item;
- The parameter MessageNo - must contain the data exchange message number, which will be generated as a result of the given server call.

As a result, the complete URL for generating an exchange message will look as follows: http://host/base/odata/standard.odata/SelectChanges?DataExchangePoint='http://host/base/odata/standard.odata/ExchangePlan_DataExchange(guid'value')'&MessageNo=34.
This will result in a list of changes, which are to be passed to another node as an atom-feed stream. Each item will be presented in the atom-entry format, and the deleted items will be presented in the format atom-deleted-entry (RFC 6721, http://tools.ietf.org/html/rfc6721, in the English language).

Notification of changes receipt

In order to notify the server that the exchange message has been successfully received, you need to execute a POST-query with usage of URL of the following type: http://host/base/odata/standard.odata/ NotifyChangesReceived?<params>. You should specify the following as parameters:

- The parameter DataExchangePoint - must contain the entity canonical URL of the required exchange plan item;
- The parameter MessageNo - must contain the data exchange message number, receipt confirmation of which must be commited.

As a result, the full URL for confirmation of the exchange message receipt will look as follows: http://host/base/odata/standard.odata/NotifyChangesReceived?DataExchangePoint='http://host/base/odata/standard.odata/ExchangePlan_DataExchange(guid'value')'&MessageNo=34.

17.2.2. HTTP services

17.2.2.1. General information

Apart from the standard OData interface, the system 1C:Enterprise 8 allows application developers to implement their own HTTP services. Creating HTTP services allows to expand (in comparison with the standard OData interface) or create (if the standard OData interface is not used) some applied interface, which will be accessible with HTTP queries. The general description of the protocol HTTP 1.1 is provided at the link: http://tools.ietf.org/html/rfc2616 (in the English language).

HTTP services (REST services) are analogues of SOAP services, which are presented by web services in 1C:Enterprise. The SOAP service is -a set of parameter-based actions, while the action list is not limited. The action set is determined during the service development. The HTTP service is -a restricted set of actions (restricted by the HTTP methods list) and a branched set of resources. The resources set is determined during the service development process. The resources set determines functional capabilities of the service. While developing SOAP services, you should determine the data structures passed between the client and the server, and while developing the HTTP service, - the structures of passed data are not rigidly determined.

Addressing the HTTP service is executed at some URL. As an example, the HTTP service returns a list of some documents (for example, invoices) while addressing one URL, and while addressing another URL, a particular invoice will be returned. The action that should be executed by the server is determined by the HTTP method addressed to the server. For example, if a GET query is

sent to the server, then receipt of some data is performed, an DELETE query results in deleting some data, etc.
While implementing a HTTP service the application developer determines the following service items:

1. He determines the basic part of the URL, at which addressing the service will be performed.
2. He determines the components of the provided functions and the structure of resources, on which these functions will be displayed.
3. He determines the actions (HTTP methods), which will be executed while addressing any resource.
4. Each chosen action implies implementing a specialized method, described in the 1C:Enterprise language, which implements the required functions. The method is located in the specialized module, related to the HTTP service.

After implementing the HTTP service, you should publish it on the web server with usage of the standard publishing mechanism.
When addressing the HTTP service an URL is generated, which looks as follows: http://host/base/hs/RootURL/relativeURL. Let's consider the address components in more detail:

- http://host/base - ordinary URL, for example, used for access to the infobase via a web client. When separators are available, specification of separator values with the command-line parameter Z of the client application startup is not supported.
- hs - the flag that addressing the HTTP service is performed (as opposed to ws, which determines access to the web service).
- rootURL - the resource name which determines the resource group related to the overall sense. Defined in the object property HTTP service.
- relativeURL - determines the resource, to which addressing will be performed. The relative URL, specified in the query, will be used for determining the particular resource, to which addressing is performed. The matching rule is defined in the object URL template

17.2.2.2. HTTP service implementation a

Creation of HTTP services in performed in the branch General HTTP services- of the configuration object tree. First, the service itself is created, then you must create a definite template of the URL fragment (a group of resources) for each service, and then you must define the HTTP method for each resource group, which will be implemented for this group. You also should understand that one service may include more than one URL template (the relation "One to many"). Actually. the URL template defines a resource group, processed under the single rules.

17.2.2.2.1. Editing a HTTP service

When creating a HTTP service you should pay attention to the property Root URL. It defines a group of resources united by the overall sense. For example, if you need to create a few resources, which process orders, the root URL will look as order in this case. Then, while addressing this service, the URL beginning will look as follows: http://host/base/hs/order.

17.2.2.2.2. Editing the URL template

When creating the URL template, possible resource addresses are described, which can be used for addressing the HTTP service. The template itself is defined in the property Template. When creating a template you can use the following symbols:

- Any symbols acceptable for the 1C:Enterprise language IDs.
- Symbol «/»;
- Symbols «{}» with nonempty text between them;
- Symbol *.

For example, the template may look as follows:

```
/query
/documents/{id}/props/{PropertyName}/*
```

When describing the template the following terms will be used:

- The segment -part of URL between two symbols «/» For example, propsfrom the example above is a segment.
- Параметризованный сегмент - сегмент, заключенный в фигурные скобки («{сегмент}»). Actually the parameter-based segment describes a variable with the name specified, accessible from the 1C:Enterprise language.

The segment (not parameter-based) from the template must be passed to the URL word by word. The parameter-based segment can be presented by any value that is acceptable for URL. The symbol «*» can be located only in the end of a template. Any number of segments, including absence of symbols, can be located under the symbol «*». One template cannot contain parameter-based segments with the same name. The parameter-based segment can consist of only letters, numbers and the symbol «_».

The template type /documents/*/number is not acceptable. If the template type /documents/{kind}/{number}is set, the following service URLs are valid:

```
/documents/trade/13
/documents/12/test
```

Upon receiving a query, the application tries to determine, which template should be used. For this purpose, the application attempts to compare the incoming query with the templates set. Comparing is performed according to the following rules:

1. All the rows are compared depending on the register;
2. The exact match always has more priority than compliance with the template with parameter-based segments or the symbol «*»;
3. The symbol «*» can be compared with any number of segments (including absence of a segment);
4. A parameter-based segment has priority over the symbol «*»;
5. If the passed query could not be fully matched to any template, - a mistake will be generated.

These rules clarify the following examples:

1. While comparing the URL /Invoice with templates /Invoice/ and /{EntityName}/, the first template will be chosen for using;
2. The template /Document/*/{Num} is not valid;
3. URL /Invoice/500/Rows/2/Product may be compared to the template /Invoice/*;
4. While comparing /Invoice/500 with templates /Document/{Num} and /{Document}/*, the first template will be chosen for using;
5. While comparing URL /Invoice/500/ to the templates {Document}/{Num} and /{Document}/{Number} , an error will occur, as choosing a proper template failed.

17.2.2.2.3. Editing the method

You should define HTTP methods for the created template; they may be used for processing the resource, which coincides with the template. To do this, you should create a required number of objects Method, that are subordinate to the object URL Template. You should specify the property HTTP-method in each method and create a handler. The handler name is automatically generated from the URL template name and the actual method name. Thus, if the name ExampleForDocumentation is specified in the URL template, and the name GET is specified for the method, then the handler name `ExampleForDocumentationGET()` will be used for the method of the 1C:Enterprise language, which will be used for processing the HTTP query.

TIP. It is recommended to name HTTP-method objects by the name of the servicing method in the English language.

There is capability to create handlers for the following methods:

- GET;
- POST;
- PUT;
- DELETE;
- PATCH;
- MERGE;
- CONNECT;
- OPTIONS;
- TRACE;
- PROPFIND;
- PROPPATCH;
- MKCOL;
- COPY;
- MOVE;
- LOCK;
- UNLOCK.

If a handler for the method Any is created, then the handler of this method will be called for all HTTP methods (from the list provided above), which are not clearly described in the template, which "processes" the incoming HTTP query.
After the required template is received, an attempt will be taken to call the method servicing the required HTTP query (GET, POST etc.). If the handler of the required query has not been found, the calling part will get error 405 returned.
If a 1C:Enterprise language error is detected during the method execution process, - the calling part will get error 500 returned, and the exclusion text will be placed in the answer body.
While generating an answer the parameter Content-Length may be non-specified. In this case, it will be defined automatically.

17.2.2.2.4. Input and output data of the method handler

The handler of any HTTP method is presented by a function that acquires one parameter of the type `HTTPServiceRequest` for the input, and must return an object of the type `HTTPServiceResponse`.
The input parameter fully describes the received query. The object allows to determine:

- Which HTTP method is used in the query (property `HTTPMethod`);

- At which URL the query has been received (property `BaseURL`) - part of the full URL query, not including the relative URL and the query parameters;
- The relative URL (property `RelativeURL`) - part of the full URL query that was used for defining the resource group and the processing rules of the incoming HTTP query;
- What headings (property `Headings`) the HTTP query contains;
- What parameter-based segments (property `URLParameters`) are separated from the incoming HTTP query and their values.
- What parameters (property `QueryOptions)` are listed in the URL of the HTTP query after the symbol "?" and their values.

You also have a capability to receive the query body as binary data or a line (depending on the information passed).

Thus, an application developer does not have "to deal with" the incoming query independently. The example provided further in this chapter allows to see the way how the incoming HTTP query is distributed between the object properties `HTTPServiceRequest`.

After the application has executed all the actions defined for this resource and the HTTP method, you should generate a service answer. To do this, you should create and object of the type `HTTPServiceResponse`. Depending on the processing result, you should specify the property `StatusCode` that describes the standard HTTP return code (http://tools.ietf.org/html/rfc2616#section-10, in the English language). Then you should generate the HTTP service answer body, if necessary. The body may be set as binary data, as a line or with usage of the file name specification, where the answer body will be loaded from. If the value type returned by the function (which implements reaction to the HTTP method) is different from `HTTPServiceResponse`, then error 500 will be returned to the client.

17.2.2.3. Examples of HTTP services implementation

NOTE. The examples below are not complete. They are intended for demonstrating some capabilities of HTTP services.

17.2.2.3.1. Displaying an incoming query

As an example, let's implement a HTTP service that will return all the incoming query parameters as an answer.

You should create a HTTP service WorkExample for this, with specification of the root URL example. Then you should create an URL template ShowQuery with the template /query/{string}/*. Now you should add the following methods to the template ShowQuery:

- Method GET for the HTTP query GET;
- Method ANY for the HTTP query Any.

Handlers should look as follows:

```
Function ShowQueryGET(Query)
  Answer = New HTTPServiceResponse(200);
  Result = "";
  Result = Result + Query.HTTPMethod + Symbols.LF;
  Result = Result + Query.BaseURL + Symbols.LF;
  Result = Result + Query.RelativeURL + Symbols.LF;
  Result = Result + "Headings:" + Symbols.LF;
  For each parameter, from Query.Headings Do
    Result = Result + Parameter.Key + ":" + Parameter.Value +
Symbols.LF;
  EndDo;
  Result = Result + "Parameters URL:" + Symbols.LF;
  For each parameter, from Query.URLParameters Do
    Result = Result + Parameter.Key + ":" + Parameter.Value +
Symbols.LF;
  EndDo;
  Result = Result + "Query parameters:" + Symbols.LF;
  For each parameter, from Query.QueryOptions Do
    Result = Result + Parameter.Key + ":" + Parameter.Value +
Symbols.LF;
  EndDo;

  Answer.SetBodyFromString(Result);
  Return Answer;
EndFunction

Function ShowQueryANY(Query)
  Answer = New HTTPServiceResponse(405);
  Answer.SetBodyFromString("This HTTP does  not support the method " +
Query.HTTPMethod);
  Return Answer;
EndFunction
```

After publishing HTTP services, the service must return full "description" of the incoming query.

When using the following query: http://localhost/httpservice/hs/example/query/value/another?p1=1&p2=2 the following answer will be received (the value of some headings depends on the web browser used):

```
GET
http://localhost/httpservice/hs/example
/query/value/another
Headings:
Accept Encoding:gzip,deflate,sdch
Connection:keep alive
Accept Language:ru RU,ru;q=0.8,en US;q=0.6,en;q=0.4
DNT:1
Accept:text/html,application/xhtml+xml,application/xml;q=0.9,image/web
p,*/*;q=0.8
Content Length:0
User Agent:Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/32.0.1700.107 Safari/537.36
Host:localhost
URL parameters:
*:/another
string:value
Query parameters:
p1:1
p2:2
```

Let's see how the query was parsed. The example specified that the root URL of the service was - example, URL template - /query/{string}/* and a handler of the HTTP query GET was generated.
It can be seen from the result that the following URL parameters have been defined:

- Parameter * with the value /another;
- Parameter string with the value value.

It fully corresponds with the URL template specified. If the template URL is set to the value /query/{string}, then the demonstrating query will end up with an error message 404, as this template will not allow information in the URL, after the parameter-based segment.
The method handler will also pass all th query parameters, that were specified in the URL.
Any other query type (different for GET) will result in error 405 with specification, which query is not supported by the service.

17.2.2.3.2. Processing documents

This example is kind of a analogue of the web service that must return the tabular section components of an sales invoice based on its passed number.
This example with provide a capability to receive a list of "documents" and receive a "document" with a defined number. Reaction to a wrong "document"

number will also be implemented. To simplify the example, actual objects of the type `Document` won't be created in the example.

To do this, you should create a HTTP service DocumentList, with specification of the root URL documents. Then you should create an URL template List with the template /* and an URL template Document with the template /{number}. You should the method GET implementation for each URL template.
Handlers should look as follows:

```
Funcation ListGET (Query)
  Answer = New HTTPServiceResponse(200);
  Result = "Sales Invoice, 1, 01.01.2014
  |Sales invoice, 2, 01.01.2014";
  Answer.SetBodyFromString(Result);
  Return Answer;
EndFunction

Function DocumentGET(Query)
  Answer = New HTTPServiceResponse(200);
  Number = Number(Query.URLParameters.Receive("number"));
  If number > 2 then
    Answer.StatusCode = 404;
    Answer.SetBodyFromString("Document with the following number is
unavailable: " +Number);
  Else
    If number=1 then
      Answer.SetBodyFromString("SalesInvoice, 1, 01.01.2014");
    Else
      Answer.SetBodyFromString("SalesInvoice, 2, 01.01.2014");
    EndIf;
  EndIf;
  Return Answer;
EndFunction
```

After publishing HTTP services, the service must return some set of document data, "kept" in the application.
Let's consider how the created service functions:

- The query http://localhost/httpservice/hs/documents will return a list of "documents":

```
Sales invoice, 1, 01.01.2014
Sales invoice, 2, 01.01.2014
```

- Query of the type http://localhost/httpservice/hs/documents/get/list ver.no list of "documents", as this query corresponds to the template List and does not correspond to the template Document.

- Query of the type http://localhost/httpservice/hs/documents/list returns execution time error, as the presented query corresponds to the template Document, but the function `DocumentGET()` does not check if the passed number is a number.
- Query of the type http://localhost/httpservice/hs/documents/2returns a "document" with number 2:

```
Sales invoice, 2, 01.01.2014
```

- Query of the type http://localhost/httpservice/hs/documents/3 returns error:

```
No document with a number is available: 3
```

Processing of the document number perfection is clearly implemented in the method handler.

17.2.2.4. Setting separator values

Separator values can be set only in the following ways:

- Using URL fragments when accessing the REST-service;
- Using the publication description file default.vrd.

Specification of separators with a command-line option Z is not supported.

17.2.2.5. Publishing a HTTP service

Publishing HTTP services is executed by using a publication dialog box on the web server (Administrating - Publishing on the web server).

17.3. Repeated usage of internet services sessions

17.3.1. General information

The structure of 1C:Enterprise is so that calling an internet service (Web service, HTTP service, standard OData service) and a client application startup is almost equal - a session with the infobase is generated, where all the required actions are performed. Upon completing the internet service call (when the client application terminates) the session is finished, and after the repeated call the process is repeated again. The evident disadvantage of such a behaviour is a considerable period of time spent on creating a session (including execution of all event handlers and load of the configuration itself). Particularly it can be noted with "big" applications that contain many different configuration objects. As for the client application, the time of session creating may be insignificant (comparing to the

overall time of the client application's continuous functioning), and as for internet services, this time may bring considerable delays to the overall time of the service call. The application provides a capability to manage the need for creating a session at each internet service call:

- Do not apply the repeated use of sessions. A new session will be created every time the internet service is addressed.
- To activate repeated usage of sessions. In this case, creation and termination of session is regulated by the calling party with usage of HTTP query headings.
- To activate automatic repeated usage of sessions. In this case, a session pool is organized. Each session of the pool is characterized by a set of parameters. If a pool session is free and characterized by the same set of parameters as the internet service query, an existing session will be provided to the query at once.

The capability of repeated usage of sessions is installed in several locations:

- In the configuration object properties of web service and HTTP service (type of repeated usage and lifetime of the session);
- in the publication file default.vrd (session pool parameters).

You should remember that the values in the file default.vrd have priority over the values specified in the properties of configuration objects.

17.3.2. Types of repeated usage of sessions

Do not use

In each case, each query of internet service will result in creating a new session.

Use

In this case, the internet service client directly controls the lifetime and character of the session repeated usage. Specialized HTTP query headings for the internet service are used for the control of creating a new session.

To use automatically

A session pool should be broken down by the following parameters:

- service type;
- service name;
- user;
- password;
- separator set;
- safe mode.

Upon detecting an incoming query to an internet service, the current session pool is analysed, and a session with parameters is searched for, which fully matches the incoming query. If such a session is detected and it is not used by a query of another internet service client, the incoming query will be served by the detected session. If no free session is detected, - an attempt to create a new session is executed. If the session pool size is exceeded when trying to create a new session, - the incoming query will be expecting for some time (pool timeout). If no free session is detected upon pool timeout expiration, - the internet query will be finished with error 406 Not Acceptable. The free session will be destroyed automatically after the free session lifetime is expired.
Settings of the automated session pool are valid within the publication. Thus, if a few publications have been created for some infobase, then the pool setting of the publication, through which the call is performed, is used for calling the internet service.
It is recommended to pick a session pool size, which would allow for a few free sessions at the maximum load (i. e. to choose a pool size with a small reserve). Otherwise, some calls will be performed with errors at the maximum load. 1C:Enterprise does not support automated determination of the necessary session pool size.

17.3.3. Settings of repeated usage of sessions

17.3.3.1. Properties of internet services

The property palette of internet services (web service, HTTP service), including the native internet services of configuration extensions, has two properties describing the repeated usage of sessions:

- The property Repeated usage of sessions -specifies the type of repeated usage of sessions for the customizable service (see details on p. 1193).
- Property Session lifetime:
  - specifies the time period, after which the "free" session is deleted from the session pool (in case of automated repeated usage of sessions);
  - specifies the idle time period, after which the session is forcedly finished by the application if sessions are used repeatedly in the manual mode.

  If this property is set to the value 0, it means that the sessions will not be used repeatedly. This setting is equivalent to the property setting Repeated usage of sessions to the value Do not use.

The session pool parameters for automated repeated usage of sessions is set by using the file default.vrd.

17.3.3.2. At the client side

If Session Reuse for an Internet service is set to Use, it means that session life cycle is managed by an Internet service client. The HTTP query heading IBSession acts as the control. This heading can take two values:

- start - In this case 1C:Enterprise creates a new session, authenticates, sets separators, all the required event handlers are executed. If a new session can't be created, the client will receive error 406 Not Acceptable. If the session has been successfully created, the directive to set cookie ibsession with the created session ID will be placed in the HTTP answer: Set-Cookie: ibsession=<session ID>.
  If you need to use the earlier created session, you should specify the ID of the earlier created session in the HTTP query to 1C:Enterprise: Cookie: ibsession=<session ID>. If the query contains the session ID, which was not created earlier or has been finished, the client receives error 400 Bad Request.
  If the HTTP query does not contain the heading IBSession, the session is created and terminated at each internet service call.
  If the separator values and the session safe mode were changed, when the HTTP query session was used, then the new parameters are ignored, and the session will use the values that were stated at the session start.
- finish - in this case 1C:Enterprise terminates the session that was specified in the query, along with the session termination command: Cookie: ibsession=<session ID>. The session termination takes place automatically, if no actions were executed in this session during its lifetime.

The session pool is not used in this case.

17.3.3.3. In the file default.vrd

The description of the file default.vrd parameters that control repeated usage of sessions.

17.3.3.4. For the standard OData interface

The mode and parameters of the repeated usage of sessions for the standard OData interface are set by 1C:Enterprise while publishing the standard OData interface on the web server. The following default values are used:

- Repeated usage of sessions - automated usage is enabled.
- The lifetime of the session is - 20 seconds.

Chapter 18. Web client and web site integration

18.1. General information

When you implement applications, you may need to integrate a web site with any specific application which is accessed through a web client. Such integration, for example, can provide for opening of a web client inside `<iframe>` element of a web site html-page.

For the purpose of the said integration, the following has to be done:

- Allocate an area on the web site page to run a web client.
- Implement an interface (in 1C:Enterprise language) on the application side responsible for exchanging data with a web site being integrated with an application. Moreover, you need to select a proper mode for the application main window and enable users to operate in a proper client application main window mode by way of exercising their respective access rights.
- Implement an interface (in JavaScript language) on the web site side being integrated with an application to enable a web client to get information as to the mode of operation and exchange of data with the said application.

This section describes all proper interfaces and lists an example of a simple web site and a configuration operating inside the said web site.

General integration requirements are as follows:

1. Make sure that an application is accessible through a web client (i.e. it is published on a web server).
2. Make sure that a web server is correctly set up, so that integration of an application with a web site is supported.
3. To manage a web site with an integrated web client, use web browsers supported by 1C:Enterprise for web clients only.

18.2. Application Interface

Each application has `ExternalSiteWindow` global context property. Applications use this property to interface with web sites, when they are integrated therein. Such an interface allows it to perform the following:

- Verify whether a web client operates as part of a specific web site, or not. This can be done by way of selecting

ExternalSiteWindow.Accessible. Further, it is recommended to perform all actions with an external web site window manager (ExternalSiteWindow), only as soon as you make sure that this property is set to True.

- Get messages from an external web site (vis-a-vis a web client) subject to prior registration of a handler method intended to handle messages received from a web site using ExternalSiteWindow.EnableMessageHandler(). Moreover, you can disable processing of messages received from an external web site, if it is no more needed. For that purpose, select ExternalSiteWindow.DisableMessageHandler().
- Send messages to an external web site using ExternalSiteWindow.SendMessage().

Only text data can be sent or received. Thus, to interact with an external web site, you need to develop your own exchange protocol and format.

Availability of selection of application main window mode is another distinctive feature of application setup process. It can be done as follows:

- Using Main Client Application Window Mode .
- Using a command line intended to start a client application (MainWindowMode).
- Using SetMainWindowMode().

Moreover, you need to ensure availability of a proper client application main window mode by way of granting required access rights.

18.3. Web Site Interface

Do the following on the web site:

- Specify an area where a web client interface will be displayed. For this purpose, use . When loaded, a web client will by default add <iframe> subordinate element to the element so selected specifying all proper details.
- Get the path to load web client JavaScript-interface code (Embedded WebClient API). As soon as this interface is loaded, WebClient1CE becomes available in the web browser. To load JavaScript-interface into the web site text, insert the following: <script src="%WebSiteURL%/%InfobaseName%/scripts/webclient1ce.js"></script>.
- Implement 1C:Enterprise API.

Let's take a closer look at WebClient1CE:

- Wizard`WebClient1CE(webClientContainerID, config)`, where:
  - `webClientContainerID` - of `String` kind - is `<div>` element ID, where `<iframe>` is embedded to run a web client. Mandatory parameter.
  - `config` - of `Object` kind - is an object which specifies web client settings. Mandatory parameter.
- `gotoURL(navigationURL)` - enables you to follow URL so specified. It operates similarly to `FollowURL()` 1C:Enterprise language method. Method parameters:
  - `navigationURL` - of `String` kind - is URL which is followed. This is a mandatory parameter.
- `postMessage(message)` - sends text specified as parameter to a web client. Whenever this method is used in an application, handlers are called which are enabled through `ExternalSiteWindow.EnableMessageHandler()`. Method parameters:
  - `message` - of `String` kind - defines text which is sent to a web client. Mandatory parameter.

A setting object which is sent as a parameter to `WebClient1CE` object wizard has the following properties:

- `webClientURL` - of `String` kind - is a line intended to start a web client (including command line commands and parameters used to start a web client). This is a mandatory parameter.
- `width` - of `String` kind - defines width of an embodiment area (`<iframe>`) specifying a unit of measure. This is an optional parameter.
- `height` - of `String` kind - defines height of an embodiment area (`<iframe>`) specifying a unit of measure. This is an optional parameter.
- `events` - of `Object` kind - is an object which describes event handlers received from the embedded web client. This is an optional parameter.

An object which describes event handlers received from a web client has the following properties:

- `onStart` - of `Function` kind - is a function (no parameters) called as soon as an application is started. It is called, as soon as operation of `OnStartSystemOperation` event handler is terminated in the embedded application. This is an optional parameter.
- `onEnd` - of `Function` kind - is a function (no parameters) called before operation an application is terminated. It is called, as soon as operation of

`OnEndSystemOperation` event handler is terminated in the embedded application. This is an optional parameter.

- `onMessage` - of `Function` kind - is a function which is called to get a message from an application. To send a message, select `ExternalSiteWindow.SendMessage()`. `onMessage` has the following parameters:
  - `message` - of `String` kind - defines text of a message sent by an application.
  - `origin` - of `String` kind - refers to a web client primary address. String includes protocol, domain and port.

  This is an optional parameter.
- `onFormOpen` - of `Function` kind - is a function which is called as soon as application form is opened (`onOpening` event handler operation is terminated). `onFormOpen` has the following parameters:
  - `url` - of `String` kind - is URL of a form being opened. This string is empty, if a form being opened has no URL.
  - `formName` - of `String` kind - defines the name of a form being opened which is generated based on the English version of 1C:Enterprise language.

  This is an optional parameter.
- `onFormClose` - of `Function` kind - is a function which is called as soon as application form is closed (`onClosing` event handler operation is terminated). `onFormClose` has the following parameters:
  - `url` - of `String` kind - is URL of a form being opened. This string is empty, if a form being opened has no URL.
  - `formName` - of `String` kind - defines the name of a form being opened which is generated based on the English version of 1C:Enterprise language.

  This is an optional parameter.

18.4. Implementation example

This section describes an example, where a web client is embedded in a web page. It is provided for reference only to show all embodiment capabilities. It is not intended to pursue a specific task.
It demonstrates the following:

- The way an element to be used by a web client has to be specified in.
- Minimum actions to be performed on the web site side, so that a web client can be started and interact with an external web site.

- Minimum actions to be performed on the application side for it to be able to operate inside and interact with an external web site.

To follow this example, you need to have a web server with a demo web page and web client. An application used for demonstration purposes consists of a single data processor defined as home page work area form. In this data processor form, create `MessageToExternalSite` command. It is available in the data processor form (`MessageToExternalSite` button). A form module specifies the following code in 1C:Enterprise language:

```
&OnClient
MessageToExternalSite(Command) procedure
  If ExternalSiteWindow.Available,
    MessageToSite = New MessageToExternalSite("Message from Web
Client");
    ExternalSiteWindow.SendMessage(MessageToSite);
  EndIf;
EndProcedure

&OnClient
Procedure OnOpen(Cancel)
  If ExternalSiteWindow.Available,
    EventHandler = New
NotificationDetails("OnReceiptMessageFromExternalSite", ThisObject);
    ExternalSiteWindow.EnableMessageHandler(EventHandler);
  EndIf;
EndProcedure

&OnClient
OnClosing(EndOfOperation) procedure
  If ExternalSiteWindow.Available,
    EventHandler = New
NotificationDetails("OnReceiptMessageFromExternalSite", ThisObject);
    ExternalSiteWindow.DisableMessageHandler(EventHandler);
  EndIf;
EndProcedure

&OnClient
OnReceiptOfMessageFromExternalSite(Message, AdditionalParameters)
Export procedure
  Notify(Message.Data);
EndProcedure
```

An application must be published on the web server. Save the publication address to further insert on the demo web page.

An application is integrated with a web page being as follows:

```
<!DOCTYPE html>
<html>

<head>
  <meta charset="utf-8" />
  <script
src="%WebSiteURL%/%InfobaseName%/scripts/webclient1ce.js"></script>
</head>

<body onload="init();">
  <div id="webClientContainer"></div>
  <button onclick="messageToWebClient();">MessageToWebClient</button>
  <div id="webClientMessageArea"></div>
  <script>
    var webClient = null;
    var onWebClientMessage = function (message, origin) {
      if (origin === '%WebSiteURL%') {
         document.querySelector('#webClientMessageArea').innerText =
message;
       }
      };

    var messageToWebClient = function () {
      webClient.postMessage("It's message sent by an external site");
    };

    var init = function () {
      webClient = new WebClient1CE('webClientContainer',
      {
        webClientURL:
'%WebSiteURL%/%InfobaseName%?MainWindowsMode=EmbeddedWorkplace',
        width: '600px',
        height: '400px',
        events:
        {
          onMessage: onWebClientMessage
        }
      });
    };
  </script>
</body>

</html>
```

For this web page to be operable, do the following:

- Replace `%WebSiteURL%` with URL of your web site (protocol, domain and port) where a web client is published.
- Replace `%InfobaseName%` with the name of your infobase being published (as specified in Name field of web server publication dialog box).

Specifically, if a web client is published at http://mysite.org/dbName, replace pseudovariables as follows:

- %WebSiteURL% is replaced with http://mysite.org.
- %InfobaseName% is replaced with dbName.

Let's take a closer look at operations performed on this web page.

- <script> loads a script implementing Embedded WebClient API.
- As soon as a web page body is loaded, init() handler is triggered, which is defined as onload event handler for body element.
- line defines element, where a web client is embedded.
- init() handler creates WebClient1CE object. A wizard is sent element ID (webClientContainer in the above example). Moreover, so far as init() handler is concerned:
  - A web client receives a startup command line where it is specified that a web client has to be started in Embedded Workplace main window mode.
  - For <iframe> element where a web client is expected to operate, width of 600 px and height of 400 px is set. These parameters are optional. Instead, either size by default or any other method of display area size assignment can be used for <iframe> (for example, CSS).
  - onMessageevent handler is registered to accept messages from the web client. This event will be handled in onWebClientMessage() (see above).
- <button onclick="messageToWebClient();">MessageToWebClient</button> line creates a button on a web page, which once pressed brings a web client to send a message with the following text: It's a message from an external website. The said message is sent from messageToWebClient() handler connected to onclick event of that button.
- line describes an area, where a web client message is displayed (from onMessage event handler).

Numbers refer to the following:

1. Whenever Message to an external website is pressed, Message from a web client is displayed below Message to a web client.
2. Whenever Message to a web client is pressed, It's a message from an external website is displayed in the web client messages panel.

Web client operates in iframe refers to data processor form representation where text is defined in 1C:Enterprise language. In your header instead of this text a message will be displayed which is defined in the form of your infobase.
`if (origin === '%WebSIteURL%')` condition is generally required to be present on a page, if one handler is used for several applications embedded in a single page. As such, it is needed to identify a web client sending a message.
Web client startup command line defined upon `webClientURL` property initialization: `webClientURL: '%WebSiteURL%/%InfobaseName%?MainWindowsMode=EmbeddedWorkplace',` can specify authentication parameters. For instance, if a user runs authentication on a web page (or web site in general), and a web page can unambiguously convert the website user name into username and password for an application. As such, a web client startup line has to be placed in a variable which needs to be generated before a web client embedded in a web page is started.
In conclusion, it should be noted as follows. When you load a web client, screen saver and loading progress bar are displayed. If a web client is embedded in a web page, screen saver and loading progress bar are displayed in the similar container, where a web client is embedded. If there is no need to do this, you can manage visibility of a web page area using `visibility` CSS property. Thus, a line intended to define a container for a web client on a web page will be as follows:

```
<div id="webClientContainer" style="visibility: hidden"></div>
```

To make a container visible, register a function to process `onStart` event received from the web client:

```
events:
{
  onStart: onStartWebClient,
  onMessage: onWebClientMessage
}
```

Further, write this function (next to `onWebClientMessage()`):

```
var onStartWebClient = function ()
{
  document.querySelector('#webClientContainer').style.visibility =
"visible";
};
```

As soon as the said modifications are made, when a web client is loaded, a web page displays no information. When loading is completed, a web page displays an application interface immediately.

Chapter 19. The functionality of jobs

19.1. General information

The functionality of jobs is intended for executing an application functionality according to the schedule or asynchronously.
The functionality of jobs solves the following tasks:

- capability of determining the rules of procedures at the configuration stage.
- performing the specified actions as per schedule;
- calling the specified procedure or function asynchronously, i.e., without waiting for it to complete;
- tracking the execution of a definite job and receiving its completion status (the value defining its successful or unsuccessful execution);
- receiving a list of current jobs;
- capability of waiting for completion of one or more jobs;
- managing jobs (capability of cancel, blocking of execution etc.).

The functionality of jobs consists of the following components:

- meta data of scheduled jobs,
- scheduled jobs.
- background jobs,
- job planner.

Scheduled jobsare intended for performing application jobs according to the schedule. Scheduled jobs are kept in the infobase and are created on the base of meta data, defined in the configuration. The schedule job meta data contains such information as name, method, usage etc.
The scheduled job has a **schedule**, which defines the time when the method related to the scheduled job must be executed. As a rule, the schedule is defined in the infobase, but it also can be defined at the configuration stage (for example, for pre-defined scheduled jobs).
Background jobs are intended for executing application jobs asynchronically and are implemented by using the 1C:Enterprise language.
Job planner is used for planning execution of scheduled jobs. The planner makes a check of each scheduled job on a periodical basis, if the current date and time match the schedule of the scheduled job. If they match, the planner assigns this job for execution. To do this, the planner creates a background job for performing the actual processing, for this scheduled job.

19.2. Background jobs

19.2.1. General information

Background jobs can be conveniently used for executing complicated calculation, when the calculation result can be received in quite a long time. The mechanism of jobs has features for executing such calculations asynchronically.
The background job is related to the method, which is called while executing the background job. The background job method can be presented by any procedure or function of the non-global general module; this procedure can be called on the server. The parameters of the background job can be presented by any values that are allowed for passing to the server. The parameters of the background job should fully match the parameters of the procedure or function it calls. The parameters of the background job can't exceed the size of 1 Gb, in the serialized form. If the size of the serialized parameters of the background job exceeds the value of 1 Gb, -an exception is generated. If the background job method is presented by a function, its return value is ignored.
The background job can have a **key** - any applicative value. The key brings restriction to the startup of background jobs -only one background job can be executed in a time unit, with a definite key value and the defined name of the background job method (the method name consists of a module name and a procedure or function name). The key allows to group the background jobs that have the same methods, according to a definite applicative flag with the purpose that no more than one background job is executed within one group.
Creation and control of background jobs is executed on the program base from any join. Creating a background job is permitted for any user. It is executed on behalf of the user who created it, at that. Receiving jobs, as well as expecting their completion is allowed from any join for a user with administrative rights or for a user who created these background jobs.
A background job is a purely session object, but it does not belong to any user session. A specialized session is created for each job, which is executed on behalf of the user who performed the call. The background job do not have a savable state.
The background job can give birth to other background jobs. In the client/server option, this allows to thread complicated calculations on the working process of the cluster, that can considerably increase the speed of the calculation process on the whole. Threading is implemented by the consequence of a few subordinate background jobs with expecting completion of each of them in the basic background job.
The background job has a capability of placing the data in the temporary storage of the calling session. For example, it can be used for passing the generated report

to the calling session, or the data prepared for processing. The data transfer from the calling session into the background job session is not possible.
The startup of a background job is executed in the following way:

- The following service actions are executed:
  - The fact of the background job startup is committed the event log;
  - The parameters of connection with external data sources are set;
  - An interface mode is set, etc.
- The event handler `SessionParametersSetting` is executed in the session module;
- The background job itself is executed.

The background jobs finished successfully or with failure are kept during twenty-four hours, then they are deleted. The history of background and scheduled jobs has the following features:

- The history is kept for each infobase;
- The execution history is kept:
  - Background jobs: - no more than 1000 jobs.
  - Scheduled jobs - no more than 1000 jobs. Along with that, the application tries to keep not less then 3 last startups for each different scheduled job. When the general restriction of 1000 startups is exceeded, the application will try to provide keeping the 2 latest startups for each scheduled job, and then 1.
  - System background jobs - no more than 1000 startups. In this section the background jobs are discussed, which are initiated for execution of some actions (execution of reports, search etc.).

19.2.2. Handling background jobs from the 1C:Enterprise language

NOTE. The examples provided in this section are not considered finished. They are intended only for demonstrating the ways of processing background jobs.

Controlling background jobs is performed with usage of the background jobs manager (`BackgroundJobs`). The manage of background jobs provides a capability:

- To execute a background job with parameter transfer (method `BackgroundJobs.Execute()`). After a background job is started, the object is returned, which describes the launched background job (`BackgroundJob`).

- To receive a list of background jobs matching a filter (the method `BackgroundJobs.GetBackgroundJobs()`) or to find a background job according to the UUID of the background job (`BackgroundJobs.FindByUUID()`).
- Expecting termination of one (`BackgroundJob.WaitForExecutionCompletion()`) or a few (method `BackgroundJobs.WaitForExecutionCompletion()`) background jobs earlier started with specifying the waiting time.

Let's consider all the stages in more detail.

If you want to launch a background job, you need an exportable procedure or function described in the non-global overall module, which is available in the server. The parameters of the called method should be listed in the specialized array of parameters. The order of parameters in the method should fully match the order of the array items: The leftmost parameter corresponds to the array item with the index 0. All the values of the background job parameters must support serialization. Then the key and the description of the started background job is passed. The key is used for ensuring the uniqueness of the started background job.

The overall module export method with the name of `UnloadData()`, located in the server overall module `OverallModulBackground`, will be used as a background job:

```
Function UnloadData(DataDescription, PathToFile) Export
  // Unloading data is performed here
EndFunction
```

In order to start a background job, you should perform the following:

```
CallParameters = NewArray;
Call.Parameters.Add(UnloadedData);
CallParameters.Add("c:\temp\data.dat");
UnloadJob =
BackgroundJobs.Execute("OverallModuleBackground.UnloadData",
CallParameters, "DataUnload", "Data unload in a disk file");
```

The parameter `DataDescription` will be presented by the variable value `UnloadedData`, the parameter value `PathToFile` will be presented by the row c:\temp\data.dat. The uniqueness key `DataUnload` and a description will be set for the started background job.

At the start, expecting the background job termination is not executed. Instead of this, after creation of a specialized session of the background job and start of execution of the background job method, the control will return to the called

code. The return method value `BackgroundJobs.Execute()` will be presented by the object, describing the started background job (type `BackgroundJob`). This object will be placed in the variable `UnloadJob`.
The background job description will allow to receive the ID of the background job, which will be used for searching the background job (property `UUID` of the object `BackgroundJob`). When you check the background job status (with usage of the background job property `Status`), you should remember that this status is docked for the object `BackgroundJob` creation moment, and is not refreshed afterwords. In order to check the status, you should receive a description of the background job being checked, before each check:

```
UnloadJob = BackgroundJobs.FindByUUID(ID);
```

If needed, the particular item of the background job can be interrupted by using the method `BackgroundJob.Cancel()`. You should remember that when the background job is interrupted, you need to implement "garbage collection". For example, you should delete temporary files that could be caused by the interruptible background job.
After the background job is started, you may need to check the job execution status on a periodical basis. The method `BackgroundJob.WaitForExecutionCompletion()`. The method parameter is presented by the time, during which the background job status change is expected. In any case, the method terminates not later than the timeout specified, and it generates an object `BackgroundJob`, which describes the current status of the background job (including the property `Status`).
Thus, expectation of termination of the background job may be executed as follows:

```
While True Do
  UnloadJob = UnloadJob.WaitForExecutionCompletion(10);
  If UnloadTask.Status = BackgroundJobState.Active then
    // continue waiting for termination
    Continue;
  Else
    // the background job completed
    Break;
  EndIf;
EndDo;
```

In the example the timeout is put to 10 seconds (the method `WaitForExecutionCompletion()` parameter). If the background job does not terminate with 10 seconds, -the do of termination expectation will continue. If the background job stopped (stopped being active), the particular termination reason will be clear after the property `BackgroundJob.Status` analysis.

It should be noted that expectation of termination is possible for one background job only. The background job manager has an analogue method (`BackgroundJobs.WaitForExecutionCompletion()`). The difference consists in the fact that the method parameter is presented by an array of objects `BackgroundJob`, their completion should be tracked.
The background job can generate messages to the users during its operation (by using the object `UserMessage`). Such messages may be received by using the method `BackgroundJob.GetUserMessages()`. Messages can be received during the background job operation and after the job is completed.

19.2.3. The background job and extensions

When the background job has started, the extensions that were attached in the parent session will be attached in the background job session. At the same time, there is a capability to start a background task so that the background job session has all the extensions attached, which are registered for the current infobase and the current separator set. To do this, you should use the method `ConfigurationExtensions.ExecuteBackgroundJobWithDatabaseExtensions()`. However, some scenarios may require starting a background job without any configuration extension. Then, you should use the method `ConfigurationExtensions.ExecuteBackgroundJobWithoutExtensions()`.

19.3. Scheduled jobs

Scheduled jobs are used when you need to execute particular periodic are one-time actions according to the schedule.
Scheduled jobs are kept in the infobase and created on the base of the schedule job meta data, defined in the configuration. The meta data set such scheduled job parameters as the called method, name, key, using capability, predetermination flag etc. When you create a scheduled job you may additionally specify the schedule (it can be set in meta data), the method parameter values, the user name for executing the scheduled job, etc.
Creation and control of scheduled jobs is performed with using of the 1C:Enterprise language from any connection, and is permitted only for users having administrative rights.

NOTE. When operating in the file mode, creating and editing scheduled jobs is possible without starting the job planner.

The method, which is called when a scheduled job is executed, is related to the scheduled job. The scheduled job method can be presented by any procedure or function of the non-global general module; this procedure can be called on the server. The parameters of the scheduled job can be presented by any values that are allowed for passing to the server. The parameters of the scheduled job should fully match the parameters of the procedure or function it calls. If the scheduled job method is presented by a function, its return value is ignored.

The scheduled job can have a **key** - any applicative value. The key brings restrictions to the start of scheduled jobs, as only one scheduled job with a definite key value can be executed at one time, among scheduled jobs related to the same meta data object. The key allows to group the scheduled jobs that are related to the same meta data object, according to a definite applicative flag with the purpose that no more than one scheduled job is executed within one group.

While configuring, you can define **pre-defined** scheduled jobs. Pre-defined scheduled jobs do not differ from ordinary scheduled jobs, excepting that they cannot be explicitly created and deleted. If a pre-defined scheduled job flag is set in the meta data of the scheduled job, then a pre-defined scheduled job will be created automatically in the infobase at the configuration upgrade. If the pre-defined flag is set off, then a pre-defined scheduled job will be created automatically in the infobase at the configuration upgrade. The initial values of the pre-defined scheduled job properties (for example, the schedule) are set in the meta data. Further, when the application operates, you can change them by the corresponding 1C:Enterprise language methods. Changes of the properties of the pre-defined scheduled job that has been already created in the meta data, will not influence the actual parameters of this job in the infobase. Pre-defined scheduled jobs do not have parameters.

The schedule of the scheduled job defines, what times the scheduled job must be started at. The schedule allows:

- to set the date and the time of the beginning and the end of the job execution;
- execution period;
- weekdays and months when the scheduled job must be executed, etc. (see description of the 1C:Enterprise language).

Examples of schedules of scheduled jobs:

Schedule	Parameter values
Every hour, one day only	`DaysRepeatPeriod` = 0 `RepeatPeriodInDay` = 3600

Schedule	Parameter values
Every day once a day	DaysRepeatPeriod = 1 RepeatPeriodInDay = 0
One day, one time	DaysRepeatPeriod = 0
Every other day, once a day	DaysRepeatPeriod = 2
Every hour 01.00 - 07.00 every day	DaysRepeatPeriod = 1 RepeatPeriodInDay = 3600 BeginTime = 01.00 EndTime = 07.00
Every Saturday and Sunday at 9.00	DaysRepeatPeriod = 1 WeekDays = 6, 7 BeginTime = 09.00
Every day one week, one week off	DaysRepeatPeriod = 1 WeeksPeriod = 2
At 01.00 once	BeginTime = 01.00
Last date of each month at 9.00	DaysRepeatPeriod = 1 DayInMonth = -1 BeginTime = 09.00
The fifth date of each month at 9.00	DaysRepeatPeriod = 1 DayInMonth = 5 BeginTime = 09.00
Second Wednesday of each month at 9.00	DaysRepeatPeriod = 1 WeekDayInMonth = 2 WeekDays = 3 BeginTime = 09.00

You can check if the job is executed for the given date (method ExecutionRequired() object Schedule of scheduled jobs). Scheduled jobs are always executed on behalf of the definite user. If the user of the scheduled job is not specified, execution is performed with the rights, defined by a set of roles, specified in the configuration property DefaultRoles. If no role is specified in this property, execution is performed without access rights restriction.

Execution of scheduled jobs takes place with usage of background jobs. When the planner finds out that the scheduled job must be started, then a background job that performs all the further processing is created automatically on the basis of this scheduled job. If this scheduled job is already being executed, it will no be started repeatedly, regardless of its schedule.

Scheduled jobs have a capability of restarting. It is especially vital, when the scheduled job method must be executed with a guarantee. Restart of a scheduled

job is performed in the case, if it was finished with a failure, or when the working process (in the client/server mode) or the client process (in the file mode), on which the scheduled job was performed, was finished with a failure. You can specify in the scheduled job, how many times it should be restarted (the property Repeat numbers at a crash), and also the interval between restarts (the property Repeat interval at a crash). After the specified number of restarts is completed, attempts to restart stop till the next time of a scheduled job start comes (according to the schedule). At the same time, the counter of repeat numbers is reset, and if a crash of the scheduled job occurs, the restart process starts again.
When you implement the method of a restarted scheduled job, you should have in mind that its execution will start from the start at the restart, rather then will continue from the point of the crash.

19.4. Features of execution of background jobs in the file and client/server modes

Execution mechanisms of background jobs differ in file and client/server modes.

19.4.1.1. File mode

Background and scheduled jobs are executed by client applications or web server extensions. Background jobs are executed in the client application that initiated the start of the background job. Background jobs are executed sequentially, because two background jobs cannot be executed simultaneously in one client application. When operating with the web server, consecutive execution of background and scheduled jobs is available for each infobase with access via this web server.
Behaviour of background and scheduled jobs in the file mode has the following features:

- Information concerning background jobs, called with the language methods or performing reports, is available only in the client application (or web-server extension) that executed it. The information is not saved after the completion of the client application operation (or web-server extension).
- The information of background jobs initiated by scheduled jobs is available in all the client applications, and is saved between starts.
- Scheduled jobs are executed by one client application only. You can prohibit startup of scheduled jobs by a definite client application or indicate that a particular client application is forcedly assigned as performer of scheduled jobs. Such a control can be performed by using:

  - The command-line parameter `/AllowExecuteScheduledJobs` for thick and thin client applications.
  - The attribute `allowexecutescheduledjobs` of the item `point` of the publication file default.vrd in case of publication of an infobase on the web server.
- Scheduled jobs are executed by the first client application according to the start order, for which execution of scheduled jobs is not prohibited. When this client application session is completed, execution proceeds to any of the client applications, which remain started. If a client application is started with clear indication of the need to execute scheduled jobs, then scheduled jobs are started by this client application, regardless of other applications being available (including web server extensions).
- Scheduled jobs are performed by the web server extension as long as the web server extension serves at least one client session.
- If the schedule of the scheduled job is set so that no client application operates at that point in time -such background job will never be performed.
- Scheduled jobs are processed once in 60 seconds.
- Initializing the planner of background jobs is performed in 60 seconds after the start of a client application or after an attempt to start the first background job. The following operations are performed before the planner of background jobs is started in the synchronous mode: searching in the dynamic list, making reports and search for entering in a row.
- Initializing the functionality of the scheduled job planner is performed in 60 seconds after the functionality of the background job planner is started. Therefore, scheduled jobs of the application start executing in the maximum time of 120 seconds after the start of the client application.
- Force closing of the background jobs is performed, which did not close within 30 seconds after completion of the client application. In this case, the event background job is committed in the event log. Force closing (`_$Job$_.Terminate`).
- Scheduled jobs are not supported in the file mode of the infobase in the session that was received using the automation-server 1C:Enterprise or COM-connection. If background jobs are created in such a session, they are placed in a queue, but startup of such jobs will not be performed.

19.4.1.2. Client/server mode

The **Job planner** that is located in the cluster manager is used in the client/server mode for performing background jobs. The planner receives a working process that is least loaded for all the background jobs in the queue for execution, and uses it for executing the related background job. The working process executes the job and notifies the planner of the execution results.

The client/server mode provides a capability to block execution of scheduled jobs. Blocking execution of scheduled jobs occurs in the following cases:

- Explicit blocking is set for scheduled jobs for the infobase. Blocking can be set by the cluster console.
- Blocking of the session starts is set for the infobase. Blocking can be set by the cluster console.
- The method `SetExclusiveMode()` with the parameter `True` is called for the 1C:Enterprise language.
- In some other cases (for example, when the database configuration is updated).

19.5. Creation of scheduled job meta data

Before a scheduled job can be created in the infobase by using the 1C:Enterprise language, you should create a configuration object for it.
In order to create the configuration object of a scheduled job in the thread Overall of the configuration tree for the thread Scheduled jobs, you should execute the command Add.
The scheduled job has a number of properties described below.
The method name - The method name of the scheduled job is specified.
Key - Arbitrary string value is specified, which will be used as the key of the scheduled job.
Schedule - The schedule of the scheduled job is specified. In order to generate the schedule, you should go to the link Open and set the needed values in the form that opened.
The bookmark Overall should contain the start date and the end date of the job and the repeat mode.

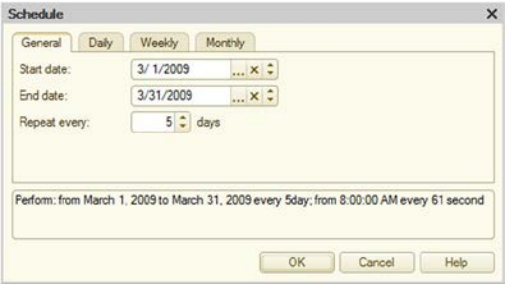


Fig. 281. Overall schedule

The daily job schedule is indicated on the bookmark Daily.

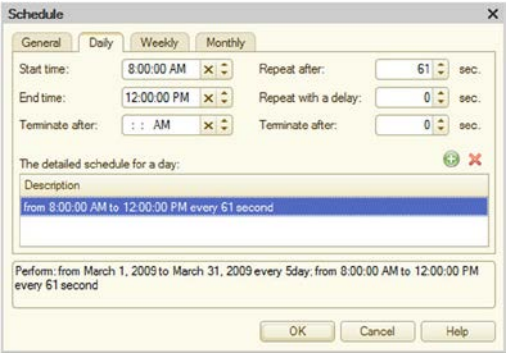


Fig. 282. Daily schedule

You can specify in the schedule:

- The start time and the end time of the job;
- the end time of the job, after which it will be forcedly closed;
- the repeat period of the job;
- duration of the pause between repeats;
- duration of execution.

You can combine these conditions.

You should indicate the weekly job schedule on the bookmark weekly.

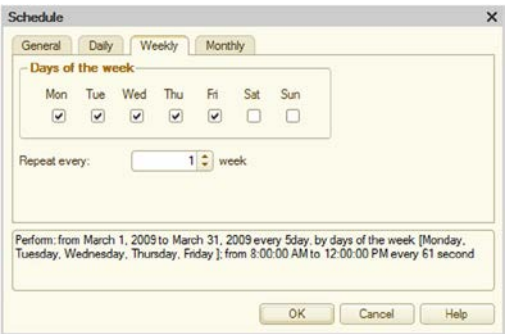


Fig. 283. Weekly schedule

You should set check boxes on the week days, on which the job will be executed. If you need to repeat the job, indicate the repeat interval in weeks. For example, the job will be executed in 2 weeks, the repeat value - 2.
If the case if a weekly schedule is set, and the property Repeat every value is more than 1, the property First weekday, set in the regional customization of the infobase, will be considered at determining the week borders.
The monthly job schedule is indicated on the bookmark Monthly.

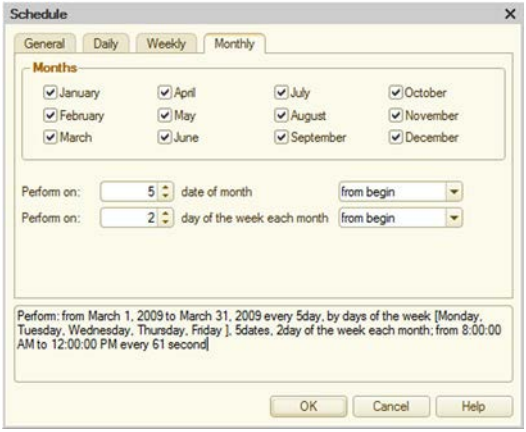


Fig. 284. Monthly schedule

You should set check boxes on the months when the job should be performed. If you need, you may specify the particular day (of the month or the week) of execution from the month/week start or end.
Usage - If it is set so, the job will be executed according to the schedule.
Pre-defined - If it is defined, the job is considered pre-defined.
The repeat number at the crash- indicates the number of repeats at the crash.
The repeat interval at the crash- indicates the interval of repeats at the crash.

An example of creation of a background job "Renewal of the full-text search index":

```
BackgroundJobs.Execute("RenewalFullTextSearchIndex");
```

An example of creation of a scheduled job "Restore of sequences":

```
Schedule = New JobSchedule;
Schedule.DaysRepeatPeriod = 1;
Schedule.RepeatPeriodInDay = 0
Job = ScheduledJobs.CreateScheduledJob("RestoreOfSequences");
Job.Schedule = Schedule;
Job.Record();
```

Chapter 20. Functionality of full-text search in the data

20.1. Overall information

Functionality of full-text search in the data of the 1C:Enterprise allows to perform search in the data base with specification of search operators (`AND`, `Or`, `NOT`, `NEAR` and other (`#Hashtag` lines)).
Functionality of full-text search is based on using two components:

- full-text index, which is created in the database and then is upgraded periodically, when needed;
- Means for performing full-text search.

Objects of full-text search are the data of the following configuration objects:

- Exchange plans
- Catalogs
- Documents
- Charts of characteristic types
- Charts of accounts
- Charts of calculation types
- Information registers
- Accumulation registers
- accounting registers
- Calculation registers
- Business processes
- Tasks

The property Full-text search is implemented for each of the listed configuration objects; it allows to include and exclude object data in/from full-text indexing.
1C:Enterprise has an event log for fixing changes of objects of the full-text search. Records to this log are performed during recording objects into the data base. Only objects adjusted for full-text indexing are sent to files of changes registration. If canceling a transaction will result in the fact that the object record in the base is canceled, the record in the event log will remain.
Full-text indexing is executed in the privileged mode (in the server context) and does not require an exclusive capture of the data base. In the process of full-text

indexing, reading of changes registration files and receiving of changed objects from the data base are performed, as well as words transliteration and replacing of letters in words input in the Roman alphabet into the Cyrillic one (the index keeps both word forms). Only attributes of the following types are to be indexed:

- row,
- date
- number
- link types,
- value storage.

Each object and attribute allow to add the following, to the full-text index:

- meta data object name or its attributes;
- synonym of meta data object or its attribute (in all configuration languages);
- presentation of the meta data object (in all the configuration languages).

Standard and user attributes are indexed in all languages, that allows to search in all the configuration languages (for example, in Russian and in English).

NOTE. At the full-text indexing and searching for a word with letter "ё", the letters "ё" are replaced by "е".

The disk ITS contains an article where you can read about the location of full-text search index files (http://its.1c.ru/db/metod8dev#content:1591:hdoc).

When full-text indexing is performed, the main index is created, and in the process of further changes of the database data, - the additional index is created, which contains all the information of the data changed after the latest upgrade of the main index.

Searching with the main index is performed effectively, while searching with the additional index is performed more slowly. Therefore, the indexation process allows the indexes merger. The merger allows adding the result of the last changes to the main index. You should note that this operation may turn out to be rather long (if the main index has a big size), that is why it is recommended to execute it in the periods of the minimum load (at night or on weekends).

Full-text search is initiated by the user in the client application context, and it is executed in the server context. It means that searching will be performed in the client computer in the file mode, and in the client/server mode -, in the server cluster of the 1C:Enterprise.

Full-text search in the data is performed according to the user rights (including, in accordance with the user rights restriction at the record level and database fields). Searching can be for words that were input into the database with mistakes: for example, if the English letter "c" is put in the word русский instead of the Russian letter "с", if the keyboard layout was accidentally changed while typing a word, what resulted in the word "системf" instead of "система", such

words will still be included in the search results. You should also remember that the functionality of full-text search in the data is performed from the word beginning. Te search string shall not contain more than 1,000 words (excluding the logical and special search operators). If the number of words in the search string exceeds the specified limit -no search is performed and exclusion is formed.
if a hash tag is specified in a search (for instance, `#Order`), all objects with their indexed data containing the said hash tag will be found. If you need to find all objects with hash tags starting from a certain sequence of symbols, specify `*` after that sequence. Specifically, if you need to find all hash tags starting from `Order`, specify `#Order*` in the search string. As such, `#Order`, `#OrderExternal`, `#OrderInternal` etc. will be found.
Search results are returned in parts, their sizes are determined at the execution of the full-text search command.
Ranging the received results is performed according to the following priorities:

- "weight" of the meta data object: the more links to the object the attributes of other objects have, the higher is its "weight";
- the object data (new objects will be in the beginning).

20.2. Mechanism usage

Full-text search in the data is performed by means of the 1C:Enterprise language. The global context property `FullTextSearch` is returned by the full-text search manager of -the object of the type `FullTextSearchManager`.
Full-text search manager methods allow:

- to receive information about the full-text index status;
- to perform full-text indexing;
- to initiate the process of full-text search in the data.

The platform provides a software interface to manage the full-text search:

- `GetFullTextSearchMode`- returns the value `True` if usage of full-text search is allowed, or otherwise `False`-.
- `UpdateDate()`- the data of the latest moment when all the data were indexed, and there was no new objects information for indexing.
- `IndexTrue()`- returns the value `True`, if the full-text search index fully matches the current status of the infobase.
- `IndexUpdateComplete()` -returns the value `True` if merging of full-text index is not needed.
- `SetMaxIndexedDataSize()` - allows to set the maximum data size (in bytes), kept in one configuration object attribute; these data index the

full-text search. The maximum size of the indexed data is equal to 1 Mb for the first call of this method.

- `GetMaxIndexedDataSize()` - returns the maximum size of data (in bytes), which is indexed by the full-text search.
- `SetIndexingJobsCount()` - allows to indicate how many background jobs will perform data indexing at the same time.
- `GetIndexingJobsCount()` - returns a number of background jobs that perform data indexing at the same time.
- `GetSeparationOfCompositeWordsMode()`/`SetSeparationOfCompositeWordsMode()` - allow getting and setting the meaningful word parts separation mode. The composite words mode change requires to rebuild a full-text search index.

The following methods are intended for performing full-text indexing:

- `SetFullTextSearchMode()` - sets the full-text search mode (`Allow` or `Prohibit`). If the search was prohibited, calling this method with the parameter `True` will automatically clear the full-text index available.
- `UpdateIndex()` - represhes the full-text search index. If indexes were unavailable, it performs full re-indexing of the whole data base. The indexing conditions are passed in the parameters of the method:
  - `EnableJoining` -if the value `True` is passed, merging of the main and additional indexes will be performed.
  - `Partial` -value `True` indicates that indexing will be performed in portions of 10 thousand objects. The process finishes after indexing the data of one portion. The indexing time of one portion considerably depends on the data.
- `ClearIndex()` - deletes all files of the full-text index. This method is recommended for using, when the data were updated fully or almost fully (for example, when the infobase load is executed). You may need to rebuild the index after clearing.

The initiating of the full-text search process in the data is performed with the method `CreateList()`. Two parameters are passed into the method:

- `SearchString` -the row containing the search expression;
- `PortionSize` - the number defining the number of objects that can be returned in one portion of the full-text search.

For a description of search expression syntax.

The method `CreateList()` returns the object `FullTextSearchList`, which allows to perform full-text search and receive results of full-text search. This object can be used repeatedly for performing search with different conditions. The properties of the object `SearchString` and `PortionSize` allow to change the used search expression and the portion size of the received data.

The method `FirstPart()` is used for executing full-text search and receiving the "first" results; it fills the list with the first found items, in accordance with the portion size. The methods `NextPart()` and `PreviousPart()` are used for receiving the further full-text search results. These methods can accept as the parameter the current initial position and fill the full-text search list with search results. If the parameter `CurrentStartPos` is not specified, the value of the initial position of the object `FullTextSearchList` is used; for example, this value can be received using the method `StartIndex()`. Usage of the parameter is more desirable, as it increase the speed of the full-text search.
The method `Quantity()` contains a quantity of items in the current portion (it will be less or equal to the portion size for the last portion), and the method `TotalCount()` contains the full quantity of items found as a result of full-text search.
The method `NextPart()` fills the list with the following items in accordance with the value of the portion size. Along with that, the current position will increase with the data quantity that the received portion contains. If there is no data for receiving another portion of data (the data end has been reached), an exception will be called, which can be processed by the instruction `Attempt...Exclusion..AttemptEnd`.
The method `PreviousPart()` fills the list with previously found items in accordance with the portion size. If there is no data for receiving another portion of data (the data beginning has been reached), an exception will be called, which can be processed by the instruction `Attempt...Exclusion..AttemptEnd`.
The method `TooManyResults()` returns the value `True`, if truncation of the results quantity has been performed during the search for performance reasons; this may influence the search accuracy (not all the objects will be found). Analysis of the value returned by this method should be executed at receiving the latest portion of the data found, for informing the user that not all the results in the database have been received.
The property `GetDescription`contains the flag that the description of the search results has been received. If it is set to the value `True`, the value `Description` will be filled for each of the search results, which helps to understand the context of the words found. At the same time, setting this property in the value `False` makes the seach quicker.
The full-text search list represents a collection of the full-text search list items, which can be iterated by using the operator `For each ... From ...Do`.
Each item of the full-text search list is presented by the object `FullTextSearchListItem` and contains the following properties:

- `Value` - identifies the data (object or record set), where the search expression was found;
- `Meta data` - objects of meta data that describes the data, where the search expression was found;
- `Presentation` -text presentation of the object found;
- `Description` - contains (with a new row) pairs `<attribute>:<value>`, where:
  - `<attribute>` - attribute of the object, in the value of which the search expression was found;
  - `<value>` - value of this attribute.

The search result with usage of the method `GetRepresentation()` can be presented as the object `ReadXML` or as a row with a text in the HTML format, where the words found are highlighted by the HTML means (in bold, the background color).

20.3. Using additional dictionaries

Additional dictionaries of morphology and synonyms for full-text search expand system dictionaries and can contain special terms and words that are used with configuration.

Additional dictionaries can be presented by templates of binary data and text templates, as well as constants of the string type and the type `ValueStorage`.

Indicating additional dictionaries is performed in the property Additional dictionaries of the root meta data object.

They should have the following contents:

```
<?xml version="1.0"?>
<Dictionary>
  <Words>
    <lemma>рунет</lemma><forms>рунета рунете рунетом</forms>
    <lemma>Ванесса</lemma><forms>Ванессе Ванессы Ванессу</forms>
  </Words>
  <Synonyms>
    <item>error bug crash</item>
    <item>stream thread</item>
  </Synonyms>
  <Roots>
    <item>bread</item>
    <item>house</item>
    <item>post</item>
  </Roots>
</Dictionary>
```

The item Dictionary

Intended for keeping the dictionary itself. The dictionary can contain two sections:

- additional words (lemmas) and their forms;
- rows of synonyms.

The item Words

Contains words in the following kind:

- In the item `lemma`, the word is kept in its basic form (the nominative case);
- In the item `forms`, case forms of the word are kept.

Fuzzy search is not performed by default. In order to perform fuzzy search, you should use the operator «*». Example: search руне* finds and рунет, and рунета, and рунете, and рунетом.

The item Synonyms

Sets of synonyms are kept. Each set is put in tags `item`.
To make synonyms participate in the full-text search, you should use the operator «!». At thatm the search !ошибка finds and ошибка, and bug, and сбой.
The items `Synonyms` and `Words` can be in any case.

Roots item

The glossary of meaningful word parts is stored. Each meaningful word part shall be included in `item`. For the languages different from Russian, the glossary of meaningful word parts will allow enabling the composite words separation and the search by the meaningful word parts.

The dictionaries are loaded at the first call of the search and at indexing. If the template contains errors, filling dictionaries will be stopped at the error spot.

If the additional dictionary changes, it is needed to perform restart of the system to ensure the new dictionary is attached by the system. Different components must be restarted depending on the system option:

- the file mode of the infobase with direct connection - you should reload all the client application attached to the infobase;
- the file mode with connection via the web server - you should restart the web server extension (actually, the web server need restart);
- client/server mode (with any connection type) - you need to restart the server 1C:Enterprise.

Along with that, the index is not updated automatically, and you have to rebuild the index manually, although new dictionaries are used at the search.

20.4. Mobile platform

Full-text search on the mobile platform has some features and restrictions.
Full-text search does not support:

- Search with morphology.
- Attach of additional dictionaries.
- Text extraction operation (object `TextExtraction`).

When using the full-text search on the mobile platform you should remember the following features:

- Update of the full-text search index is performed automatically after creation or change of the configuration object, for which usage of full-text search is set.
- In order to update the full-text search index, you should create a specialized background job. A background job is created in case the server call is not performed and no other background job is performed. Update is performed in portions of 10 000 objects.
- If the case all the infobase objects are indexed, the mobile platform starts merging of indexes after 20 seconds of inactivity.

If it is required to guarantee data indexing, for example, after synchronization of the infobase with the infobase on the PC, then you should explicitly perform the update of the full-text search index using the method `UpdateIndex(True,False)` of the full-text search manager.

Chapter 21. The temporary storage functionality, handling files and pictures

21.1. Temporary storage

1C:Enterprise has the functionality of temporary storage providing storage of some data attached to the session. Besides, the mechanism of handling files is implemented, which provides for exchange of files between the infobase and the client application. This mechanism feature consists in its orientation for using in the thin client and the web-client, and is developed with account of restrictions on files, constrained by web browsers.

Temporary storage - this is a specialized information storage, where the value can be placed. The main purpose is - temporary storage of information at the client/server interaction before it is transfered into the data base. The mechanism of temporary storage combined with the files functionality provides a set of methods, which allows to place the data locally kept at the user, into the temporary storage of the infobase, to transfer this information from the temporary storage into the database and get it back into the user computer. The most widespread application tasks resolved by this functionality are -keeping in-line information, for example, pictures of goods, related to document contracts etc. Mechanisms of temporary storage and those related to files are often used jointly, but can be used separately.

The temporary storage is necessary, for example, when the web-client needs to transfer a file from a client computer to an application server computer (or web-server extension for a file variant of information base). The file is placed in the temporary storage and later (in 1C:Enterprise system server application) may be used for the object recording to a database or for processing.

When you place data into the temporary storage, you should remember that the temporary storage can contain the data with types that do not support serialization, both directly and as components of any collections.

You can use the temporary storage as an universal storage for the controlled data lifetime:

- If you place data into the temporary storage and bound the data to some form, the lifetime of the saved data will depend on the lifetime of the form, to which the data are attached. When you delete the form object, the temporary storage will be cleaned of all the related information.
- If attaching data to the form was not completed at placing the data into a temporary storage, clearance of the temporary storage will be performed in the following cases:
  - At the following query of the form.
  - At the following server call from the overall client module.
  - At the context and non-context client calls from the form.
  - At the server call from the command module. If a server call is performed for placing the value into the temporary storage, clearance will not be performed. Clearing is performed after the call completes its functioning.

It means you can place one or a few values into the temporary storage, and use this value in the next call. Along with that, the replaced value will be automatically deleted after usage and before the server call is completed.

The most typical application task that can be solved by the temporary storage is - providing access to files and pictures before the object is recorded in the infobase, for example, in the item form. When you arrange such access, it is recommended to pass data from the client computers to the server as files, without preliminary "transformation" into a text document, spreadsheet document or similar formats of the 1C:Enterprise. It is recommended to pass the file to the server side, and on the server side, to load the file data into the required format 1C:Enterprise.

The data placed in the storage are identified by a unique address, which can further be used in operations of recording, reading and deleting. This address is given by the recording methods of values in the temporary storage. A separate method in the 1C:Enterprise language allows to determine, if the passed address is the address that indicates the data in the temporary storage.

The temporary storage generated in one session is unavailable from another session. The exception is the capability of data transfer from a background job into a session, which initiated this background job, using the temporary storage (see details on p. 1242).

21.2. Features of handling files in the asynchronous mode and in the web client

Handling files in the asynchronous mode is available on the side of the client application only. In the general case, you should use synchronous methods for work on the server side and asynchronous ones - for working on the client application side.

Let's consider an example of handling files in the asynchronous mode: deleting all files in the directory of temporary files. This action will look as the follows for the synchronous mode:

```
&OnClient
Procedure Delete (Command)

  DeleteFiles(TempFilesDir(), GetClientAllFilesMask());
  Report("Delete successful");

EndProcedure
```

Asynchronous mode will look as follows:

```
&OnClient
Procedure Delete (Command)

  CallBack = New
NotifyDescription("ReceiveCatalogTemporaryFilesCompletion", ThisObject
, "FileError", ThisObject);
  BeginGettingTempFilesDir(CallBack);

EndProcedure

&OnClient
Procedure
ReceiveCatalogTemporaryFilesCompletion(TemporaryFilesCatalogName,
AdditionalParameters) Export

  CallBack = New NotifyDescription("DeleteCompletion", ThisObject,
"FileError", ThisObject);
  BeginDeletingFiles(CallBack, TemporaryFilesCatalogName,
GetClientAllFilesMask());

EndProcedure

&OnClient
Procedure DeleteCompletion(AdditionalParameters) Export
  Report("Delete successful");
EndProcedure

&OnClient
Procedure FileError (ErrorInfo, StandardProcessing,
AdditionalParameters) Export
  Notify("When executing a file operaton an error is found: " +
BriefErrorDescription(ErrorInfo));
EndProcedure
```

You should pay attention to the fact that almost all file operations are asynchronous. Even such simple actions as receiving a directory of temporary files. The need to receive this directory will result in the first asynchronous call.

The call of deleting all files will take place then. The process of deletion completion requires the second handler, where actions must be located, that must occur after deletion of all files in the directory of temporary files. In this example, this is an error message about the operation completion.

If an error occurs during execution of file operations, - control will be passed to the handler `FileErrors()`. The situation can be processed correctly in it.

Handling files in the web client has some features:

1. If you don't install plug-ins for handling files, in any browser, you have the methods available: `GetFile()`, `BeginPutFile()` and `BeginPlacingFiles()`. Capabilites of parameterization of the methods `BeginPutFile()` and `BeginPuttingFiles()`.

2. The method `BeginPutFile()` (in the interactive mode) stops code execution in the 1C:Enterprise language in the thin and thick client, as opposed to the web client. The call of the callback procedure, passed to the`BeginPutFile()` method, will be performed after the method completion. The whole code that is to be executed after file location shall be located only in the call back handler to avoid differences in the behavior in the thin client and in the web client.
3. Only asynchronous file operation methods are available in the web client. This behavior is not affected by the application property Modality use mode.
4. Installation of the file system extension gives more options for parameterization and asynchronous file operation methods. However, options for the synchronous file operation in the web client are unavailable in any operation mode.
5. The file system extension does not support HTTPS operation using the client certificate.

NOTE 1. Microsoft Core XML Services (MSXML), vs 4.0 or 6.0 is recommended to be used for the correct operation of the file system extension in the web browser Microsoft Internet Explorer.
NOTE 2. The file system extension for the web browser Microsoft Internet Explorer shall be installed to the directory %APPDATA%\1C\1CEWebExt\FileSystemExtIE.

21.3. File and temporary storage operation methods

21.3.1. General information

All file operations performed in the client application (including the file transfer between the client and server and the work with the temporary storage) are asynchronous in any client application and any variant of operation.
It may be required to install an extension for work with files for the file transfer to the server from the web-client. In the general case, it can be stated that an extension will be required only when a file operation is not accompanied by the user interactive action, for example, -it will require an extension to place a file to a server with a fixed path specified in the code of the application solution. At the same time, a file specification with the help of file selection dialog -will require no extension for the work with files.

The amount of data transferred between the client and server shall not exceed 4 GB (in serialized form)) at a time. This restriction applies to one file (the methods `BeginPutFileToServer()` and `BeginPutFileToServer()`) or any other object put to the temporary storage from the client application (one call of the method `PutToTempStorage()`).

21.3.2. Migration of the file data to the temporary storage

21.3.2.1. Single file

Use the method `BeginPutFileToServer()` to migrate the single file to the temporary storage. Once the method is completed, the file will be migrated from the local file store to the temporary storage. Some auxiliary information can be specified when calling a method (depending on the method variant used). The following information can be specified:

- The address in the temporary storage where you need to save the file (`Address` parameter). If no parameter is specified then a new item, which address will be transferred to the file transfer to temporary storage completion handler, will be created. If the parameter is specified the file will be put at the specified address. In this case, the data stored at the transferred address as of the method call will be deleted.
- Description of a file to be put to the temporary storage. A file can be set interactively, in the file selection dialog; as a complete path to the file formed in the embedded language; as a special item formed when a file is put on the form items with the help of drag operation (`FileRef` object).
- Description of the dialog to be opened for the selection of the file put to temporary storage (`PuttingFilesDialogParameters` object).
- Form unique identifier. If this parameter is set then the platform will automatically delete all items of the temporary storage after the form with the specified CID is closed. Otherwise, the value will be deleted automatically after the next server call. If a CID which does not correspond to any form is specified as the parameter the item of the temporary storage will be deleted upon the client session completion. But it would be more correct to delete an unused value from embedded when this value is no longer used (by the global context method `DeleteFromTempStorage()`).

The method enables processing several events that emerge during its (the method's) operation:

- The fact of the selection of the file to be migrated to the temporary storage. The parameter of `BeforeStartPutFileNotifyDescription`

method is used for it. This notification will be called directly before the file is put to temporary storage. This handler has the option to retract the file migration if, for instance, the size of the migrated file is too big. If you retract the file migration in this handler, the store shall not call the final handler.

- A periodic event arising in the process of a file transfer to the server. For this, the parameter of `NotifyDescriptionOnPerformanceProces` method is used. The information on the transferred file and on what part of the file is transferred to the server will be transferred to this event. If the developer implemented a special form displaying the transfer progress it is possible to abort the file transfer to the server, for which it is necessary to set the developer's special parameter `RetractFileMigration` in the value `True`. If, during several files placement, the develope's parameter `RetractFileMigration` is set in the value `True` for all stored files it will be equal to setting the developer's parameter `RetractAllFileMigration` in the value `True`. In any of these cases, the value `Undefined` will be transferred to the file migration completion notification handler.
- End of file migration to the temporary storage. For this, the parameter of `NotifyDescriptionOnCompletion` method is used. This notification shall be called after migration of the file to the temporary storage is completed. The first parameter will contain the information on the file migration operation: success or failure, address in the temporal storage, and transferred file description.

Various types of values may be indicated as the description of the stored file as was said above:

- No file name is specified. No extension for the work with files is required in the web-client. In this case, a selection dialog which parameters are set by default will be offered to specify the file.
- Complete path to the stored file (the parameter name is `PathToFile`). An extension for the work with files is required in the web-client. In this case, an attempt to put the file the path to which is specified in the parameter to the temporary storage will be made.
- `FileRef` object (is the parameter of the same name). No extension for the work with files is required in the web-client. An object of such type can't be created in the embedded language, it can only be obtained in the result of interactive action. The file dragging from the operating system interface to any item of the client application form (including web-client) is such action.

- `FileSelectionDialogParameters` object (`DialogParameters` parameter). No extension for the work with files is required in the web-client. This method call variant is analogous to the first one, when no certain file is specified but is offered to be selected interactively. The described call variant differs with the ability to adapt the dialog to the real work scenarios: specify title, file selection mask.

The result of the file migration operation shall be passed to the handler of the final notification including the address of the temporary storage and the full name of the migrated file.

Below is an example of file migration to the temporary storage in which the file selection dialog is set and the process of the file transfer to the server is displayed. The example is for education only and is not a complete variant of any mechanism implementation:

```
&OnClient
MigrateFileCommand (Command) procedure

  DialogParameters=New PuttingFilesDialogParameters;
  DialogParameters.Title = "BeginPutFileToServer";
  DialogParameters.MultiSelect = True;
  DialogParameters.Filter = "Text file|*.txt|All files|*.*";
  CompletionCallback = New
NotificatinDescription("CompletionCallback", ThisObject);
  ProgressCallback = New NotificationDescription("ProgressCallback",
ThisObject);
  BeforeStartCallback = New
NotificationDescripption("BeforeStartCallback", ThisObject);
  BeginPutFileToServer(CompletionCallback,ProgressCallback,BeforeStart
Callback, ,DialogParameters);
EndProcedure

&OnClient
CompletionCallback(StoredFileDescription, AdditionalParameters)Export
procedure
  Report("File Migration"
+?(StoredFileDescription.FileMigrationCancelled, "cancelled",
"successful"));
  Report("Temporary storage address: " +
StoredFileDescription.Address);
  Report ("File Name: " +StoredFileDescription.FileRef.Name);
  Report("File size, byte: "+StoredFileDescription.FileRef.Size());
EndProcedure

&OnClient
Procedure ProgressCallback(StoredFile, Stored, RetractFileMigration,
AdditionalParameters) Export
  Status("File " + MigratedFile.Name, Stored, "File size " +
StoredFile.Size() + " byte", PictureLib.Document);
EndProcedure

&OnClient
Procedure BeforeStartCallback(MigratedFile, RetractFileMigration,
AdditionalParameters) Export
  Report("Migrated file - " +MigratedFile.Name + ", size = " +
MigratedFile,Size());
EndProcedure
```

You should know that indication of the filter list for parametrization of the dialog box does not make the user to select files only with these extensions or masks. Indication of files only facilitates selection of necessary files for the user.

21.3.2.2. Several files

Method operation general scheme `BeginFilePutToServer()` corresponds to the description of the method `BeginFilePutToServer()`. The distinction is that the following is used for the transfer of description of several files:

- Line with mask describing the migrated files on the client computer disk. File system extension is required in web client. All files corresponding to the search string will be transferred to the server in this case.
- Array of objects describing the migrated files. It is important to remember that the values of one type can be contained within the framework of one array call. The objects can be of the following types:
  - Type `TransferredFileDescription`. File system extension is required in web client. Each array element must contain a non-empty path to the file on the hard drive.
  - Type `FileRef`. No file system extension is required in web client. It can be used in the form to which different files are dragged from the client computer memory.
- File selection dialog parameters for the user to indicate the files to be transferred to the server. No file system extension is required in web client.

The number and description of event handlers correspond to the events from the method `BeginFilePutToServer()` with the only difference that an array of objects describing the processed files gets in each handler.
The event handler allows aborting the transfer of only all selected files to the server before the migration beginning. The process of the files transfer to the server is consecutive. The process notification event handler is called for each file. This handler allows aborting the transfer of any single file (with the help of the handler parameter `RetractFileMigration`) and the transfer of all files (with the help of the handler parameter `RetractAllFilesMigration`).

21.3.2.3. Memory data

`PutToTempStorage` method is used to temporary store the data represented as an existing object of some type in the core memory, not as a file on a disk drive. If no address of the temporary storage is given when the method is called, - the data shall be stored at the new address retrieved as a result of the function. The stored data must belong to some form and they are automatically deleted when the form is deleted or after the server call (depending on the method's parameters).
If `DeleteFromTempStorage()` method has been used for the address of the temporary storage that is a result of the operation of `PutToTempStorage()` method, the specified address may not be reused as `Address` parameter of `PutToTempStorage()` method. If such attempt is completed, exemption shall be generated.

IMPORTANT! The actual serialization is not made during migration to the temporary storage. The link to the value is located and stored in the cache for 20

minutes. At the end of this period the value is serialized, recorded to the hard drive (to the session data storage) and deleted from the cache.

21.3.3. Data retrieval from the infobase to the file

21.3.3.1. Single file

`BeginGetFileFromServer()` method retrieves the file from the infobase and saves it to the user's local file store. The method has several variants of use:

- The file address in the information system (navigation link) and the initial name of the file in which the data from the base is to be stored are specified. No extension for the work with files is required in the web-client. The user can specify any file name in the save dialog. The platform does not notify the configuration about how the file saving is completed.
- The handler of the notification about the operation completion, the file address in the information system (navigation link), and the full name of the file in which the data from the base is to be stored are specified. An extension for the work with files is required in the web-client. In this case, the user can't change the file name and the position of the obtained file in the client computer memory. After the operation completion, the platform will call the handler of the notification about completion, where the information is transferred, and how the operation is completed.

The user on whose behalf the file is saved must have the rights to `View` the properties of the database in which the data is stored.

21.3.3.2. Several files

The method `BeginFileGetFromServer()` allows getting and saving several files stored in the information base in the client computer local memory. The method has several variants of use:

- The array of descriptions of obtained files (the objects of the type `TransferredFileDescription`), file save dialog description, and the parameters of the archive in which the obtained files will be placed (if necessary) are transferred to the method. No extension for the work with files is required in the web-client. The platform does not notify the configuration about how the files saving is completed.
- The array of descriptions of obtained files (the objects of the type `TransferredFileDescription`), the startup directory in which the transferred files are supposed to be saved, and the parameters of the archive in which the obtained files will be placed (if necessary) are transferred to

the method. An extension for the work with files is required in the web-client. After the operation completion, the platform will call the completion notification handler to which it will transfer the result of the operation completion.

The parameters of the dialog which will be shown for the selection of the files location can be set when several files are got from the server. `FileGetDialogParameters` parameter of the same name is responsible for it. The value of the property `FileGetDialogParameters.DirectorySelection` determines how the directory will be selected:

- `True` -the directory will be selected once and all files will be placed in it. The parameter is ignored in the web client.
- `False` -the directory selection dialog will appear for each file got from the server.

The property `FileGetDialogParameters.Title` allows setting the title of the dialog for the selection of the directory in which the files will be saved and the property `FileGetDialogParameters.Directory` allows setting the initial value of the directory in which the files will be saved (it is ignored in web client).

-Automatic formation of an archive for files getting is another useful feature accessible when getting the files from the server. `FileArchiveGetParameters` parameter of the type of the same name is responsible for it. No archive is formed if no parameter is specified. If a parameter is specified (a value of the type `FileArchiveGetParameters`), the created archive is determined by the value of the property `FileArhiveGetParameters.Mode` and the presence of the extension for the work with files:

- The property `FileArchiveGetParameters.Mode` is set in the value `GetArchiveAlways`. In this case, the archive is formed always using the method `BeginFileGetFromServer()`.
- The property `FileArchiveGetParameters.Mode` is set in the value `GetArchiveIfNecessary`. In this case, an archive is formed only when several files are got with the help of the method `BeginFileGetFromServer()` and no extension for the work with files is installed.

The created archive name is determined with the help of the property `FileArchiveGetParameters.ArchiveName`.

In case of an archive formation, the required objects are saved in a temporary server then archived (by the platform tools) and the resulting archive is transferred to the client application.

The location of the files (both in the archive and in the recipient-directory) is determined by the property `TransferredFileDescription.Name`:

- A file name is set without a directory -the file will be located directly in the recipient-directory or in the root of the archive.
- A file name is set with the specification of path -the structure of the directories specified for the file will be created in the recipient-directory or archive.
- File absolute name is set -the file will be saved according to the specified path. The structure of directories corresponding to the full path to the file will be recreated in the archive.

21.3.4. Data retrieval from the temporary storage

If you save the object to the infobase, you may need to extract the data from the temporary storage and to save them, for instance, to the attribute of the infobase object. The special -`GetFromTempStorage()` method is used for it. This method extracts the data from the temporary storage and restores them as the output. You need to specify the address in the temporary storage to retrieve the data. This address restores methods of data migration to the temporary storage if successfully completed (see previous sections).

IMPORTANT! If you retrieve the value from the temporary storage on the server, please remember that it is retrieved at the link. This link actually specifies the value stored in the cache. The value is saved in the cache within 20 minutes since it is migrated in the storage or since the last call, then it is saved on the hard drive and deleted from the cache. During the next call the value is imported from the hard drive and migrated to the cache again.
Links are not restored after deserialization and recovery of the value from the temporary storage. The value in the cache is restored from the hard drive. However, it is impossible to recover links to other objects after serialization/deserialization.

21.3.5. Data removal from the temporary storage

After the data are stored in the attribute of the infobase object, the data may be removed from the temporary storage. For this, `DeleteFromTempStorage()` method is used. The method acquires the address in the temporary storage in the parameter.

21.3.6. Check whether the address is attributed to the temporary storage

The address may specify both the temporary storage and the attribute in the infobase. `IsTempStorageURL()` method is used to check its type.
It checks whether the passed address is the address specifying the storage. It returns `True` if the address specifies the temporary storage.

21.3.7. Retrieve the address of the attribute

After the data are migrated to the attribute of the infobase object, you may need to get the access to them using file methods.
However, before data retrieval, for example, from the attribute, you need to get the address of this attribute. `GetURL()` method is used for it.
It may return the address of the value in the infobase using initial parameters. For this you should pass the key of the object (it may be either the link to the object or the key of the information register record) and the name of the attribute. If you need to get the address of the value stored in the tabular section attribute you should add the name of the tabular section and the point «.» to the name of the attribute in the parameter that sets up the name of the attribute.

Example:

```
Goods.Image
```

21.3.8. Authorization for the group of procedures with files

You may need to get the authorization for several procedures with files to complete some procedures in the web client. For example, you need to complete retrieval of the document from the infobase and to open the saved document using the associated application.
You shall have to answer the question about storage of the document and the question about the launch to complete this procedure. If the number of procedure is more the user shall have to answer more questions too.
You may use `BeginRequestUserPermission()` method to decrease the number of questions. When this method is used, the list of all the procedures to

be completed is displayed for the user and it is suggested to authorize completion of the group of procedures. If the user has authorized completion, requested procedures shall be completed without additional requests to the user. If the authorization is not granted, procedures shall be completed in a standard mode: one request for one procedure.

In the thin and thick client applications, the method `BeginRequestUserPermission()` always returns the value `True` without any interaction with the user. An extension for the work with files is required in the web-client.

Let's point out several features of using `BeginRequestUserPermission()` method.

- The permission can be requested for the following methods:
  - `BegingFilePutToServer()`/`BegineFilePutToServer()`;
  - `BeginFileGetFromServer()`/`BeginFileGetFromServer()`;
  - `BeginRunningApplication()`;
  - `BeginDeletingFiles()`;
  - `BeginFindingFiles()`;
  - `BeginCopyingFile()/BeginMovingFile()`;
  - `BeginCreatingDirectory()`/`BeginCeratingTemporaryFile()`;
  - `BegingCreatingBinaryDataFromFile()`;
  - `FileStreams.BeginCreation()`;
  - `FileStrems.BeginOpening()`;
  - `FileStreams.BeginOpeningForReading()`;
  - `FileStreamsBeginOpeningForWriting()`;
  - `FileStreams.BeginOpeningForFiningWriting()`;
  - `BinaryData.BeginWriting()`.
- The authorization is requested for the specified set of the method's parameters. If values of parameters will be different from the values for which the authorization is granted during the actual execution of the file operation method, this authorization will not be valid and the user will get the individual request to confirm the procedure.
- If you need to complete two (or more) similar file procedures (even with the same parameter set) you should specify the relevant number of items in the parameter array of `BeginRequestUserPermission()` method. For example, you need to retrieve the same file from the infobase twice and to record it to the docked place of the file store, you should request the authorization for two procedures.

- If you request the authorization for the procedure for which the interactive procedure is needed (for instance, `BeginFileGetFromServer()` functions interactively) such procedure is removed from the request.

The acquired authorizations are saved either until the authorized call is completed or until 1C:Enterprise language is finalized.

21.3.9. Operating temporary storage in the background job

The feature for temporary storage operation has the option to pass the data from the background job to the session that has initiated the background job. To complete such migration, you need to record the blank value in the temporary storage in the parent session (using `PutToTempStorage()` method) after you have specified any ID of the generated temporary storage (`Address` parameter). Then pass the acquired address to the background job using the background job parameters. If you use this address as the value of `Address` parameter of the `PutToTempStorage()` method in the background job, the result will be copied to the session where the background job started.

The data recorded in the temporary storage in the background job shall not be available from the parent session until the background job is completed.

21.3.10. Access to standard directories

When you use the store you need some space in the file store where you can store various file data that are both temporary and -should be stored for a long time. Such files include draft for document management systems, add-ins operated by the client computer etc.

The special directory attached to the specific infobase user is used to store such files. The same user operating two infobases will have access to two different user data directories. The location of this directory is specified by `BeginGettingUserDataWorkDir()`/ `UserDataWorkDir()` method. If there is no directory, it is generated after the first call to it. If it is impossible to generate the directory -the system calls for exemption.

NOTE. Methods `BeginGettingUserDataWorkDir()`/`UserDataWorkDir()` method is unavailable from the server.

The operating system has the special directory for permanent user data storage. They may be some reports, print document forms etc. This directory stores the data that may be sent to external consumers. `BegingGettingDocumentDir()`/`DocumentsDir()` method is used to access this directory. The physical path of the directory depends on the operating system where the application is made and it is given in the Syntax Assistant.

21.4. Work with pictures

21.4.1. Parameters of figure and operation of conversion

AC:Enterprise platform allows getting some picture characteristics. To get picture characteristics, create `Picture` object using the relative wizard. The following properties of the resulting object can be obtained:

- Picture width and height. The size returns in the points in the relative coordinate. `Width()` and `Height()` methods are used.
- Color depth. For the raster pictures, the color depth in form of the value of the system enumeration `ColorDepth` is returned. `ColorDepth()` method is used.
- The flag that shows whether the picture is grayscale. For the pictures which format presupposes the storage of such flag (PNG and TIFF formats). `GrayScale()` method is used.
- Picture density horizontally and vertically. Allows getting the number of pixels per inch (DPI) in the relative coordinate. `HorizontalDensity()` and `VerticalDensity()` methods are used. If the picture format does not presupposes such information storage -the value of 72 pixels per inch is returned.
- Uploaded picture format. Returns the picture format in the form of system enumeration value `PictureFormat`. `PictureFormat()` method is used.
- The flag indicating that a picture is a set of pictures. `OptionSet` property is used.

If a set of pictures acts as a picture all methods return the information on the picture current variant. The picture currently displayed in the given client application is the "current" variant.

If the uploaded picture is a collection of pictures then the methods `Width()`, `Height()`, `HorizontalDensity()`, `Vertical Density()`, and `FileSize()` are meaningless for such picture.

Some conversions of the obtained picture are possible if necessary. For this, there is a special `PictureToProcess`. The picture to process is obtained on the basis of the original picture with the help of the object wizard `PictureToProcess`.

The picture parameters can be changed later and, in this case, each conversion method is applied to the picture currently placed in the object `PictureToProcess` taking into account the earlier transformations. The following actions are possible:

- To change the picture size vertically or horizontally. `SetSize()` method is used.
- To change the picture format with the help of the method `SetFormat()`.
- To zoom the picture in or out with the help of the method `SetFormat()`.
- To rotate the picture by some angle with the help of `Rotate() method`.
- To change the color depth or convert the picture in grayscale. `SetColorDepth()` and `ConvertToGrayScale()` methods are used for it (respectively).
- `SetDensity()` method allows setting the picture density value in the relative coordinate in the picture metadata. The values set will be used to determine the correlation between the picture real size and the size in pixels. If the picture current format does not presuppose such data storage -the call will be successful but no actual changes will be made.
- `SelectArea()` method allows specifying the picture fragment which will replace the original picture in `PictureToProcess` object after the method operation completion.
- `SelectPictureContent()` method allows selecting the content of the original picture.

To save the conversion result, get `Picture` object from `PictureToProcess` object and save it in a file on the disk or use the picture as intended in any other way.

The conversion of all pictures located in some directory from TIFF format to PNG format can be viewed as an example. In this case, the original pictures will be deleted from the disk.

Example:

```
SourceFiles = FindFiles(DirectoryWithPictures, "*.tif");
For each SourceFile From SourceFiles Cycle
  Picture = New Picture(SourceFile.FullName);
  Processing = New PictureToProcess(Picture);
  Processing.SetFormat(PictureFormat.PNG);
  Processing.GetPcture().Write(SourceFile.Path +
SourceFile.NaeWithNoExtension + ".png");
  Picture = Undefined;
  DeleteFile(SourceFile.FullName);
EndDo;
```

21.4.2. Support of addresses in the image field

The item of Field form, the type Image Field supports display of the image set by the address of the value (which may be the image or the binary data) in the temporary storage or in the database.

For this, set the attribute of the string type in the Data property of the form item. The value of this attribute shall be interpreted as the address of the image.

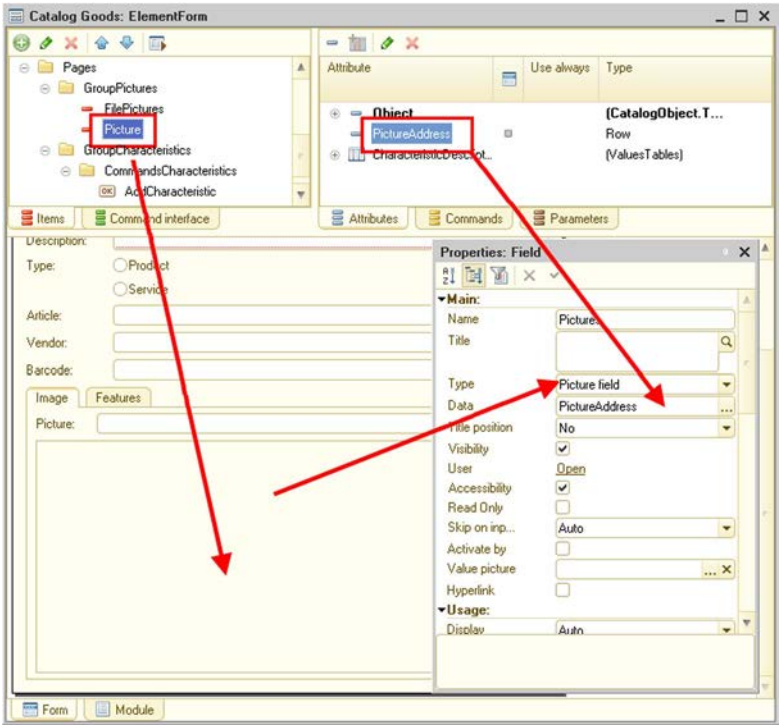


Fig. 285. Image display on the form

```
// Example 1
// Attachment of the image field to the image address in the temporary
// storage. ImageAddress - the form attribute of the string type
RecordFile (ImageAddress, OriginalName, SelectedName, True, UUID);

// Example 2
// Retrieval of the image address from the attribute of the infobase
// object
ImageFile =Object.ImageFile;
If No ImageFile.Blank() Then
  ImageAddress = GetURL(ImageFile, "FileData");
Else
  ImageAddress = "";
Endif;
```

Chapter 22. Event log

22.1. General information

To perform administrative tasks, it is often required to find out which events occurred at a particular point in time or what actions a particular user performed. For these purposes, event log is intended. Events are registered in this log. The administrator can get the history of user activities from the event log.
The event log is not a part of a database and is not saved when exporting/importing an infobase.
1C:Enterprise logs the major actions by users who modify infobases, give access or denial to the data, perform routine operations, log on, log off, etc.
Apart from interactive tools for the log operation granted in the designer, the program log operation is also available.
This chapter describes program tools for the event log operation.

22.2. Specification management

It is possible to manage the specification level of the messages sent to the event log. For this, `GetEventLogUsing` and `SetEventLogUsing` methods are used.

```
// install registration of all events in the log
Levels = New Array;
Levels.Add(EventLogLevel.Error);
Levels.Add(EventLogLevel.Information);
Levels.Add(EventLogLevel.Warning);
Levels.Add(EventLogLevel.Note);

SetEventLogUsing(Levels);
```

22.3. Recording of events

When you perform various procedures, you need to dock your actions for the further analysis. You may use the event log for such docking. Events are recorded in it using the global context method `WriteLogEvent()`.

NOTE. It is impossible to record system events in the event log using this method.

Names of user events may contain the point in their names, thus they form groups of user events.

22.4. Event log management

22.4.1. General information

Recording of any event in the event log may be switched on or off. For this, global context methods `GetEventLogEventUse` and `SetEventLogEventUse` and the object `EventLogEventUse` are used. The following example shows how to switch off the event recording Performance error (`_$PerformError$_`):

```
EventUse = New EventLogEventUse();
EventUse.Use = False;

SetEventLogEventUse("_$PerformError$_", EventUse);
```

Please note that there are two events with additional options of the record setting. They are events Access (`_$Access$_Access`) and Access denied(`_$Access$_AccessDenied`). We will describe more details of how to operate these events. However, when you set up recording of these events, please remember the following features:

- You should change recording settings when the exclusive mode is switched on.
- The change in the recording setting performed in the current session is to be applied only after all active sessions of this infobase will be restarted.

NOTE. Transaction-related events (`_$Transaction$_.Begin`,`_$Transaction$_.Commit`,`_$Transaction$_.Rollback`), may not be switched off using the method `SetEventLogEventUse()`.

22.4.2. Setting of the Access event parameters

The event Access is used to record facts of the system user access to this or that data. To set up recording of the event Access you should set up:

- need in event recording;

- list of meta data objects to record the access to;
- list of meta data object fields to record reading (**access fields**);
- list of meta data objects fields to record values (**record fields**).

The general operational scheme looks like this (provided that the event is recorded). If one of set access field is read during data operation (the selected meta data object), record fields are docked in the event log under the set rules. At this, the fact of access is docked at any access to the data made from the user session including the access from the code on the 1C:Enterprise language made on the 1C:Enterprise 8 server.

The amount of the data to be recorded in the event log depends on the setting of the event recording:

Usage	**Access fields**	**Record fields**	**Result**
Not assigned			The event is not recorded
Assigned	Not assigned	Not assigned	The event is not recorded
Assigned	Assigned	Not assigned	The event is registered without specification
Assigned	Assigned	Assigned	Event is recorded with specified record fields

Event is generated only after the data is successfully read.

Look at parameter settings using the specific sample:

```
DirectorySetting = New EventLogAccessEventUseDescription();

// Specify the object to record the access
DirectorySetting.Object="Directory.Individuals";

// Specify access fields
DirectorySetting.AccessFields.Add("PassportData");
DirectorySetting.AccessFields.Add("Children.BirthCertificate");

// Specify record fields
DirectorySetting.RegistrationFields.Add("PassportData");
DirectorySetting.RegistrationFields.Add("Children.ChildName");
FieldAlternatives = NewArray();
FieldAlternatives.Add("LastName");
FieldAlternatives.Add("Name");
DirectorySetting.RegistrationFields.Add(FieldAlternatives);

MetaDataObjectSetting = New Array();
MetaDataObjectSetting.Add(DirectorySetting);
```

This example shows recording the access to items of the directory `Individuals`.

At this, events of the event log shall be generated if the request to the data of the directory `Individuals` includes:

- field`PassportData`;
- field`BirthCertificate`of the tabular section`Children`.

If the request to the data does not include the described fields, the event of the access to the data shall not be recorded in the event log.

After the system specifies the access to the controlled fields the data of the following fields in the directory `Individuals` shall be recorded in the event log:

- Field`PassportData`.
- The field `ChildName` from the tabular section `Children`.
- Any field `Surname` or `Name`. The type of the field to be recorded in the event log is determined by the presence of these data in the request to the data. Moreover, if all listed fields are used, the event log shall contain the field docked with a smaller index (in the example -`Surname`). If one field of the specified pair is used in the request, this field shall be included in the event log.

Let's look at one more example of how the event Access is set up:

```
InformationRegisterSetting = EventLogAccessEventUseDescription();

// Specify the object to record the access
InformationRegisterSetting.Object="InformationRegister.EmployeeWages";

// Specify access fields
InformationRegisterSetting.AccessFields.Add("Wage");

// Specify record fields
InformationRegisterSetting.RegistrationFields.Add("Employee");

MetaDataObjectSetting.Add(InformationRegisterSetting);
```

In this example, the event Access will be recorded with the following components of the field `Data.Data` when the field `InformationRegister.EmployeeWages.Wage` is called:

- if the field `InformationRegister.EmployeeWages.Employee` (and the field `Link` is set in the setting of record fields for the directory `Individuals` the field `Employee` refers to) will result from the query, the following will be recorded to the event of the event log:
  - the table of values with the column `DirectoryIndividualsLink` containing values of links to objects of the directory `Individuals` will be recorded to the field `Data`;
  - the array with the string `InformationRegister.EmployeeWages` will be recorded to

the field MetaData, i.e. name of the object that has been the reason to record the access event;

  - the field MetaDataDisplay will contain the array with the string InformationRegister. Employee wages -display of the meta data object;

- if the field Link is not included into record fields of the directory Individuals, it will not be docked into the query to the information register;
- if the data will be acquired in the query using the link of the type Directory.Individuals stored in the information register, recording will be applied to the field acquired via the link stored in the field of the register Employee:
  - Name, Surname and PassportData (all of them are -acquired via the link stored in the field of the register Employee), the fields PassportData and Surname will be recorded to the event log (see description of recording for the directory Employees in the previous example).

22.4.3. Setting of parameters for the event "Access Denied"

The event Access denied is used to record facts of access to this or that data denied for system users. To set up recording of the event Access denied, you should set up the following:

- need in event recording;
- the list of meta data objects to record fields for denied access (denied access for other objects will be recorded without specification);
- the list of meta data object fields with values to be recorded (**record fields**) when access is denied.

The general operational scheme looks like this (provided that the event is recorded). If the access is denied during operation of the data (the selected meta data object), record fields are docked in the event log according to the preset rules.
The amount of the data to be recorded in the event log depends on the setting of the event recording:

Usage	Record fields	Result
Not assigned		The event is not recorded

Usage	Record fields	Result
Assigned	Not assigned	The event is recorded without specification (the field `Data` is not filled)
Assigned	Assigned	The event is recorded with specification of record fields in the field `Data`

The event is generated in the following cases:

- when the access to the data object is completely checked:
  - non-compliance with rights during call to methods and properties of the applied object from 1C:Enterprise language or from standard interface functions (forms, commands);
  - if access is denied when configuration rights are checked;
  - when the event emerges, the structure with the property `Right` containing the denied action will be migrated to the field `Data` of the log entry;
- when data access restrictions are checked:
  - if terms of data access restriction are not complied with;
  - when the event emerges, the structure with two properties will be migrated to the field `Data` of the log entry:
  - `Action` -the attempted action led to the denied access is specified;
  - `Data` - contains the information about record fields (if set).

NOTE. When the data is read to the temporary table, the field `Data.Data` of the event log is left blank.

Look at parameter settings using the specific sample:

```
DirectorySetting = New EventLogAccessDeniedEventUseDescription();

// Specify the object to record the access
DirectorySetting.Object="Directory.Individuals";

// Specify record fields
DirectorySetting.RegistrationFields.Add("PassportData");
DirectorySetting.RegistrationFields.Add("Children.ChildName");
FieldAlternatives = NewArray();
FieldAlternatives.Add("LastName");
FieldAlternatives.Add("Name");
DirectorySetting.RegistrationFields.Add(FieldAlternatives);
```

If the data access is breached, the event AccessDenied will be recorded in the procedure `Read` and the record field will be recorded according to the rules described in the record field of the event Access. The event is recorded for other actions (`Change`,`Delete`,`Add`) but the data are not recorded.

22.5. Getting event log records

During operation of the applications the situations requiring the program analysis of the event log including acquisition of events in the event log for some criteria (filter) may emerge. For program acquisition of entries in the event log, you should use the global context method `UnloadEventLog()`. You may programmatically acquire both entries in the event log of the infobase where the code is performed on the 1C:Enterprise language and of any other infobase (even if other 1C:Enterprise 8 instances operate this base) if the parameter `InputFileName` is given for the method `UnloadEventLog`. Please remember that such reading is recommended only in the case when files of the event log in other infobase are locally available for the instance of 1C:Enterprise 8 reading the log. If such access is unavailable, you should develop other methods to acquire the event log.

You should understand that entries of the event log may be acquired for a considerable period. It is especially important to know when entries are received for the large period or with some filter. The system does not limit the number of sessions that acquire the data from the event log in the file mode of operation. There are such limitations in the client/server mode of operation:

- not more than 4 simultaneous queries of the data from the event log per an infobase;
- if the divided infobase is used the additional limitation in 1 session of the data acquisition from event log is introduced for a data area.

When you operate the event log, the data display string recorded in the log during event logging is be displayed to the user only if it is possible to determine that this user has rights to view this display or the right of `Administration`. If the user has no right of `Administration` and it is impossible to determine whether the user has rights to view the display, the data display is hidden from them. Thus, the column `DataDisplay` shall be blank in the following cases:

- In designer: when the event log is viewed if the user has no right of `Administration`;
- In 1C:Enterprise 8 mode, when the method `UnloadEventLog()` is viewed and used, provided that all the following terms are complied with:
  - the user has no right of `Administration`;
  - the kernel mode is not set up;
  - the relevant data display is unavailable for this link, for any reason:
    - the data object is deleted;
    - the session uses not all independent separators that contain the data object;
    - the user has not enough rights to acquire the data object display.

- any term is complied for the relevant meta data object:
  - the user has no right of View;
  - the data access is limited for display fields.

You should use the parameter Filter of the method UnloadEventLog() to set up terms to filter selection. This parameter may contain either the single filter value set using the object Structure of the special format (more details of the structural format are described in the Syntax Assistant: Global context - Procedures and functions of event log operation -UnloadEventLog) or the array of such structures.

At this, if the filter is set up by the one item of the type Structure, the resultant selection will include all the entries that meet all the terms set in the structure (the terms are combined "for AND"). If the filter is set up by the array, the resultant selection will include the entries that meet even the single term listed in items of the array (the terms are combined "for OR").

Let's note the option to set up the filter for the field Data of the event log entry. Values of the type String,Number,Date,Link,Structure(for events Access,Access denied,Authentication,Authentication error,Add user,Change user and Remove user) as well as the array of structures may be uses as the filter for this field.

The certain value, Structure or Array may be used as the value for the item of the structure passed to the parameter Filter. Relevance of the data in the log entry is checked for relevance to the filter parameters as follows:

- If the certain value is passed, values in this entry is compared with the value of the filter.
- If the array of values is passed, values of the data in the event log entry are compared with all values of the array passed as the value of the filter. The term is considered to be complied with if the value specified in the log entry matches even single value of the passed array.
- If the structure is passed as the value, the term is considered to be complied with provided that relevance to the data of the event log entry will be set up for all items of the structure passed as the filter item.

For more details, see below.

1. The structure is located in the field Data of the event log entry, and the structure is also given in the field Data. In this case, the entry will meet the term if the field Data of the log entry contains all keys from the similar field of the filter and values for these case are relevant to each other.
2. The field Data of the log event entry contains the table of values, the structure is given in the field Data of the field. In this case, the entry will meet the term if the table of values has all columns with names equal to names

of keys in the filter structure and it has the strings where values of relevant columns are equal to values of the structure from the filter.

3. The field `Data` of the event log entry has the value of the type `Structure`, `Array` or `ValuesTable`, the field `Data` specifies the array. In this case the entry will meet the term if at least one value of the array passed as the filter value will be found in the data of the log entry.
4. The field `Data` of the event log entry has any value (not a structure, not an array, not a table of values), the filter field `Data` contains the array. In this case the entry will meet the term if the value of the log entry is equal to at least one value from the passed array.
5. The field `Data` of the log entry contains the item with the value of the type `ValuesTable`, the filter field `Data` contains the item with the value of the type `Array`, and keys of items are similar. In this case, the event log entry shall meet the term if at least one cell in the table of values is equal to at least one value of the passed array.
6. The item of the field `Data` of the log entry has the value of the type `Structure`,`Array` or `ValuesTable`, the filter field `Data` has the value of the type `Structure` or `Array`. The log entry shall meet the term if the entry that meets the term under paragraphs 1 and 5 is found in the nested data. However, if the structure is set up in the filter, items in the structure or columns of the table of values in the event log relevant to items of this structure will be searched only at the certain nesting level relevant to the nesting level in the filter. Thus, certain values and values from arrays in the filter are searched for the whole depth and items of the filter structure are searched only at the certain level.

If the generated table of values in the field `Data` of the event log entry has columns with the similar name, these columns are combined into one. At this, the number of strings in this table is increased so that the resultant column contains all unique values from columns with the same name. The values specified in other columns are copies to the generated strings.

Let's show the example. For instance, the application contains the query that retrieves the table with the following columns:

- `CatalogRef.Goods`,
- `CatalogRef.Goods`,
- `String`.

The following transformation will be completed once the entry is generated in the event log:

- First two columns will be combined into one.
- One more string will be generated for each string where values `Goods.Link` and `Products.Link` are not equal.

- Strings will differ by the value of the combined column where values of links will be located. Other values of columns will be similar.

Initial table

Link	Link	Item
Hot dogs	Pepper	16-AB-1675

Resultant table

Link	Item
Hot dogs	16-AB-1675
Pepper	16-AB-1675

Let's also give the examples of how filters operate when entries of the event log:

Example 1:

- value of the field `Data` in the event log entry for the event Authentication:
  - type: `Structure`:
    - key: `OSUser`, value Ivanov;
- value of the field `Data` in the filter of the method `UnloadEventLog()`:
  - type: `Structure`:
    - key: `OSUser`, value Ivanov;
- result: record matches.

Example 2:

- the value of the field `Data` of the event log entry for the event Access:
  - type: `Structure`:
    - key: `Data`, value: ValuesTable:

Last Name	City	Phone
Petrov	Tula	111-22-33
Ivanov	Moscow	222-33-44

- value of the field `Data` in the filter of the method `UnloadEventLog()`:
  - type: Array:
    - item 1: type `Structure`:
      - key: `OSUser`, value Ivanov;
    - item 2: type `Structure`:
      - key: `Data`, value: `Structure`;
      - key: `Last Name`, value: Ivanov;
      - key: `City`, value: Moscow;
- result: the entry matches the filter because the match with the entry data (string 2) is found during comparison of the 2nd item with the filter array. The event log entry has the table of values with columns `Name` and `City`

and both values of columns are equal to relevant values in the filter structure.

Example 3:

- the value of the field `Data` in the event log entry for the event Add user:
  - type: `Structure`:
    - key: `Roles`, value: `Array`:
      - value 1: Roles.Administrator;
      - value 2: Roles.Storekeeper;
      - value 3: Roles.Salesman;
- value of the field `Data` in the filter of the method `UnloadEventLog()`:
  - type: `Structure`:
    - key: `Roles`, value: Roles.Storekeeper;
- result: the record matches the filter, because the field `Data` has the item `Roles`, and values of the item include the value Roles.Storekeeper.

Example 4:

- the value of the field `Data` in the event log entry for the event Add user:
  - type: `Structure`:
    - key: `Roles`, value: `Array`:
      - value 1: Roles.Administrator;
      - value 2: Roles.Storekeeper;
      - value 3: Roles.Salesman;
- value of the field `Data` in the filter of the method `UnloadEventLog()`:
  - type: `Structure`:
    - key: `Roles`, value: `Array`:
      - value 1: Roles.Manager;
      - value 2: Roles.Salesman;
- result: the record matches the filter, because the field `Data` has the item `Roles`, and values of the item include the value specified in the filter array (the value Roles.Salesman).

Example 5:

- the value of the field `Data` of the event log entry for the event Access:
  - type: `Structure`:
    - key: `Data`, value: ValuesTable:

Last Name	**City**
Petrov	Tula
Ivanov	Moscow

- value of the field `Data` in the filter of the method `UnloadEventLog()`:

    - type: Array:
        - type`Structure`:
            - key: `Data`, value: Tula;
- result: the entry matches the filter because the initial value (Tula) is found in one cell in the table of values located in the property `Data` of the field `Data` in the event log entry.

For the field `Meta Data` of the event log entry where the meta data array is stored in the event, the entry matches the filter provided that at least one item of the array from the field `Meta Data` of the filter is equal to at least one item of the log entry array.

22.6. Additional methods

To facilitate interactive generation of event log filters, it's good to use the method `GetEventLogFilterValues()`. This method helps you to retrieve available filter values for the following filter parameters: `User`, `Computer`, `ApplicationName`, `Event`, `MetaData`, `IWorkingServerInfo`, `MainIPPort`, `AuxIPPort`. Thus, if you need to set up the filter for the event log event, you may first acquire the list of events that are actually present in this log and then select from the acquired list.

The method `EventLogEventPresentation()` is used to generate displays of event log evens on the platform interface language

If you need to copy the fragment of the event log with the specific filter, you should use the method `CopyEventLog()`.

Chapter 23. Cryptographic feature

23.1. General information

If you use 1C:Enterprise 8 in automation systems you may need to check whether this or that document stored in the system has not been changed (e.g. the document containing the text of the contract is attached to the database object `Contract`). You may also need to provide for transfer any authorized information or to organize transfer of any document within the system. The situations may arise when you need to provide for transfer of information via open channels so that nobody will be able to see it by interception of transfer (to encrypt the data).
For such situations the cryptographic feature is implemented in 1C:Enterprise 8 that is based on the asymmetrical encryption (the couple of keys is used **-open** and **closed**).

IMPORTANT! The cryptographic feature of 1C:Enterprise 8 does not contain implementation of cryptographic algorithms as they are. It provides a set of objects for interaction with external third-party cryptography modules.

The open key is used for transfer via open channels. **The closed key** is not used for distribution and must be protected at the highest level.
You need to know open keys of receivers to encrypt the data. To decipher, you need to have the closed key which is the pair for the open key listed during encryption. You need the closed key to generate the digital signature and the - open key of the signatory (the open key is frequently included into the signature).
The symmetrical encryption is used alongside with the asymmetrical encryption to encrypt the data. The symmetrical encryption - is the method of encryption when the same key is used for ciphering and deciphering. The section further describes features of symmetrical encryption application.
The certification center is used to confirm that the certain open key really belongs to some subject. It ensures it by the signature.
The certificate- is the open key signed by the certification center. The closed key exists in the container due to the need of protection. Apart from the closed key, **the key container** may include the open key (for example, as a certificate).
When you operate the cryptographic feature you should remember where you operate it. The reason is that cryptographic operation designs are considerably

different for the client application and for the server. The server uses so called synchronous cryptographic operation method and the client application uses asynchronous operation methods.
The cryptography extension must be installed for operation in the web client. The feature of the web client is execution of the query for the user's authorization to some procedures: access to the file system and to the closed key.

23.2. Basic concepts

Some terms will be used to describe the features.
The electronic digital signature (hereinafter -referred to as **EDS**) - is the sequence of the data acquired as a result of the cryptographic initial data transformation using the closed EDS key. It helps to confirm integrity and stability of this data as well as its authorship provided that the open EDS key and its certificate are used.
The center for certification of the electronic digital signature (hereinafter referred to as -**Certification Center**)- is the legal entity or the special division of the legal entity with authorities to certify that the certain open EDS key belongs to the certain user.
The owner of the key certificate- is the individual for whom the certification center makes the open key certificate and who owns the relevant closed (secret) key.
The certificate -electronic document including the open key and the information about the owner of this key certified by the certification center using the EDS.
The cryptography module -library of functions where cryptographic algorithms are directly implemented or used for access to cryptographic features.

23.3. The general design to use cryptographic features

Two principal options may be found for operation of cryptographic features:

- data encryption/decryption;
- data signature/signature check.

The general operational design for encryption/decryption looks like this:

1. The needed object of access to cryptographic functionality is generated (the object `CryptoManager`).
2. The certificate is selected that is **the open** key of the recipient (the object `CryptoCertificate`). This certificate will be used to **encrypt** the data.

3. The needed file or the binary data are encrypted using the methods `Encrypt()` of the generated object, the type `CryptoManager`.
   Combined symmetrical and asymmetrical encryption is used for encryption. First, the system generates the key for symmetrical encryption and then this key encrypts the required data. Then the key of symmetrical encryption is encrypted by the asymmetrical key. The data batch suitable for transfer via open channels is generated by three components: the encrypted data, the encrypted key for symmetrical encryption and the list of recipients. The data batch is generated according to the CMS specification (based on PKCS#7).
4. The encrypted data is ready for transfer via open channels.
5. The reversed procedure is done when the encrypted data is received.
6. The needed object of access to cryptographic functionality is generated. The cryptography module related to the object should support operation of algorithms used for the data encryption.
7. The received data are decrypted using the method `Decrypt()` of the object generated for access to cryptographic functionality.
   The encrypted key for symmetrical encryption is extracted from the incoming batch during decryption and it is decrypted using the closed key (the pair to the open key used for encryption). Then the acquired key is used in the symmetrical algorithm to decrypt the array of the acquired data. Thus, if the receiving party ("from" of the symmetrical encryption) does not have any closed key, it will be impossible to decrypt the data by the symmetrical algorithm.

The general operational design during signature/signature check looks like this:

1. The needed object for access to cryptographic functionality is generated (the object `CryptoManager)`.
2. The needed certificate related to the **closed** key of the signatory is selected (the object `CryptoCertificate`). The EDS will be **generated** using this certificate.
3. The EDS of the needed file or the binary data will be generated using methods `Sign()` of the generated object of the type `CryptoManager`.
   The following features should be remembered during generation of the signature: first, the hash of the signed data is generated and then the acquired hash is signed. It is done to reduce the time needed to complete the data signing procedure.
   Due to that it is recommended to expressly specify the algorithms to be used for both calculation of the hash (the property `HashAlgorithm`) and the signature (the property `SignAlgorithm`). If the recipient does not support the used algorithms (hashing and signature) it will be impossible to check the signature.

You need to use the method `ReceiveCryptographyModuleInformation ()` of the cryptographic manager to expressly specify the used algorithms. As a result of the method, the object `CryptoModuleInformation` will be acquired where properties `SignAlgorithms` and `HashAlgorithms` will contain the list of algorithms supported by the used cryptography module. The data contained in properties of the object `CryptoModuleInformation` should be specified in properties of the cryptographic manager.
The information about used algorithms shall be contained in the signature so it is not necessary to pass it together with the signed data.

4. The signed data and the electronic digital signature are passed to the recipient.
5. The reversed procedure is done once the signed data and the EDS are received.
6. The needed object of access to cryptographic functionality is generated. The cryptography module related to the object should support operation of algorithms used to generate the EDS.
7. The electronic signature is checked using the method `VerifySignature()` of the generated object for access to cryptographic functionality. If the cryptography module used to check the signature does not support the algorithm used to calculate the hash and the data signature, it will be impossible to check the signature and the error will be displayed.

The certificates needed to complete cryptographic procedures are acquired from the relevant certification center.
It is not recommended to acquire several storages of certificates with similar features to operate the object `CryptoManager` because modification of one storage may lead to different behavior of the other storage depending on the used cryptography modules.

23.4. Operation of cryptography modules

Microsoft CryptoAPI is used for interaction with cryptography modules in Windows OS.
The direct interaction with installed components is used for interaction with cryptography modules in Linux OS.
The list of supported components:

- CryptoPro, the means for cryptographic data protection. You should specify the value `75` as a parameter `CryptoprotectionModuleType` of the constructor for the object `CryptoManager` to use this means.

23.5. Examples of usage

This section gives examples for implementation of some standard tasks for usage of the cryptographic feature.

Generation of the object for access to the cryptographic functionality

Generation of the object for access to the cryptographic functionality is the basic procedure without which it is impossible to complete further procedures with the cryptographic feature.

```
CryptoManager = New CryptoManager("", "", 75);
```

In this example, the module is generated to operate Russian cryptography modules (the value of the parameter `CryptoprotectionModuleType` is 75).

Getting a certificate list

The list of certificates is generated from the selected storages of certificates for further operation of this list.

```
&OnClient
Function AcquireCertificateList (CryptographicManagerType, TypeArray,
CheckEndDate)

  // List of certificates
  CertificateList = New Array;

  CryptoManager = New CryptoManager("", "", CryptographicManagerType);

  For Each StoreType From TypeArray Do

    // Get certificates for each type of certificate storage
    Storage = CryptoManager.GetCertificateStore(StoreType);

    // Select all certificates
    StorageCertificates = Storage.GetAll();
    CurrentDate = CurrentDate();

    For Each Certificate From StorageCertificates Do

      If CheckEndDate And Certificate.EndDate < CurrentDate Then

        // Skip expired certificates if needed
        Continue;

      EndIf;

      CertificateList.Add(Certificate);

    EndDo;

  EndDo;

  Return CertificateList;

EndFunction

&OnClient
Procedure GetCertificateList()

  StorageTypes = New Array;
  StorageTypes.Add(CryptoCertificateStoreType.PersonalCertificates);
  StorageType.Add(CryptoCertificateStoreType.RecipientCertificates);
  List = GetCertificateList(75, StorageTypes, True);

  // ...

EndProcedure
```

File encryption

The interactively selected file is encrypted and the encrypted file is interactively recorded to the hard drive of the client computer.

The first certificate from the list of all certificates installed on the computer is always used for the encryption procedure in the purpose of demonstration.

```
&OnServer
Function EncryptOnServer(DataAddress, CertificateData)

  // Generate certificates based on the binary data of certificates
from the client
  Certificates = New Array ();
  For Each CertificateData From CertificateData Do

    Certificates.Add(New CryptoCertificate(CertificateData));

  EndDo;

  CryptoManager = New CryptoManager("", "", 75);

  // Get the file for encryption from the temporary storage
  Data = GetFromTempStorage(DataAddress);
  If TypeOf(Data) <> Type("BinaryData") Then

    Return False;

  EndIf;

  // Encrypt the binary data
  EncryptedBinaryData = CryptoManager.Encrypt(Data, Certificates);

  // Save in the temporary storage
  DataAddress = PutToTempStorage(EncryptedBinaryData);

  Return True;

EndFunction

&OnClient
Procedure FileEncryption()

  Address = "";
  Result = MoveFile(Address, , ,True);
  If No Result Then

    Return;

  EndIf;

  CertificateTypes = New Array;
  CertificateTypes.Add(CryptoCertificateStoreType.PersonalCertificates
);
  List = GetCertificateList(75, CertificateTypes, True);

  // Always encrypt using the first certificate of the list in the
example
  Certificates = New Array;
  Certificates.Add(List[0].Export());
```

```
  // Encrypt the file
  Result = EncryptOnServer(Address, Certificates);
  If No Result Then

    Return;

  EndIf;

  // Save the encrypted file interactively on the hard drive
  GetFile(Address, , True);

EndProcedure
```

File decryption

Decryption of the selected file is attempted. The function `GetAccessPassword` should be implemented to return the value of the password for access to the closed key.

NOTE. While using the method `Decrypt()` the exemption will be called only after attempts of decryption have failed using all available certificates, not after the first error.

```
&OnClient
Procedure FileDecryption()

  // Selected the encrypted file
  Address = "";
  Result = MoveFile(Address, , ,True);
  If No Result Then

    Return;

  EndIf;

  Data = GetFromTempStorage(Address);

  // Decrypt the file
  CryptoManager = New CryptoManager("", "", 75);
  CryptoManager.PrivateKeyAccessPassword = GetAccessPassword();
  DecryptedData = CryptoManager.Decrypt(Data);

  Address = PutToTempStorage(DecryptedData);
  GetFile(Address, , True);

EndProcedure
```

EDS generation

The file is formed with the electronic digital signature. In this example, the EDS is always saved to the file signature.sign. The function

`GetAccessPassword` should be implemented to return the value of the password for access to the closed key.

```
&OnClient
Procedure SignFile()

  Address = "";
  Result = MoveFile(Address, , ,True);
  If No Result Then

    Return;

  EndIf;

  CertificateTypes = New Array;
  CertificateTypes.Add(CryptoCertificateStoreType.PersonalCertificates
);
  List = GetCertificateList(75, CertificateTypes, True);

  Certificate = List[0];
  Data = GetFromTempStorage(Address);
  CryptoManager = New CryptoManager("", "", 75);
  CryptoManager.PrivateKeyAccessPassword = GetAccessPassword();
  CryptoManager.Sign(Data, "signature.sign", Certificate);

EndProcedure
```

EDS checking

The function checks whether the EDS is valid for the couple original file - EDS file.

```
&OnClient
Function CheckFileSignature(SignedFileName, SignatureFileName)

  Certificate = Unspecified;
  CryptoManager = New CryptoManager("", "", 75);
  Try

    CryptoManager.VerifySignature(SignedFileName, SignatureFileName,
Certificate);

  Exception

    Return False;

  EndTry
  Return True;

EndFunction
```

23.6. The asynchronous operation of the cryptographic feature

The asynchronous design for operation of cryptography has some features:

1. It is available only from the client application. You should use the synchronous operation from the server.
2. Initialization of objects for access to the cryptographic functionality is different from its synchronous analogues.

Let's look at operation of the cryptographic functionality in an asynchronous mode. As an example, let's have a look at the procedure for acquisition of the certificate list from the certain type of the certificate storage.

NOTE. The example below is not complete. It is used to demonstrate the asynchronous operation of cryptographic means.

```
&OnClient
Procedure GetCertificateList()
  AdditionalParameters = New Structure;
  AdditionalParameters.Insert("CertificateStorageType",
CryptoCertificateStoreType.PersonalCertificates);
  AdditionalParameters.Insert("CheckEndDate", True);
  NotifyDescription = New NotifyDescription("GetCertificateListEnd",
ThisObject, AdditionalParameters);
  AdditionalParameters.Insert("EndNotification", NotifyDescription);
  Callback = New NotifyDescription("EndInitialization", ThisObject,
AdditionalParameters);
  Cryptography = New CryptoManager;
  Cryptography.BeginInitialization(Callback, "", "", 75);
EndProcedure

&OnClient
Procedure GetCertificateListEnd(CertificateList, AdditionalParameters)
Export
  // use acquired certificates here
EndProcedure

&OnClient
Procedure EndInitialization(Manager, AdditionalParameters) Export
  Notification = New NotifyDescription("GetCertificateStorageEnd",
ThisObject, AdditionalParameters);
  Manager.BeginGettingCertificateStore(Notification,
AdditionalParameters.CertificateStorageType);
EndProcedure

&OnClient
Procedure GetCertificateStorageEnd (Storage, AdditionalParameters)
Export
  Notification = New NotifyDescription("CertificateArrayEnd",
ThisObject, AdditionalParameters);
  Storage.BeginGettingAll(Notification);
EndProcedure

&OnClient
Procedure CertificateArrayEnd(Certificates, AdditionalParameters)
Export
  Result = New Array;
  CurrentDate = CurrentDate();
  For Each Certificate From Certificates Do
    If AdditionalParameters.CheckEndDate And Certificate.EndDate <
CurrentDate Then
      Continue;
    EndIf;
    Result.Add(Certificate)
  EndDo;
  ExecuteNotifyProcessing(AdditionalParameters.EndNotification,
Result);
EndProcedure
```

This example shows all features of operation with cryptography in an asynchronous mode. The list of certificates starts being acquired from the call of the method `ObtainCertificateList()`and ends by the call of the method `GetCertificateListEnd()`. Acquisition as it is consists of initilization of the cryptographic operation module and using asynchronous analogues of the following synchronous methods: `GetCertificateStore()` and `GetAll()` The individual method is used to check expired certificates. Notification `EndInitialization()` will be called after initialization of the cryptographic manager ends. Then the operation continues in the same way: the next action is called in the handler for the end of the previous method in which end handler the next action happens etc.
The accurate sequence of actions is acquired:

```
ObtainCertificateList()-->
  EndInitialization()-->
    GetCertificateStorageEnd()-->
      CertificateArrayEnd()-->
GetCertificateListEnd()
```

Only the last transfer needs the special attention. No asynchronous method requiring the instruction of the notification handler is called in the notification handler `CertificateArrayEnd()`. The needed description of the notification is received by the very last (in the chain of calls) notification handler once additional parameters are set up. The needed actions are completed in the method `GetCertificateList()` and the expressed call of the handler which "indicator" has been consequently passed via the chain of handlers using additional parameters happens in the very last handler (`GetCertificateListEnd()`). The actual return happens as a result of the method `ExecuteNotifyProcessing()`.
The example is developed to be used in the module of the form but processing of the example to be operated from common modules is not a significant issue. In this case, first two procedures of the example will be in the same common module and the rest will be -in the other with the relevant adjustment of description of notification and method call handlers.
The other standard methods for operation of cryptography will look the same.
You should install the cryptography extension to operate the cryptographic feature in the web client. However, you should remember that both synchronous and asynchronous cryptographic operation methods may be used in web browsers except Google, Chrome and Mozilla Firefox. At this, the used set of methods is determined by the application property Mode of using synchronous platform extension calls and add-ins. When you use web browsers Google Chrome and

Mozilla Firefox, only asynchronous cryptographic operation methods are available. You may see correlation of methods in Syntax Assistant, in the description of synchronous methods.

Chapter 24. External data sources

24.1. General information

When you operate the information system based on 1C:Enterprise 8, the tasks may arise that are related to acquisition of the information from external databases (both relational and analytical databases) and using this information in 1C:Enterprise 8 in a different manner, for example, as reports or for any calculations. The configuration object `ExternalDataSources` is used to solve such tasks in 1C:Enterprise. The external data source may include tables of the relational data source and cubes of the analytical data source. The application may have the random amount of attached external data sources. We will look at the operation of every type of the external data source in details.

24.2. Operation of relational external data sources

24.2.1. General information

The object describing the external data source attached to the relational database consists of tables and each table consists of fields. The data in tables may refer to other tables - for example, the value in the field of the table may be the ID of the entry in the other table. The object describing the external data source may be used as follows:

- as the data source for queries;
- as the data source in the data composition system;
- as the source for dynamic lists;
- be included into general attributes;
- entries of the table may be displayed in managed forms of 1C:Enterprise 8 (ordinary forms are not supported for objects of external data sources);
- tables of the external data source may be used as types of the infobase attributes;

- you may apply access rights and data access restrictions to for tables (and fields) of external data sources;
- tables and fields are accessible from the 1C:Enterprise language;
- the table of the external data source may be included into subsystems;
- the table of the external data source may be included into functional options;
- you may generate characteristics for the table of the external data source.

The ODBC feature is used to get access to external data sources. The data of external data sources are available for both **reading** and **recording**.

IMPORTANT! The feature of external data sources should not be used for access to databases of 1C:Enterprise 8 because the data model of 1C:Enterprise 8 is not intended for operation of the data at the level of the physical storage structure in database management systems (DBMS). The option for record to external data sources should not be used to replace the standard feature of the data storage in the application.

Options of 1C:Enterprise 8 are maximally considered when used as the external data source of the following DBMS (the property DBMS type of parameters for connection with the data source):

- Microsoft SQL Server,
- IBM DB2,
- PostgreSQL,
- Oracle Database,
- MySQL.

NOTE. When you operate the external data source, errors may appear if the property DBMS type of parameters for connection with the external data source contains the value not relevant to the system that is actually used.

When you use other DBMS, options for operation of external data sources depend on the DBMS. For such data sources the following functions and expressions of the query language are used: SUBSTRING, YEAR, QUARTER, MONTH, DAYOFYEAR, DAY, WEEK, WEEKDAY, HOUR, MINUTE, SECOND, DATADIFFERENCE, YESNULL, EXPRESS, SPECIALSYMBOL, CONNECTION, LEFT CONNECTION, RIGHT CONNECTION, FULL CONNECTION, are transformed into the query to DMBS using controlling DBMS sequences (DBMS Escape Sequences, http://msdn.microsoft.com/en-us/library/windows/desktop/ms715364(v=vs.85).aspx, in English).

If you use the external data source as the source for the dynamic list, you should remember that the dynamic list is arranged using the sorting order installed for the database used as the external data source. The sorting order may not coincide with the sorting order used in the case when the database is used by 1C:Enterprise 8. Due to this, the situations are possible when the result of arrangement will be different in the infobase and the external data source (when the arranged data are the same). This behavior is normal.
In order to connect to the external data source, you should generate the string of connection that may contain either all connection parameters needed for the selected ODBS driver or indication to the generated description of the data source DSN (Data Source Name).
Examples of connection strings are given below.

The connection string with indication of access login and password:

```
DRIVER={SQL
Server};SERVER=(local);UID=user;PWD=password;DATABASE=AdventureWork
sLT2008
```

Access login and password are to be indicated additionally:

```
DRIVER={SQL Server};SERVER=CASH SERVER;DATABASE=CasheReceipts
```

Indication of the preset data source (DSN):

```
DSN=MyDB
```

You should also remember that you should indicate connection with the external data source not only in the designer (if you use the feature for table structure import from the external data source) but in the mode 1C:Enterprise 8 to get the data.

24.2.2. The general operation design

You should comply with the following general design to use the information from external data sources based on 1C:Enterprise 8:

- To study the structure of the external data source and to know what information (tables and fields) are needed to operate algorithms or reports in 1C:Enterprise 8.
- To generate the object External data sources in the designer and to generate subordinate tables and fields for it. You can do it using the special helper.
- To implement usage of generated objects in the application.

- To set up parameters for connection to the external data source in the network where the application will be used. These parameters may be different from the ones used for downloading the structure of the external data source.

NOTE. Parameters of access to the external data source that have been set up in the designer will not be used by the system in the mode 1C:Enterprise 8.

24.2.3. Editing the structure of the external data source

24.2.3.1. General information

The branch with names of the relevant external data source in the branch External data sources-<Name of the external data source>-Tables of the configuration tree is used to operate tables of external data sources. Determination of the external data source consists of the following steps:

1. Determination of the external data source;
2. Determination of data source tables;
3. Determination of fields for each table of the external data source.

At this, configuration may describe the less amount of tables and fields in them then the real database has but it is impossible to set up the table or the field not included in the real database.

The structure of the external data source can be generated both in the manual mode and when this structure is uploaded with the special constructor.

24.2.3.2. External data source

You should indicate the name of the external data source when you generate it. This object is used to identify the group of tables during the call of the data it contains. The data source consists of one or several tables that consist of fields.

If the entry could be made to the external data source, you should set up the needed value in the property Mode of data lock management. If the lock management mode is set up to the value Automatic and manageable for the external data source, the lock mode for each table is determined by the value of the relevant table property. If the certain special lock management mode is selected for the external data source, the property of the external data source table with the same name is ignored.

24.2.3.3. The table of the external data source

When you generate a new table, the window for editing the object is open.

The property of the table Name is used to identify the table inside the application. For the system to know what physical table of the external data source is related to the object of configuration you should fill the property Name in the data source. The value in this property should exactly match the name of the table in the external data source. Placing the application of the data from several physical tables of the external data source in the same table is not supported.
The property The table data type determines what entities are stored in this table: object-based or non-object-based. If the one field that uniquely determines the entry in the table can be singled out in the table, it means that the table may store the object-based data. The directory may be the closest analogue of these tables. If the entry in the table is identified by several key fields, such table contains the non-object-based data. The information register is the closest analogue of these tables. Key fields of the table should be indicated in the property Fields of the key.

NOTE. It is not recommended to use the fields with the values `NULL` as key fields of the object-based external data source table.

The field that will be used as the representation of the object may be indicated for the tables with the object-based data. You may do this using the property View field.
Depending on the DBMS object that describes the generated object in 1C:Enterprise 8, you should correctly set up the property Table type. If the property is set up in the value Table- the configuration object will be the analogue of the real database or view table. In other cases the property Table type should be installed in the value Expression. If the external data source table has the type Table you should indicate the name of the real table or non-parametrized view, in the property Name in the data source, for instance `dbo.ErrorLog`. You should indicate the property Expression in the data source for the table of the type Expression. In the simplest case, it will be the call of the function returning the table data, for instance `dbo.ufnGetContractInformation(&1)`. If the expression is recorded in this field, it is recommended to indicate it in brackets to minimize various conflicts for real performance of the query to the DBMS.
If the table stores the hierarchic data you may inform the system about it using properties Parent field, Unfilled parent flag and Unfilled parent value. These properties are available only for the tables where the object data are stored. The property Parent field is used to indicate the field of the database table that stores the link to the parent for this entry. The fields that may be selected in this property should have the type of the link to the used table. For

the table products from the example the parent field should have the type ExternalDataSourceTableLink.DBF.products. Properties Non-filled parent flag and Non-filled parent value determine what value will specify the entry without the parent entry. Once these properties are indicated:

- the data composition system uses them to operate hierarchical groupings and when hierarchic conditions are checked;
- the query language uses them to operate the construction IN HIERARCHY;
- these properties are used to display the tables storing the hierarchical data (display options are the same as the options used to display hierarchical directories).

The property String input allows to specify names of the fields for which the search in the field will be done, which data type specifies the table of the external data source.

When tables are manually generated the properties Key field, View field and String input may be specified only when fields of the external data source table are generated.

The check box Read only is responsible for the option of the record in the table of the external data source. If the check box is cleared - the information may be recorded in the specified table. The check box is set up in the value True automatically for the table based on views and functions. You should note the following properties for the table to which the information may be recorded:

- Transaction isolation level -determines the isolation level for transaction that will be set up at implicit transactions of the entry to this table.
- Data version fields -this property specifies the field of the external data source that increases its value automatically after each record into this table. If the field is specified the system uses the value of this field during interactive modification of the object. The value of this field is compared in the database and in the form during recording: if values are different it means that the object is modified.
  If the field is not specified the object is read to the memory when the form is interactively open and the object is read again and compared with the copy in the memory during recording. If two copies are different it means that the object is modified.
- Data lock fields -indicate fields for which the data may be locked.
- Introduced on the grounds -configuration objects are specified on which grounds records in the table may be generated.

- Data lock management mode -specifies what lock management mode will be applied for the entry to this table.

24.2.3.4. The field of the external data source table

Fields of the table describe what data of the physical table will be available from the application. The property of the field Name is used to identify fields of the table in the application. The property Name in the data source is used to specify relations between the field of the application and the field of the physical table. The value in this property should exactly match the name of the table's column specified in the property Name in the data source of the parent object in this field. Combination of the data from several columns of one or several external data source table in the same field of the data table is not supported.

If the value in the property Name in the data source is inside the single quotes, this value goes to the SQL query to the data base without transformations, regardless components of symbols. If the value in the field is not inside the single quotes, this value will go to the SQL query to the database inside the double quotes if the name contains special symbols.

The field Type allows to specify the type of this field. The limited amount of types is available for selection. Field types may be:

- `Number`;
- `String`;
- `Date`;
- `Boolean`;
- `UUID`;
- `BinaryData`;
- the types determined by tables of external data sources.

If you have to specify the compound type for the field, only types `Number`, `String`, `Date`, `Boolean` may be included in this compound type.

When the data are received from external data sources, the system automatically transforms these data to the type specified at the relevant field in the application.

The property Read only specifies that this field cannot be recorded in the table. This property should be set up in the value `True` for fields with automatic modification, automatically generated key fields, calculated fields etc. However, when you operate the external data source, you will need to record the fields that are intended for read-only in the standard operation mode and vice versa. At this, such need is temporary, for example, it is needed to complete some single or by-the-book procedure. You should use methods

GetChangeableFields() and SetChangeableFields() to modify the list of recorded fields. Thus, the condition of the property Read only describes the default behavior of the field and in some rare cases you can modify the behavior of the field using the said methods.

The property Allow Null specifies whether you can record NULL in the field of the table or not. If NULL is recorded in the field of the table, this value will be displayed in the form as Not filled.

24.2.3.5. Other properties

The form used to display the record and the list of the external data source table may be generated by the system automatically or by the applied developer. At this, remember that if tables are non-object-based and key fields are not set up then:

- the form of the record cannot be displayed;
- no access to the object ExternalDataSourceTableRecordSet.

It is related to the fact that it is impossible to uniquely identify the necessary record set in the table.

24.2.3.6. Operation of transactions

Both automatic and manageable locks may be used for recording in external data sources.

Methods BeginTransaction(), CommitTransaction(), RollbackTransaction() and TransactionActive of the external data source manager are used to operate transactions. In particular, it means that transactions for several data sources (distributed transactions) are unavailable.

Lock management is made using the following properties and objects:

- The property Data lock management mode of the external data source;
- The property Data lock management mode of the external data source table;
- The fields specified in the property Data lock fields of the external data source table;
- The property Transaction isolation level of the external data source table;
- Parameters of the method BeginTransaction() of the external data source manager.

The property Data lock management mode of the external data source is used to set up in default the lock mode of external data source tables (the

same as the configuration property Data lock management mode). This property can have the following values:

- Automatic -mode of automatic locks for all external data source tables. Serializability is used as the transaction isolation level. This method allows using transactions to the external data source base simultaneously from the platform and from other clients. The property with the same name is ignored at the external data source table in this mode.
- Manageable -the manageable lock mode for all external data source tables. Read docked is used as the transaction isolation level. This mode may be used when only 1C:Enterprise 8-based system is related to the external data source base. The property with the same name is ignored at the external data source table in this mode.
- Automatic and manageable -the lock mode is selected for each table depending on the property of the table with the same name. The transaction isolation level is selected depending on the property Transaction isolation level of the external data source table.

If the property Transaction isolation level of the external data source table is set up in the value Auto the actual transaction isolation mode is set up depending on the property Data lock management mode (described above). If the certain property is specified - it will be used. If DBMS of the external data source does not support the transaction isolation level specified in this property, the level will be automatically modified at the following chain (until the first supported one): Read non-docked → Read docked → Repeated read → Serializability.

When operation is done in the manageable lock mode the locks at the database level are overlaid by setting of the relevant transaction isolation level and by specification of the expression `FOR CHANGE` in queries to the database.

The method `TransactionActive()` allows to determine the current condition of the transaction in the used external data source.

24.2.3.7. Special features of the system behavior during recording in external data sources

When operation of the external data source is not interactive, you should remember the following special features of the behavior:

- When the record is generated in the external data source table:
  - It is impossible to change attributes of the recorded objects where the property `ReadOnly` is set up in the value `True`.

    - If the property `ReadOnly` of the attribute that is the component of key fields is set up in the value `True` such attribute may be changed only as a result of calling the method `SetNewObjectRef()`. If the property `ReadOnly` of such attribute is set up in the value `False` its value may be changed expressly (using the assignment procedure).
    - The system does not automatically fill values of attributes used to generate the key of the object (or the set of records). It means that the record will be completed with the error if values of key attributes are not set up for the recorded object.
- When the record is changed in the external data source table:
    - Only the attributes for which the property `ReadOnly` is set up in the value `False` are changed.
    - When key fields are changed the value of the link is changed.
    - If the attribute `DataVersion` is filled this value is used in the expression `WHERE` for the query to update the data. After the query is successfully completed the value of the attribute is read again from the database.
- When the record is deleted from the external data source table:
    - Only direct deletion is completed. The deletion mark is unavailable and not supported.
    - Integrity of the link is not controlled during deletion.

When DBMS PostgreSQL is used as the external data source the record of fields of the type `UUID` is not supported.

When UUID is recorded in the field of the external data source table, the platform 1C:Enterprise 8 rearranges the data of UUID. Thus, the data will be moved to the external database in a modified type. When UUID is read the data is arranged back. That's why UUID preserves its initial value during recording/reading.

Storage of the value of the type `UUID` in the computer memory can be represented as following (the byte arrangement "young-old"):

- the value `Data1` 4 byte long;
- the value `Data2` 2 byte long;
- the value `Data3` 2 byte long;
- the value `Data4` 8 byte long.

The value of the type `UUID` will be recorded in the database for such storage as follows:

- the value `Data4` as a whole;
- 1st byte of the value `Data3`;

- 0 byte of the value `Data3`;
- 1st byte of the value `Data2`;
- 0 byte of the value `Data2`;
- 3rd byte of the value `Data1`;
- 2nd byte of the value `Data1`;
- 1st byte of the value `Data1`;
- 0 byte of the value `Data1`.

24.2.3.8. Upload of the table structure from the external data source

There is the option to upload the table structure from the external data source available for the developer (for example, copies of the real database).
To complete this procedure you should specify the paragraph Select from the list of external data source tables in the wizard of external data source tables while generating the table. Then you should specify the string for connection with the external database using the window Connection with the data source. While the connection string is generated you may use the wizard of the string for connection with the external data source. For this you should press the button "…" at the right of the field Connection string.

After connection with the external base is successfully completed the list with tables and fields of the attached data sources will be open. Then you should select the tables and the field that will be used by the designer to generate the structure of objects describing the current data source.

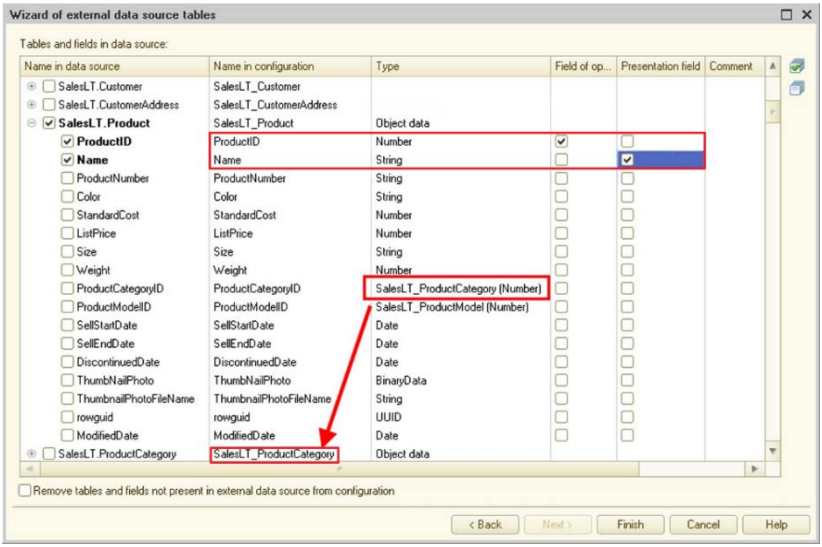


Fig. 286. External data source table wizard

While acquiring the structure of the external data source the system completes the following actions:

- It attempts to determine the type of the table with the data: the object-based or non-object-based data are located in the table. The table will be considered the object-based one if only one key field is specified for it and it will be the non-object-based one in the opposite case. If the system has made the error while specifying key fields you may manually change components of fields forming the key of the table. If the table has been determined as the object-based-one you may specify its field that generates the data view of such type. The view field should be manually specified.
- It transforms types of data source columns to types of 1C:Enterprise 8 that will be used to specify types of table attributes. All types from the external data source are transformed to the following types of 1C:Enterprise 8: number, string, date, boolean, UUID, binary data and the types related to object-based data source tables.

- It attempts to determine types for table fields. In this case the system tries to determine what type of the data is stored in the table column and in the case it may be considered to be the link to the data of the other table it specifies the relevant type in the column. If the system has chosen the incorrect type of the table column it may be manually changed.
- When you set up uploading you have the option to set up the following parameters of the uploaded data structure:
  - Name of the object in configuration -column Name in configuration;
  - Type of the uploaded attribute -column Type;
  - Specify the fields that are components of the key (for the object-based data) -column Key field;
  - Specify the field storing the view of the object -column View field;
  - Specify the field storing the version of the object (for the optimistic data lock) -column Version field;
  - Specify that the table or the attribute are available for read only -column Read only;
  - Specify the option to specify the value `NULL` in the attribute - column Allow NULL;
  - Set up the filling value for the attribute - column Filling value.

Then you should use the check box to mark tables and fields to be moved to the meta data of the application.
If the check box Delete tables and fields absent in the external data source from configuration is set up the tables and fields absent in the external data source (for example, tables or fields have been deleted in the external source) will be deleted from configuration once the table wizard stops.
Once the button Ready is pressed, the structure of the external data source will be uploaded.

24.2.4. Sample for generating the external data source

For example we have the database of dbf format access to which is to be acquired from the application.
The database consists of three tables (files):

- The table `products` (stored in the file products.dbf) that contains the information about products and consists of the following fields:
  - `id` -product ID (the key field). Type: `Number`.
  - `code` -product code. Type: `String`.
  - `name` -product name. Type: `String`.

    - article -product article. Type: String.
- The table price (stored in the file price.dbf) that contains the current prices of products and consists of the following fields:
    - product -ID of the product for which the price is stored. Type: Number.
    - price -product price. Type: Number.
- The table sales (stored in the file sales.dbf) that contains the data about each sale of the product specifying the price, the number and the amount. The table consists of the following fields:
    - product -ID of the sold product. Type: Number.
    - price -retail price of the product. Type: Number.
    - qty -the quantity of the sold product. Type: Number.
    - summa -retail price of the sold product. Type: Number.

Let's generate the necessary configuration objects to have access to all the data of these tables.

First, generate the external data source. Name it DBF. Then manually generate tables of the data source. At this, you should remember that the real name of the physical external data source table is specified in the property Name in the data source of the generated object. While we are generating the tables, name configuration objects as follows:

- the table products will be displayed into the object Products;
- the table price will be displayed into the object Prices;
- the table sales will be displayed into the object Sales.

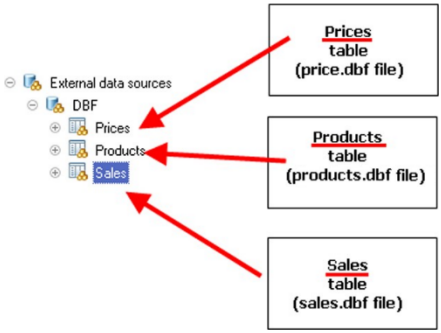


Fig. 287. Tables of the external data source

Now you should generate fields for each table, specify types of fields and assign fields of keys and views (if any).

The example will contain one object table (the table `Products`). Other tables will be non-object-based. You may see relations between fields of physical tables and fields of the configuration object as well as types of fields onfig. 288.

Name in data source	Name in configuration	Type	Field of o...	Prese...	Commer
price	Prices	Non-object data			Product
Product	Product	ExternalDataSourceTableRef.DBF.Product	☑		
Price	Price	Number	☐		
products	Products	Object data			List of pr
id	id	Number	☑	☐	
Code	Code	String	☐	☐	
Name	Name	String	☐	☑	
Article	Article	String	☐	☐	
sales	Sales	Non-object data			Sales of
Product	Product	Products (Number)	☑		
Price	Price	Number	☐		
Qty	Qty	Number	☐		
Total	Total	Number	☐		

Fig. 288. Structure of tables in the application

Now you should specify the subsystem containing the generated tables and other parameters of configuration objects (if necessary).If then you start the system in 1C:Enterprise 8 mode and correctly specify parameters of connection with the external data source, the navigation panel will contain the generated tables.

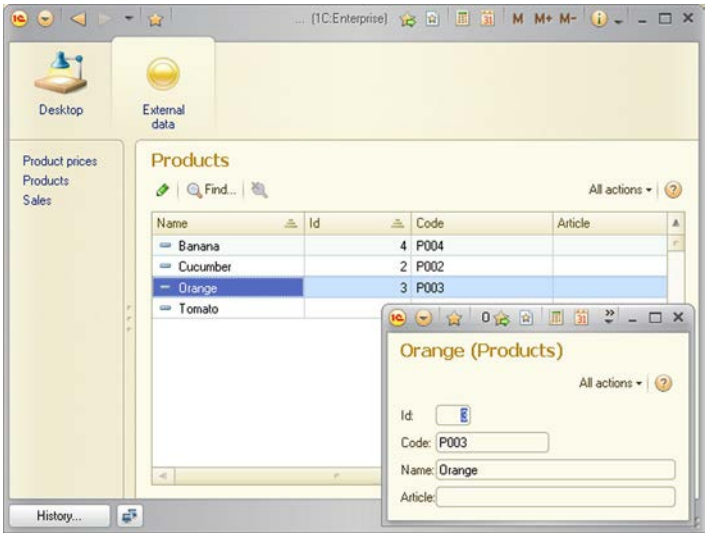


Fig. 289. Dynamic list of the external data source

24.2.5. ODBC connection string

24.2.5.1. General information

To connect with the external data source, you should generate the connection string that may contain either all connection parameters needed for the selected ODBC driver either indication to the generation description of the DSN data source (Data Source Name).

24.2.5.2. Full connection string

You may get the description of the connection string at the following address (in English): http://msdn.microsoft.com/en-us/library/ms722656.aspx. The following things should be noted in this description:

- Parameters of the connection string are recorded in pairs `KeyWord=Value` and divided by the symbol "`;`".
- If several parameters with the same keyword are found in the connection string, the last indication of the parameter in the connection string will be used. Thus, if such parameters as Username or Password are specified both in the connection string and in the dialog box Connec-

tion parameters, the values from the dialog box Connection parameters will be used because these parameters will be added to the specified connection string from the right side.

You may use the website http://www.connectionstrings.com/ to simplify generation of the connection string.

24.2.5.3. Description of the data source

Instead specifying the full connection string you may use extra utilities that may generate the description of the data source (DSN) and specifying the construction of the type DSN=NameDSN in connection parameters. Descriptions of the data source may be user and system ones. User descriptions of data sources are available on this computer and only for the user who has generated this description. The system description is generated by the system administrator and available for all users of this computer (if they have relevant authorizations).

24.2.5.4. The utility of ODBC data source administration

You should use the utility of ODBC data source administration to generate the description of the data source. This utility is located in Windows OS as follows: Control panel -Administration -Data sources (ODBC). At this, the utility of administration relevant to the version of the operating system is open from the Windows control panel. In Linux OS this utility is called ODBCConfig and available if the relevant package is installed (depending on the version of Linux OS).

You should remember that drivers and data sources are different for 32-bit and 64-bit ODBC in the 64-bit version of Windows OS. Thus, to generate the description of the data source you should use an administration utility relevant to the version of 1C:Enterprise system that will complete queries to the external data source. So, for example, if ODBC is called from the 64-bit version of 1C:Enterprise 8 server, you should use the 64-bit version of the administration utility. If a 32-bit version of 1C:Enterprise is used on 64-bit version of Windows OS, use a 32-bit administration utility.

Utilities of administration are located on the 64-bit version of Windows OS as follows:

- The 64-bit version: %SYSTEMROOT%\System32\odbcad32.exe.
- The 32-bit version: %SYSTEMROOT%\SysWOW64\odbcad32.exe.

24.3. Operation of OLAP external data source

24.3.1. General information

OLAP - is the data processing technology that prepares the aggregated information based on big data arrays structured according to the multidimensional principle. The data in analytical data bases (OLAP systems) are generated based on the data of transaction data processing systems (also called OLTP systems). The information in OLAP systems is given as the space (called **cube**) which axes are **dimensions** and some **measures** are located in nodes of this space. Each dimension of the cube is characterized by certain **members** of the dimension.

The closest (but not the exact!) analogue of the data storage in OLAP system may be considered to be the accumulation register where the register itself looks like the cube, the dimension of the type `CatalogRef` looks like the dimension of the cube, the directory contains members of the dimension and resources of the register look like measures of the cube.

In the 1C:Enterprise 8 system the cube of OLAP system is given as the model where:

- The cube is given as the cube in configuration;
- Dimensions and members of dimensions are given as two objects:
  - Dimensions are analogues of dimensions,
  - Tables of dimensions are used to describe the collection of dimension members.
- Measures are represented by resources.

Thus, the external source attached to the OLAP system contains cubes, each cube consists of dimension tables, dimensions themselves and resources. Tables of dimensions consist of fields. Resources of cubes may be numeral or string.

The object describing the external data source may be used as follows:

- as the data source for queries;
- as the data source in the data composition system;
- as the source for dynamic lists;
- be included into general attributes;
- entries of dimension tables and cubes may be displayed in managed forms of 1C:Enterprise 8 (ordinary forms are not supported for objects of external data sources);

- dimension tables of the external data source may be used as types of the infobase attributes;
- access rights may be applied to cubes, dimension tables, fields of dimension tables, dimensions and resources of external data sources;
- access to cubes, dimension tables, fields of the dimension table, dimensions and resources is possible from 1C:Enterprise language;
- cubes and dimension tables of the external data source may be included into subsystems;
- cubes, dimension tables and dimensions of the external data source may be included into functional options;
- you may generate characteristics for cubes of the external data source.

The XML feature for analytics is used to operate multidimensional external data sources (XML for Analysis, XMLA). The platform gets access to the data using HTTP queries to the web server that may be external (in relation to the OLAP system) or built in the OLAP server. Issues of setting access to the data of the OLAP server may be found in documents to the used system.
The connection string to the analytical external data source is combination of XMLA provider URL (for this OLAP system) and parameters used by 1C:Enterprise 8 to operate the source. The data of external data sources are available for read only.
Options of 1C:Enterprise 8 system are maximally considered when the following OLAP systems are used as the external data source (the property DBMS type of parameters for connection with the data source):

- IBM Infosphere Warehouse,
- Microsoft SQL Server Analysis Services,
- Oracle Essbase.

NOTE. When you operate the external data source, errors may appear if the property DBMS type of parameters for connection with the external data source contains the value not relevant to the system that is actually used.

When you use other DBMS, options for operation of external data sources depend on the DBMS. You should generate the connection string to connect with the external data source.
Also remember that you should specify connection with the external data source not only in the Designer (if the feature for importing table structures from the external data source is used) but also in 1C:Enterprise 8 to get the data.
Documents for OLAP systems:

- IBM Infosphere Warehouse:

- http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/topic/com.ibm.dwe.navigate.doc/welcome_db2warehouse.html (in English).
- Microsoft SQL Server Analysis Services:
  - 2008 version: http://technet.microsoft.com/ru-ru/library/bb522607(v=sql.100).aspx;
  - 2008 R2 version: http://technet.microsoft.com/ru-ru/library/bb522607(v=sql.105).aspx;
  - 2012 version: http://technet.microsoft.com/ru-ru/library/bb522607(v=sql.110).aspx.
- Oracle Essbase:
  - http://docs.oracle.com/cd/E17236_01/nav/portal_3.htm (in English).

24.3.2. The general operation design

You should comply with the following general design to use the information from external data sources based on 1C:Enterprise 8:

- Study the structure of the external data source and understand what information (cubes, dimension tables, dimensions and resources) are needed to operate algorithms or reports in 1C:Enterprise 8.
- Generate the object External data sources in the Designer and to generate the required structure for it. You can do it using the special helper.
- To implement usage of generated objects in the application.
- To set up parameters for connection to the external data source in the network where the application will be used. These parameters may be different from the ones used for downloading the structure of the external data source.

NOTE. Parameters of access to the external data source that have been set up in the designer will not be used by the system in the mode 1C:Enterprise 8.

24.3.3. Editing the structure of the external data source

24.3.3.1. General information

The leg with the name of the relevant external data source in the leg External data sources-<Name of the external data source>-Cubes of the configuration tree is used to operate tables of external data sources. Determination of the external data source consists of several steps:

1. Determination of the external data source as it is.
2. Determination of cubes;
3. Determination of dimension tables and fields for each dimension table;
4. Determination of dimensions for the cube;
5. Determination of resources for the cube.

At this, configuration may describe the smaller quantity of objects than contained in the real database but it is impossible to generate the object absent in the real database.

The structure of the external data source can be generated both in the manual mode and when this structure is uploaded with the special constructor.

24.3.3.2. External data source

You should indicate the name of the external data source when you generate it. This object is used to identify the group of cubes during calling of the data contained in it. The data source consists of one or several cubes that, in their turn, consist of other objects (dimension tables, dimensions, resources).

24.3.3.3. Cube of the external data source

The object edit window is open when the new cube is created.

The property of the table Name is used to identify the cube inside the application. For the system to know what physical cube of the external data source relates to the configuration object you should fill the property Name in the data source. The value in the property should exactly match the name of the cube in the external data source. Location of the data application from several cubes of the external data source in one cube is not supported.

24.3.3.4. Dimension table

Dimension table describes dimension members of the OLAP system cube. The property Name in the data source contains the exact name of the dimension or the hierarchical level in the data source. The feature View field

contains name of the dimension or hierarchical level property that is used by 1C:Enterprise 8 system to generate the view of the dimension table item.
If the dimension is hierarchical one, the hierarchical dimension table may be used for such dimension. The hierarchical dimension table is described using the following properties:

- Hierarchical -contains the flag that the dimension table describes the hierarchy in the data source.
- Name of the hierarchy in the data source -specifies the name of the hierarchy the dimension table describing this level relates to.
- Level number -in the case when the table describes any hierarchical level, this property contains the number of this level. For the hierarchical table the value of this property is 0 and the property Name of the hierarchy in the data source coincides with the value of the property Name in the data source.

Properties Flag of the non-filled parent and Value of the non-filled parent allow to specify for 1C:Enterprise 8 how to identify values of the highest hierarchical level.

24.3.3.5. Field of the dimension table

Fields of the dimension table describe what properties of dimensions will be available from the application. The property of the field Name is used to identify the property (fields of the dimension table) in the application. The property Name in the data source is used to specify relation between the field of the application and the attribute of the dimension. The value in this property should exactly match the name of the dimension attribute specified in the property Name in the data source of the parent object in this field. Combination of the data from several attributes of one or several dimensions in the external data source in the same field of the dimension table is not supported.
The field Type allows to specify the type of this field. The limited amount of types is available for selection. Field types may be:

- `Number`;
- `String`;
- `Date`;
- `Boolean`;
- `UUID`;
- `BinaryData`;
- types determined by dimension tables of external data sources.

If you have to specify the compound type for the field, only types `Number`, `String`, `Date`, `Boolean` may be included in this compound type.

When the data are received from external data sources, the system automatically transforms these data to the type specified at the relevant field in the application.

24.3.3.6. Dimensions

Dimension of the external data source describes dimension of the OLAP system cube. The type of the dimension may be only the link to the relevant dimension table. Actually, dimension tables and dimensions themselves are related one to one: one dimension is described by one dimension table.
Combination of the data from several dimensions of the external data sources in the same dimension field is not supported.

24.3.3.7. Resources

The resource of the external data source describes the fact of the OLAP system cube. The resource of the cube may be numeral or string. Values of resources are calculated using values of cube dimensions. The property Name in the data source should match the name of the fact (resource) of the external data source cube.
Combination of the data from several facts (resources) of the external data source in the same resource is not supported.

24.3.3.8. Downloading the cube structure from the external data source

There is the option to download the structure of the cube from the external data source available for the developer (for example, the copy of the real database).
To complete this procedure, you should specify the paragraph Select from the list of cubes in the external data source in the constructor of external data source cubes when generating the cube. Then you should specify the string for connection with the external database using the window Connection with the data source. While the connection string is generated you may use the wizard of the string for connection with the external data source. For this you should press the button "…" at the right of the field Connection string.

Once connection to the external base is successfully completed, the list of cubes, dimension tables, fields of dimension tables and resources of the attached data source will be open. Then you should select the requested cube as well as dimension tables, fields and resources that will be used by the Designer to generate the structure of objects describing the current data source.

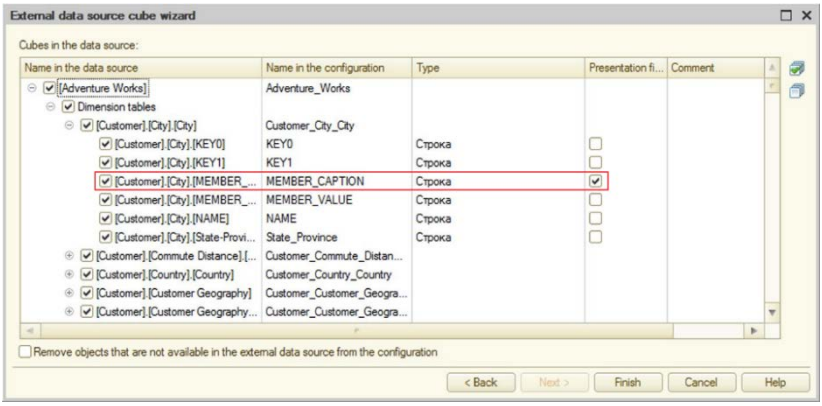


Fig. 290. Constructor of external data source cubes

While acquiring the structure of the external data source the system completes the following actions:

- It attempts to determine fields generating views for dimension tables. If the system has made a mistake, you may manually specify the view field.
- It transforms types of data source columns into types of 1C:Enterprise 8 that will be used to specify types of object attributes. All types from the external data source are transformed to the following types of 1C:Enterprise 8: number, string, date, boolean, UUID, binary data and types related to dimension tables of the data source.
- It attempts to determine types for fields of dimension tables. In this case the system tries to determine what type of the data is stored in the column of the dimension table and if it can be considered to be the link to the data of the other dimension table, it specifies the relevant type in the column. If the system has chosen the incorrect type of the table column it may be manually changed.

Then you should use check boxes to mark the objects to be moved to the meta data of the application.

If the check box Delete the object absent in the external data source from configuration is set up the objects (cubes, dimension tables, fields of the dimension table, dimensions, resources) absent in the external data source

(for example, the cube or the resource have been deleted in the external source) will be deleted from configuration after the constructor of cubes finishes operation.
Once the button Ready is pressed, the structure of the external data source will be uploaded.

24.3.4. Limitations of the query language when the analytical external data source is used

When you operate cubes and dimension tables in the query language, you should consider the following limitations:

Expression	Microsoft SQL Server Analysis Services	Oracle Essbase	IBM Infosphere Warehouse
ALLOWED	–	–	–
DISTINCT	–	–	–
GROUP	–	–	–
HAVING	–	–	–
JOIN	–	–	–
MERGE	–	–	–
Subqueries	–	–	–
FOR UPDATE	–	–	–
EMPTYTABLE	–	–	–
YEAR	Yes	–	–
QUARTER	Yes	–	–
MONTH	Yes	–	–
DAYOFYEAR	Yes	–	–
DAY	Yes	–	–
WEEK	Yes	–	–
WEEKDAY	Yes	–	–
HOUR	Yes	–	–
MINUTE	Yes	–	–
SECOND	Yes	–	–
BEGINOFPERIOD	Yes	–	–

Expression	Microsoft SQL Server Analysis Services	Oracle Essbase	IBM Infosphere Warehouse
ENDOFPERIOD	Yes	–	–
DATEADD	Yes	–	–
DATEDIFF	Yes	–	–

Where,

- - -the specified expression is not supported during operation of the relevant OLAP system;
- **Yes** -the specified expression is supported during operation of the relevant OLAP system.

24.3.5. String for connection with the OLAP server

The string for connection with the OLAP server is of the following kind:

```
http://<OLAP host address>:<port>/<source>?<parameters>
```

Where:

- OLAP host address, port, source -address of access to the OLAP system generated under the rules described in documents of the certain OLAP system.
- parameters -parameters used by 1C:Enterprise 8 for access to the data of the OLAP system. Parameters are set up in the type `Parameters=Value`. Parameters are divided by the symbol &. The following parameters are used:
  - `ProviderName` -name of the XMLA service in the OLAP source;
  - `DataSourceName` -name of the OLAP source;
  - `Catalog`-name of the directory or the database in the OLAP source.

Examples of connection strings are given below:

For Microsoft SQL Server Analysis Services:

```
http://localhost:80/msolap/msmdpump.dll?ProviderName=Microsoft
Analysis Services& DataSourceName=host&Catalog=Adventure Works DW
```

For Oracle Essbase:

```
http://localhost:13080/aps/XMLA?ProviderName=Essbase XML for
Analysis&DataSourceName=host&Catalog=Sample
```

24.4. Operation of procedures and functions in external data sources

24.4.1. General information

Components of the query language and the data composition system gives great opportunities for the data processing but does not include specific options of different DBMS that may include functions, sequences, stored procedures and different specific DBMS constructions. At the same time application of specific DBMS options may be critically important during operation of external data sources. The example may be the object-based DBMS table which key is retrieved from the sequence. Functions and procedures may not return anything (the procedure) as well as return the single value or table.

The concept of the **external data source function** exists top operate such DBMS objects in 1C:Enterprise 8. Functions of 1C:Enterprise 8 may belong to the following types:

1. The table of the expression type in the relational data source;
2. The function that does not return values (the procedure);
3. The function that returns any value.

Any function described above may take parameters, the quantity of parameters may be both docked and variable. The function of the external data source may be both the layer between the real function (or the saved procedure) of DBMS and expressly describe some construction that will be placed into the completed query.

Functions may be used in the 1C:Enterprise language, the query language and the data composition system. For functions you may specify access rights regulating usage of these functions.

24.4.2. The general operation design

To use functions external data sources in the system based on 1C:Enterprise 8 you should comply with the following general design:

- To study the structure of the external data source and to understand what functions, procedures or constructions on the DBMS language are needed to operate algorithms or reports in 1C:Enterprise 8.
- To generate the object External data sources in the Designer and to generate subordinate functions for it. You can do it using the special helper.
- To implement usage of generated objects in the application.
- To set up parameters for connection to the external data source in the network where the application will be used. These parameters may be different from the ones used for downloading the structure of the external data source.

NOTE. Parameters of access to the external data source that have been set up in the designer will not be used by the system in the mode 1C:Enterprise 8.

24.4.3. Editing functions

24.4.3.1. General information

The log with the name of the relevant external data source in the log External data sources-<Name of the external data source>-Functions of the configuration tree is used to operate functions of external data sources. Determination of the external data source consists of the following steps:

1. Determination of the external data source;
2. Determination of data source functions;
3. If procedures that return tables exist in the external data source you should also need to determine tables of the external data source belonging to the expression type.

At this, configuration may describe the lesser quantity of functions and procedures than the real database has. You may also describe some functions (on the language of the used DBMS) that is absent in the real database. At this, you should remember the following features:

- If it is specified for the function (as the configuration object) that the expression returns the value then:
  - If used in the 1C:Enterprise language -the expression is placed into the construction `select<expression>`.

- If used in the query language (and in the data composition system) -call of the function is replaced by the expression itself.

- If it is specified for the function (as the configuration object) that it does not return the value then:
  - When used in the 1C:Enterprise language -depending on the DBMS either the call using the construction `select <expression>`, or the controlling ODBC sequence `call` will be used. Due to that it is not recommended to use the unspecified expression for the function that does not return the value to achieve the maximum compatibility with various DBMS.
  - - Not supported when used in the query language (or the data composition system).

The structure of the external data source can be generated both in the manual mode and when this structure is uploaded with the special constructor.

24.4.3.2. External data source

You should indicate the name of the external data source when you generate it. This object is used to identify functions when called. The data source consists of one or several functions, and the situation is possible when expression tables are located in the external data source.

If the object of the external data source is used only to call procedures and functions the value of the property Data lock management mode is meaningless. If the lock management mode is set up to the value Automatic and manageable for the external data source, the lock mode for each table is determined by the value of the relevant table property. If the certain special lock management mode is selected for the external data source, the property of the external data source table with the same name is ignored.

24.4.3.3. Function of the external data source

The window of the external data source function constructor is open when the new function is generated. If the manual generation mode is selected - the property palette of the generated external data source function will be open.

The property of the function Name is used to identify the function when it is called. The property Expression in the data source will contain the expression that will be used in the real query in the database when the function is used. You may specify not only the call to the real function or the saved procedure of the database but also write the expression on the language of the external data source DBMS in this property.

The property Returns the value actually specifies what is described in the configuratio object:

- `True` - the function or the expression that returns the value. In this case, you should alsp specify what type of the value will be returned by this expression.
- `False` -the procedure or the expression that does not return the value.

The procedure or the function may take parameters.
The function may be the expression on the DBMS language, for example, the function that acquires the following value of the sequence (`SimpleSequence`) will look as follows:

```
NEXT VALUE FOR dbo.SimpleSequence
```

If you need to acquire the following value of the sequence from the 1C:Enterprise language it will look as follows:

```
KeyValue = ExternalDataSources.ExternalData.NextKeyValue();
```

In this example `ExternalData` -name of the external data source where the function `NextKeyValue()` is described. The function `NextKeyValue()` is described above.
The function also may be written on the language of the used DBMS. At this, it is recommended to insert the expression into round brackets to avoid various side effects during execution of the real query. You should remember that during transformation of the query from the query language of the 1C:Enterprise 8 platform into the SQL dialect of the certain DBMS, functions are placed directly into the text of the retrieved query (parameters of functions are automatically placed).

24.4.3.4. The external data source table of the expression type

Functions may return not only values but also tables. In this case the function should be generated as the **table of the expression type**. You should also describe fields of the returned data in the same manner as standard fields of the external data source table.

24.4.3.5. Downloading functions from the external data source

There is the option to upload the table structure from the external data source available for the developer (for example, copies of the real database).
To perform this procedure you should mark the point Select from the list of external data source functions in the constructor of the external data source function while generating the table. Then you should specify the string for connection with the external database using the window Connection with

the data source. While the connection string is generated you may use the wizard of the string for connection with the external data source. For this you should press the button "..." at the right of the field Connection string.
After connection to the external base is successfully completed, the list of procedures, functions and sequences of the attached data source will be opened. Then you should select the objects that will be used by the Designer to generate the structure of objects that describe the current data source.

There is the option to specify under which name the function will be generated in the configuration 1C:Enterprise 8.
If the check mark Delete functions absent in the external data source from configuration is set up the functions absent in the external data source (for example, functions have been deleted in the external source) will be deleted from configuration once function constructor stops operating.
After the button Ready is pressed, the structure of external data source functions will be downloaded.

24.4.3.6. Parameters of external data source functions

Formal parameters of external data source functions are described as `&n` where `n` -is the number of the parameter. Parameters are numbered since 1. If parameters are not necessary, they are put into curved brackets (`{}`). Presence of non-obligatory parameters are determined every time when the procedure or the function is actually called. `NULL` (for every non-obligatory parameter) replaces actual parameters absent in the non-obligatory block after the call.

```
//function with two parameters
dbo.uspGetWhereUsedProductID(&1, &2)
//function with two non-obligatory parameters
dbo.uspGetWhereUsedProductID(&1{, &2, &3})
```

If the procedure or the function may take the variable number of parameters this construction is described as `&n[]`. In this case, actual parameters will be placed into the expression instead of the construction `&n[]` with the comma. Indication on the option to transfer the variable number of parameters may be only the last one in the list of formal parameters.

```
//function where from 2 to the amount of parameters may be passed,
// limited by the used DBMS
dbo.uspGetWhereUsedProductID(&1, &2[])
```

24.5. Management of external data sources

Before you start using the data from external sources in the application, you should set up connection parameters of used external sources. For this, the standard function Management of external data sources called from the window All functions -Standard functions.

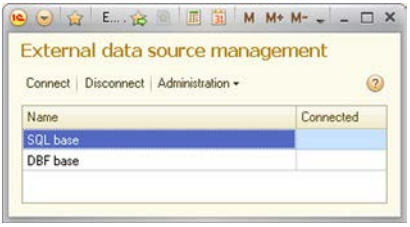


Fig. 291. Management of external data sources

This form lists all external data sources generated in the Designer and available for the user who has opened the form.
The column Attached specifies the condition of connection to the data source in this session.
Using the menu Administration you may specify the general connection parameters (command Change general parameters...) and parameters for

connection of certain users if these parameters are different from general parameters (command Change user parameters...).

Fig. 292. Connection parameters

The check box in the left part of the form (before names of parameters) means that this parameter is used in this set of connection parameters. At this, parameters are analyzed as follows:

- parameters installed for the session using the method `SetSessionConnectionParameters()` of the external data source manager;
- parameters set up for the user during interactive setup or using the method `SetUserConnectionParameters()` of the external data source manager;
- general parameters set up interactively or using call of the method `SetCommonConnectionParameters()` of the external data source manager.

The resultant set of parameters will be used for connection with the external data source. If any parameter is specified in several settings the value of the parameter that will be found earlier in the chain described above will be used. For example, connection parameters may be combined as follows:

- the user and its password is specified in connection parameters of the session;
- the string for connection with the database of the external source is specified in general parameters;
- the other connection string that may be used to test the application is specified in the parameter of the certain user.

When you specify the connection string you should remember that the external data will be accessed not from the computer where the setting is carried out but from the other computer.
When you press the button Connect the dialog box is opened where you can set up (or specify) parameters for connection to the selected external data source.

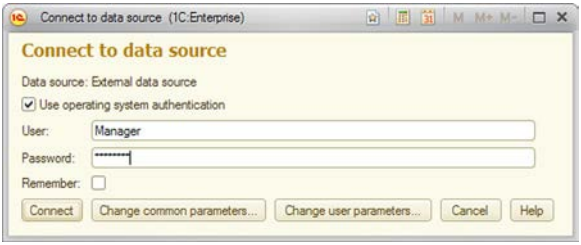


Fig. 293. Connection parameters

The ODBC driver and components of the connection string determine the need in specification of the user and the password in this dialog box. The situations can occur when the login and the password are not needed. If the user is not the administrator of the external data source, buttons General parameters... and User parameters... are unavailable for it.
The check box Use operating system authentication is available only if the user has the right `SessionOSAuthenticationChange`.
After you press the button Connect, connection to the external data source occurs. If connection is successful, components of the column Connected in the list of external data sources are changed.
When you press the button Disconnect, 1C:Enterprise 8 will be disconnected from the selected external data source.

24.6. Connection to the external data source in the 1C:Enterprise 8 mode

When you perform any procedure during which you need to get the data from the external data source, the attempt of connection occurs if connection to this data source has not been completed yet. If connection is successfully completed, selected actions are continued. If connection is not done, exception is called.

If the exception related to connection to the external data source occurs in the client application, the user is offered with the dialog box where it may specify connection parameters (if the user has rights to modify parameters) and repeat connection. If connection is successful, the user will be offered to repeat the action during which the connection error has occurred.

NOTE. In the mode of compatibility with the version 8.2.13, during operation of the dynamic list, in some cases the dialog box for connection to the external data may not be displayed.

There is also the option of manual connection. For this, the standard function Management of external data sources is used.

24.7. Features of external data sources

24.7.1. Place for completion of queries to external data sources

Queries to external data sources are completed:

- in the file mode -on the computer with the client application;
- in the file mode with access via the web server -on the computer where the web server is extended;
- in the client/server mode -on the computer where the following is completed:
  - the service for operation of external data sources via ODBC - for access to relational data sources;
  - the service for operation of external data sources via XMLA - for access to analytical data sources (OLAP systems).

24.7.2. Usage of external data sources

If one external data source contains both the data of relational data sources and the data of analytical data sources, they cannot be used simultaneously. It is explained by the fact that one external data source may be at one moment connected either to relational or to analytical DBMS.

24.7.2.1. In queries

24.7.2.1.1. To relational external sources

Tables of external data sources may be used as sources of queries. At this, the table of the external data source is described as follows:

```
ExternalDataSource.<source name>.Table.<table name>
```

Example:

```
SELECT
  Products.Code,
  Products.Name,
  Products.Article
FROM
  ExternalDataSource.DBF.Table.Products AS Products
```

In this example, fields `Code`, `Name` and `Article` are selected from the table `Products` of the external data source `DBF`.

Temporary tables may be used in the query to the external data source. At this, the attempt to generate the temporary table directly in the database related to the external data source occurs. If DBMS does not support generation of external data sources - exception will be called. The name for the temporary table, during its calling, is generated as follows:

```
ExternalDataSource.<Name of the external data
source>.TempTable.<Name of the temporary table>
```

Example:

```
SELECT Name, ProductID
MOVE ExternalDataSource.AdventureWorks.TempTable.Rest
FROM & ValuesTable

SELECT Name, ProductID
MOVE ExternalDataSource.AdventureWorks.TempTable.Rest
FROM ExternalDataSource.AdventureWorks.Table.Production_Balance
```

When external data sources are used in queries, you should remember the following limitations:

- One data source may be used in one query. For example, the external data source and the data of the 1C:Enterprise 8 infobase cannot be used simultaneously.

24.7.2.1.2. To analytical external sources

Objects of analytical external data sources may be used as sources of queries. At this, the cube of the external data source is described as follows:

```
ExternalDataSource.<Name of the source>.Cube.<Name of the cube>
```

Example:

```
SELECT FIRST 10
AdventureWorks.InternetSalesAmount
FROM
ExternalDataSource.OLAP.Cube.AdventureWorks AS AdventureWorks
```

In this example, first 10 resources of `InternetSalesAmount` are acquired from the cube `AdventureWorks` of the external data source `OLAP`.

IMPORTANT! It is not recommended to get **all** entries from the cube of the external data source. The amount of entries is very large and may cause overrun of the system.

The dimension table for the cube of the external data source is described as follows:

```
ExternalDataSource.<Name of the source>.Cube.<Name of the
cube>.DimensionTable.<Name of the dimension table>
```

Example:

```
SELECT
AdventureWorks DimensionTableProduct.Link
FROM
ExternalDataSource.OLAP.Cube.AdventureWorks.DimensionTable.ProductA
S AdventureWorks DimensionTableProduct
```

In this example, the list of links to members of the dimension table `Product` is acquired from the cube `AdventureWorks` of the external data source `OLAP`.

Temporary tables may be used in the query to the external data source. At this, the attempt to generate the temporary table directly in the database related to the external data source occurs. If DBMS does not support generation of external data sources - exception will be called. The name for the temporary table, during its calling, is generated as follows:

```
ExternalDataSource.<Name of the external data
source>.TempTable.<Name of the temporary table>
```

When external data sources are used in queries, you should remember the following limitations:

- One data source may be used in one query. For example, the external data source and the data of the 1C:Enterprise 8 infobase cannot be used simultaneously.

24.7.2.2. In the data composition system

Sets of the data acquiring the data from various uniform data sources may be used in the data composition system. Thus, you may generate the set of data describing the list of products. The data in this set will be acquired from the infobase of 1C:Enterprise 8. The other set of the data may describe the data acquired from the external data source, for example, the table of product sales from the external database.

Then you may set up connections of data sets in the constructor of the data composition scheme and to get the information in the report from both 1C:Enterprise 8 and the external data source.

24.7.3. Rules of value transformation

When the data are received from external data sources, the system automatically transforms these data to the type specified at the relevant field in the application. Transformation is done according to the following rules:

- Type`String`:
  - The string value is transformed into the string value.
  - The numerical value is transformed into the string value (local setting of the operating system on the computer where the query to the external data source is done is used for transformation).
  - The value of the date is transformed into the string value (the local settings of the operating system on the computer where the query to the external data source is done is used).
  - The value of the UUID is transformed into the string view of he UUID.
  - The binary data (and the data of other types) are transformed into the string that is the textual view of the transformed data in the hexadecimal type.
- Type `Number`:
  - The numerical value is transformed into the number.
  - For the string value, the string is attempted to be transformed into the number according to the local settings of the operating system on the computer where the query to the external data source is completed.
  - For the fields which accuracy in the base of the source exceeds accuracy of platform types, it is recommended to either expressly use `EXPRESS` in queries or to generate the view in the base so that accuracy of fields matches accuracy in the designer.

    - Exception occurs for other types.
- Type `Boolean`:
    - The numerical value is transformed into the value `False` if it is equal to `0` and into the value `True` -for any other value in the external source.
    - The binary data are transformed according to the value of the first byte's type. If the value of the first byte is `0` the result of transformation will be the value `False` and the value `True` in case of any other value.
    - Exception occurs for other types.
- Type `BinaryData`:
    - All types of values are transformed into the binary data.
- Type `Date`:
    - The value of the date is transformed into the value of the type `Date`.
    - Exception occurs for other types.

24.7.4. The external data source that is the component of the separator

If the external data source is the component of the general attribute you should remember the following features:

- The structure of external data source tables is not changed so the data themselves are not separated.
- Parameters for connection with the external data source are stored broken down by values of separators containing the external data source.
- The unused separator is the individual value broken down by storage of parameters for connection with the external data source.

Thus, when you include the external data source in the separator, you will get the option to implement one of the following usage scenarios:

- The data available for users of various data regions are located in different databases which table structures coincide. As connection parameters are stored broken down by values of the separator, there is the option to set up operation with the necessary database for each data region.
- The data available for users of various regions are located in the same database but access to the data is managed by the DBMS means (username and password). In this case there is the option to specify own

name and password of the user on whose behalf the data is accessed for each data region.

The given scenarios are not only ones. They are given as the example of external data sources included in separators.

Chapter 25. Data separation feature

25.1. General information

Data separation feature - is the specific feature that allows to divide all stored data and the operation of the application into individual parts. If the property Data separation of the general attribute is set up into the value Separate, the data separation mode is on. Further:

- the behaviour of configuration objects included into the general attribute (hereinafter referred to as - the **separator**) changes;
- the current value of the separator and the flag of this separator's use is determined for each separator in the infobase;
- configuration objects included in such general attribute are called **separated configuration objects**;
- configuration with such general attributes is considered to be **separated configuration**;
- the infobase with the separated configuration is called **separated infobase**;
- the infobase data available for selected values of separators and the data of shared configuration objects are called **data region**.

In all procedures of database entry reading 1C:Enterprise 8 automatically selects only the entries where values of used separators coincide with current values of separators. When the data are recorded, the platform checks that the recorded data contain values of used separators equal to current values of separators or equal to default values for relevant types. If it is not true, exception is called.

Example:

```
// The object is initially recorded with the value of separator
Subscriber = 1
Object = LinkToObject.GetObject();

// Change the value of the separator in the current session
SessionParameters.SubscriberValue = 2;

// The object will not be recorded and exception will be called
Object.Write();
```

If the recorded data contain values of separators equal to default values for relevant types, values of used separators in the current session will be recorded into separators of the object. For separators not used in the current session:

- filter for these separators is not done during data reading;
- when the data are modified the values included into the recorded data are used.

The separator may contain the following configuration objects:

- constants
- Catalogs
- Documents
- sequences
- Document journals
- Charts of characteristic types
- Charts of accounts
- Charts of calculation types
- Business processes
- Tasks
- Information registers
- Accumulation registers
- Accounting registers
- Calculation registers
- Exchange plans
- scheduled jobs,
- infobase users.

You should remember that sequences and document journals are separated objects only if they include separated configuration objects. Components of separators for objects included into the sequence or the document journal should be identical.

The separator in the mode Individually and together may be used in data access restrictions.

Type of the separator **may not be composite**.
If the general attribute is used as the separator, it will have several additional properties affecting the behavior of the system and the behavior of configuration objects included into the separator also changes. More details of these properties will be considered in next sections.

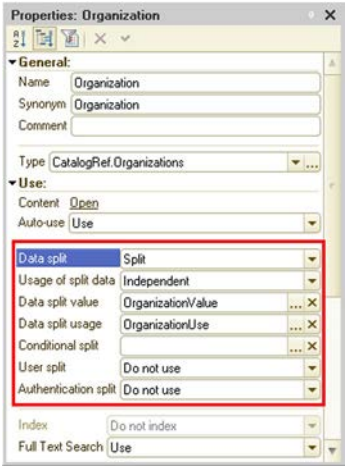


Fig. 294. General attribute with separation

IMPORTANT! If you use the separator with the type `String` the language of regional infobase settings for operation of Microsoft SQL Server cannot be changed.
IMPORTANT! Data separation is not supported on DBMS Microsoft SQL Server 2000, IBM DB2 9.1 as well as on DBMS IBM DB2 of other versions if comparison of strings sensitive to the register of letters is installed in the DBMS (parameter `COLLATION_SEQUENCE` is not equal to the value `UCA500R1_LROOT_AN_CX_EX_FX_HX_NX_S2`).

25.2. Properties of the general attribute

25.2.1. Separated data use

The property Separated data use determines the option for use of the separated data in configuration object if the separator is not used in 1C:Enterprise 8 mode:

- The value Individually -any call to separated objects is forbidden (except exchange plans), if the separator is not used in the current session of the infobase operation. At this, the separator is unavailable in the object-based model, in the query language, in the format of upload into XML and XDTO as well as in data access restrictions. When the separated data are recorded, the system automatically fills values of the general attributes with the values set up in the current session. In this mode configuration extensions may be added to the data of native configuration objects and generate native objects storing the data in the infobase. This mode may be used when the application uses only the data of the single data region.

IMPORTANT! In this mode **links of objects** in the database **may be similar** for objects recorded in different data regions.

- The value Individually and together -operation of separated objects is allowed regardless the use of the separator in the session. The data region determined by values of used separators will be available for this mode. At this, the separator is available in the object-based model, in the query language, in the format of upload to XML and XDTO as well as in data access restrictions. In this mode, predetermined elements for the objects included in such separator cannot be generated. If the session does not use all separators that include the object, operation of this objects will be less effective. This mode does not support any data extension.
  This mode may be used in the cases when the application mostly operate the data of the single data region (for example, during input of documents) and you may use the data of several or all regions in some modes, for example, to obtain the consolidated statements.

IMPORTANT! In this mode **links of objects** in the database **cannot be similar** for different objects recorded in different data regions.

25.2.2. The value of the separator and the flag of use

The property Data separation value determines the parameter of the session that stores **the value of the separator** set up in the current session. The type of the session parameters should exactly match the type of the separator.
The property Data separation use determines the parameter of the session (type `Boolean`) that manages the use of the separator. If the value of the specified session parameter is `True`, it means that **the separator is used** in this session.

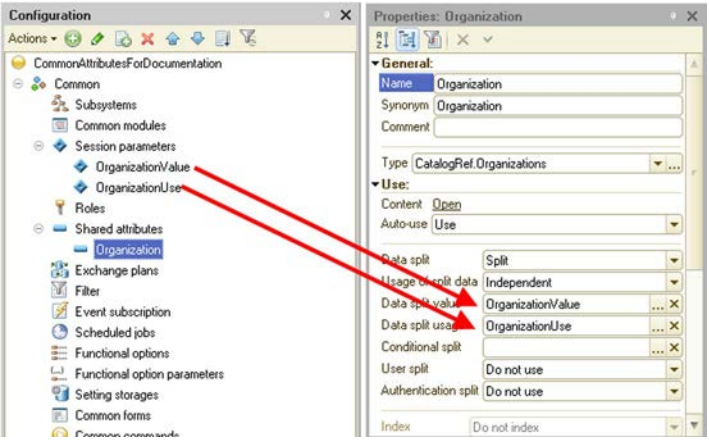


Fig. 295. Parameters of the session and the general attribute

You can change values of session parameters during the system operation if the user on whose behalf the session operates has rights to change necessary session parameters. 1C:Enterprise 8 does not warrant integrity of the application data once session parameters are changed.
When the value of the session parameters referred to by at least one separator is changed from the 1C:Enterprise language, the following actions are needed:

- object cache is cleared;
- repeated values are removed on both the side of the client and the side of the server.

The value of the separator may be specified in the command start string of 1C:Enterprise 8 using the parameter Z or in the connection string using the parameter Zn.

25.2.3. Separation of users

The property Separation of users determines what part of the user list will be available depending on whether this separator is used or not. If the property is set up in the value Don't use the single user list is used for any values of the separator. Separation of users is used for the program access to the user list as well as in the Designer when the user list is edited.
You may use the following table (related to one separator) to determine whether the user is available in the session or not:

	The separator is used in the session	The separator is not used in the session
The value is set up for the user	Available if values of the separator are equal	Available
The value is not set up for the user	Unavailable	Available

When the user is edited there is the option to specify values for all separators set up in configuration and not only those for whom the property Separation of users is set up in the value Separate. Values of separators specified in properties of the infobase user will be used to set up values of separators when the session of this user starts. Values of separators in the command string or the connection string has a higher priority than values specified in user properties if the user has rights to change relative session parameters.

25.2.4. Separation of authentication and

The property Separation of authentication is used to manage the option to generate users with the same name for different data regions. If the property is set up in the value Don't use uniqueness of usernames is traced down for all data regions. If the property is set up in the value Separate there is the option to generate users with the same name in different data regions. For example, you may generate several users Administrator that will be different only by values of separators (including for the non-set value of the separator).

If there is the separator Subscriber (type Number) and the value of the property Separation of authentication for it is set up in the value Separate there is the option to generate the following user list:

Username	Value of Subscriber separator
Administrator	Not assigned
Administrator	1634
Administrator	2245
Administrator	1245

If the value of the property Separation of authentication for this separator is set up in the value Don't use you cannot generate such list. Usernames should be unique, for example:

Username	Value of Subscriber separator
IBAdministrator	Not assigned
1634Administrator	1634
2245Administrator	2245
1245Administrator	1245

25.2.5. Separation of configuration extensions

The property Separation of configuration extensions is used to manage the option for separation of extensions between various data regions.
If the property is set up in the value Separate it means that only the extensions that have been attached with the same value of the separator as the user will be available for the user. If the property is set up in the value Don't use values of such separator will not be considered when extensions are attached and availability of extensions for the current user is determined.
When the value of this property for the separator is changed you should consider the following features:

- When the value of the property Separate is replaced by Don't use this separator stops being considered when availability of extensions that have already been downloaded is determined.
- When the value of the property Don't use is replaced by Separate all existing extensions become available in the data region that matches the special value of the separator describing the disconnected separator.

It is recommended to disable separation of extensions for the separator that is in the mode Individually and together.

25.2.6. Conditional separation

25.2.6.1. General description

Conditional separation is needed when it is suggested to use the application both in separated and shared options. For example, the application may be used for autonomous operation - in this case separators are not used but they are provided in the application and it may be used to operate some amount of subscribers not related to each other (with the individual data set in the infobase). This behavior may be implemented using conditional separation: in case of autonomous use separation will be off and it will be on - in the separated mode.

Conditional separation is made as follows:

- for the separator (or the object included into the separator) the object that stores condition of separation for this separator may be set up;
- when you change the value of this object, you may turn on or off separation for this separator for all (if conditional separation is set up for the separator itself) or for selected objects (if conditional separation is set up for the object included into the separator);
- if the separator is conditionally off it will use the default value for the type of the separator as the value of the separator.

NOTE. For each separator the value of conditional separation may not be directly or indirectly depend on the value of the separator itself.

The recursive management of conditional separation is possible when conditional separation is set up for the separator used to determine conditional separation of the other separator etc. In this case, determination of conditional separation will start from the separator which value is not conditionally separated (but it affects other objects involved into conditional separation) and then using the bound separators until conditional separation is not determined for the separation which condition does not affect any separator anymore. It is impossible to generate loopable conditional separation.

Conditional separation may turn on or off separation as follows:

- For all objects included into the separator. In this case you should use the property of the separator Conditional separation.
- For individual objects included into the separator. For this you should use the column Conditional separation in the window for editing components of the separator.
- By combination of properties described above.

When you arrange conditional separation you should consider the following features:

- The following items may be used as the object storing the value of conditional separation:
  - The constant of the type `Boolean` not included into the separator for which conditional separation is set up.
  - The attribute of the type `Boolean` in the reference configuration object that:
    - Is not included into the separator for which conditional separation is set up.
    - It should be used as the type of the other separator and the separator of this type should be only one.
- When the system is started, values of session parameters storing the flag of use and values of separators is initially set up according to starting command-line parameters or values specified for the infobase user.
- If separation of the general attribute is conditionally off then:
  - When the data are read and recorded, current values of session parameters responsible for the value and the use of the separator are ignored. At this, the following values are used:
    - The flag of use -`True` (the separator is used).
    - The value of the separator - the default value for the type of the separator.
  - When infobase users are read and recorded, current values of session parameters responsible for the value and the use of the separator are ignored. At this, the following values are used:
    - The flag of use -`False` (the separator is not used).
    - The value of the separator - is not used.
- If the value determining conditional separation is not read or ambiguous, exception will be given.
- If the separator used in conditional separation of the other separator is not used the other separator is also considered to be unused.
- If all separators are turned off in the session (using conditional separation) actions with the infobase that are unavailable if at least one separator is used become available.

If conditional separation is set up both in the property of the general attribute (the property of the general attribute Conditional separation) and using the column Conditional separation in components of the general attribute, one of the following values will be used as the value of the separator:

- The current value of the separator (set up in the relevant session parameter) when separation is conditionally on and the object (the attribute or the constant) specified in the relevant column of general attribute components and the object (the attribute or the constant) specified in the property of the separator itself.

- The default value for the type of the separator if conditional separation is turned off by any method.

NOTE. Conditional separation is not used in the Designer. It is believed that attributes are always conditionally **turned on** in the session of the Designer.

25.2.6.2. Conditional separation set up in properties of the general attribute

25.2.6.2.1. Using the constant

The separator may be completely turned off using the constant. The constant should have the type `Boolean` and should not be separated by the attribute.

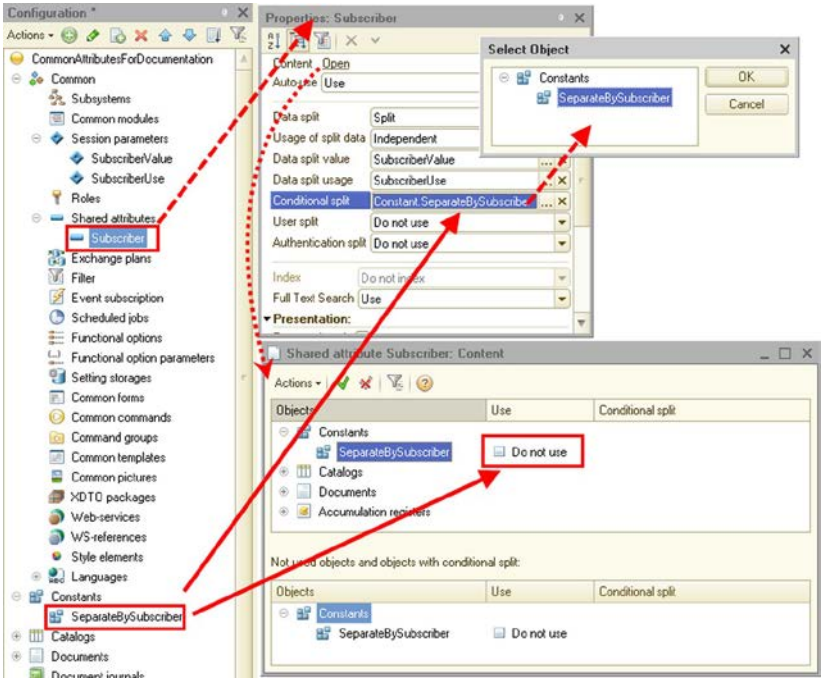


Fig. 296. Conditional separation using the constant

IMPORTANT! When 1C:Enterprise 8 is initially started (once conditional separation is added) the value of the constant will be `False` and separation will be **turned off**.

Let's look at this case in more details.

There is the general attribute `Subscriber` with the type `Number`. This attribute includes the directory `Products` containing the following:

Code	Name	Subscriber
1	Sandals	0
2	Boots	1
3	Felt boots	2
4	Trainers	1
5	Flip-flops	0

Conditional separation managed by the constant `SeparationForSubscribers` is used for the general attribute. If the value of the constant is `False` it means that the separator is disabled (separation is **turned off**). If the value of the constant is `True` it means that the separator operates (separation is **turned on**).

Thus, the following situations may be found out:

- The value of the separator `Subscriber` is `1`, the separator is used, the value of the constant `SeparationForSubscribers` is `True`. In this session **separation** is conditionally **turned on** (for the separator `Subscriber`). In this situation the following items are available in the directory:

Code	Name	Subscriber
2	Boots	1
4	Trainers	1

- The value of the constant `SeparationForSubscribers` is `False`. In this session **separation** is conditionally **turned off** (for the separator `Subscriber`). In this situation, regardless the value of session parameters `SubscriberUse` and `SubscriberValue`, the following items are available in the directory:

Code	Name	Subscriber
1	Sandals	0
5	Flip-flops	0

25.2.6.2.2. Using the attribute of the object

A more complex option of conditional separation management is used if the data of one separator is used for conditional separation of the other separator. Then, various behaviors of the separator describing the other data region is possible within various data regions.

For example, there is the set of configuration objects describing the regulatory and legislation data (hereinafter referred to as the RLD): directories Banks, Currencies, VAT rates. You should implement the following option:

- you can make accounting for various company groups in the infobase;
- each company groups consists of organizations;
- components of RLD objects may be common for all organizations of the company group as well as individual for each organization.

Implementation of such design will require the following configuration objects:

- the directory `CompanyGroups` with the attribute `CommonRLD` of the type `Boolean`;
- the directory `Organizations`;
- the directory `Banks`;
- the directory `Currencies`;
- the directory `VATRates`;
- the separator `CompanyGroups of the` type `CatalogRef.CompanyGroups`;
- the separator `Organizations` of the type `CatalogRef.Organizations`. This separator includes only the objects describing the RLD:
  - the directory `Banks`,
  - the directory `Currencies`,
  - the directory `VATRates`.

The separator `CompanyGroup` include all considered directories (except the directory `CompanyGroups`) because we need its own list of organizations and components of the RLD for each company group.

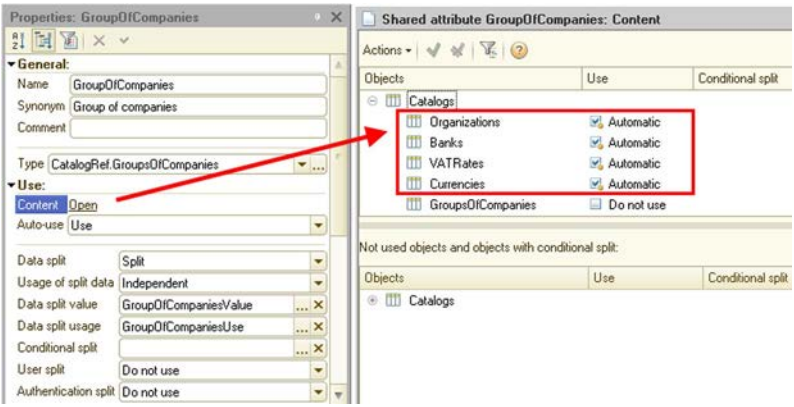


Fig. 297. Components of the separator "CompanyGroup"

The separator `Organization` includes only directories `Banks`, `Currencies` and `VATRates`. These directories are the RLD and two options of usage are needed for them:

- for each organization of the selected company group,
- for all organizations of the selected company group.

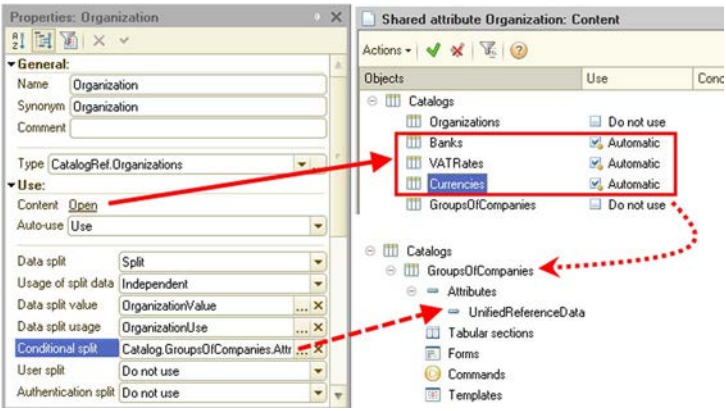


Fig. 298. Components of the separator "Organization"

The directory CompanyGroups is not included into the separator Organization because it is required by the rules for arrangement of conditional separation.
Now you may specify the value of the attribute CommonRLD for each value of the separator CompanyGroup (the item of the directory CompanyGroups):

- The attribute is set up in the value False -conditional separation is **turned off,** and the **common** set of values for directories Banks, Currencies and VATRates will be for any value of the separator Organization(within one value of the separator CompanyGroup).
- The attribute is set up in the value True - conditional separation is **turned on** and **its own** set of values for directories Banks, Currencies and VATRates will be for any value of the separator Organization (within one value of the separator CompanyGroup).

In fact, while you are changing the value of the property CommonRLD, the separator is turned on or off for the separator Organization.
Let's the following entries are in the directory CompanyGroups:

Name	**CommonRLD**	**Link**
Progress	False	GK1
Trading house	TRUE	GK2

The following entries (two for each value of the separator CompanyGroup) are in the directory Organizations:

Name	**CompanyGroup**	**Link**
Component base	GK1	OR1
World of air conditioners	GK1	OR2
StroyTorgVse	GK2	OR3
Light	GK2	OR4

The following information will be in the directory Banks:

Name	**CompanyGroup**	**Organization**
AlfaBank	GK1	Blank link
SberBank	GK1	OR1
VTB	GK1	OR2
Avangard	GK2	OR1
Uralsib	GK2	OR2

Now let's look at the example of what data will be available in what case.

Example 1:

The attribute CommonRLD for the item Progress is set up in the value False.

It means that separation for the directory Banks is **turned off**. So components of this directory don't depend on the value of the separator Organization and in this case the value equal to the default value for the type of the separator will be recorded in the item of the directory Banks (for the separator Organization). In our case, such value is the blank link (because the type of the separator Organization is -the link to the directory).

That's why if the separator CompanyGroup is set up in the value GK2 the list of the directory Banks will display only one entry: AlfaBank.

Example 2:

The attribute CommonRLD for the item Progress is set up in the value True.

It means that separation for the directory Banks is **turned on**. It means that components of this directory depend on both the value of the separator CompanyGroup and the value of the separator Organization.

That's why the list of banks for the company group Progress and for the organization World of air conditioners will include only the bank VTB.

When you change the value of the attribute CommonRLD (and values of separators) during operation of the system, you may acquire these or those components of the list for the directory Banks (according to rules of conditional separation).

25.2.6.3. Conditional separation set up for the object (in the property "Components")

Conditional separation may be set up for each configuration object included into the separator. In this case separation will be turned off not for all objects included into the separator but for the objects for which conditional separation is set up.

Conditional separation may be managed by both the constant and the attribute of the configuration object.

Let's look at the example of such separation.

For example, several subscribers may do accounting in the same infobase, and you may do accounting within each subscriber on behalf of several partners. At this, the situation may occur when all partners operate the same list of warehouses and the situation may occur when you may generate its own list of warehouses for each organization.

Implementation of such design will require the following configuration objects:

- the directory `Subscribers` with the attribute `IndividualWarehousesList` of the type `Boolean`;
- the directory `Organizations`;
- the directory `Warehouses`;
- the separator `Subscribers` of the type `CatalogRef.Subscribers`;
- the separator `Organization` of the type `CatalogRef.Organizations`.

We need to do the following to arrange the considered design:

- we need to exclude the directory `Organizations` from components of the separator `Organization`;
- for the directory `Warehouses` (included into the separator `Organization`) we need to set up conditional separation via the attribute `IndividualWarehousesList` of the directory `Subscribers`.

Let's the directory `Subscribers` has two entries:

Name	IndividualWarehousesList
Subscriber 3453	`TRUE`
Subscriber 4617	`False`

If the user logs in on behalf of the subscriber No. 3453, it will be possible to maintain the individual list of warehouses for each organization, because the attribute `IndividualWarehousesList` is set up in the value `True` (separation of the directory `Warehouses` in the separator `Organization` is turned on).

At the same time when you log in to the system on behalf of subscriber No. 4617, the list of warehouses will be the same for all organizations, attribute `WarehousesSeparateList` is set to `False` (disabled division of the `Warehouses` directory in the `Organization` separator).

25.2.6.4. Features of conditional separation in a distributed system

Usage of conditional separation in the distributed system has some features. These features can be divided into "logical" and "technical".

The "technical" features include the inability to arbitrarily change the state of the object controlling the conditional separation in a separated session, if this object is part of a separated exchange plan. This is due to the peculiarities of

automatic registration of object changes by separated exchange plans. At automatically registration of changes it is possible to change the object that controls the conditional separation either in a unshared session or when the current state of the object is `False`. That is, it is only possible to enable conditional separation, and there is no possibility to turn it off (in automatic mode).
The "logical" feature is that the change in the state of conditional separation in the main node of the distributed infobase has a significant impact on the state and behavior of the entire distributed information system. This change can affect one data region if the conditional separation is controlled by the attribute of the object that serves as the type of the other separator. Also, changing the state of conditional separation can affect the entire infobase if the conditional separation is used to switch between the two modes of operation of the application.
Thus, in the first case (example with a regulatory and legislation data) when changing the conditional separation in the peripheral database users, "disappears" the regulatory and legislation data. Therefore, only the data domain administrator or the person with sufficient authority to accept such a change should have access to the change to the attribute that controls the conditional separation.
The second example (changing the mode of the application) will have an impact on the whole infobase. In peripheral databases, users can "lose" all their data (due to changing the separator value), etc. It is obvious that only the administrator of the entire infobase (or the person authorized for such actions) can perform such a critical change, while other users may not even be able to read this attribute (or constants).

25.3. The definition of the ability to perform the action or set of data

In the case when the system uses multiple separators to determine accessibility of a certain action with a particular object, you should use the following rule:

- For each separator that separates required object, the possibility of performing the necessary action is determined as if the separator in the system is one (this, of course, takes into account the mode of separation -independent or independent and joint).
- The result for all separators is sum up "by I".

The following sections will discuss how to perform an action on an object for a single separator (unless otherwise explicitly described). Then is needed to apply this rule to get the final action if the object is part of multiple separators. For example, in the system there are two independent separator: `Subscriber` and `Organization`. There are two directories: `Item` and `Contractors`. Each directory is separated by both separators.

Example 1:

It is necessary to determine whether the item list can be retrieved if the `Subscriber` separator is used and the `Organization` separator is -not. As both separators are an independent, to obtain the list of directory it is necessary that the separator was used. Therefore:

- For the `Subscriber` separator, it is possible to get the list of items because the separator is used. The result is equal to `True`.
- For the `Organization` separator, the item list cannot be obtained because the separator is not used. The result is equal to `False`.
- Thus, in a session with such separators settings, the directory list view will be (`True` **AND** `False` = `False`) **impossible**.

Example 2:

It is necessary to determine a possibility to obtain a list of contractors if both separators are used.

As both separators are an independent, to obtain the list of directory it is necessary that the separator was used. Therefore:

- For the `Subscriber` separator, the list of contractors can be obtained, since the separator is used. The result is equal to `True`.
- For the `Organization` separator, the list of contractors can be obtained because the separator is used. The result is equal to `True`.
- Thus, in the session with such separators settings, the directory list view will be (`True` **AND** `True` = `True`) **possible**.

Example 3:

It is necessary to determine what values for `User` column, will return the `GetEventLogFilterValues()` method. Two separators have been created in the system, the first separator is used in the current session and the second one is not used. Therefore:

- for the first separator, the method returns only those users who are logged in to the event log for the separator values equal to the value in the current session;
- for the second separator, the method returns all users;

- thus, in a session with the specified separators state, the method will return only those users that will be the intersection of the lists for the first and the second separators (operation **AND**).

25.4. Setting the separator values at startup

25.4.1. General information

When start up the "1C:Enterprise", it may be necessary to specify a separator value directly in the startup command line (for example, to perform authentication, if authentication separation is selected).
If one or more separators values are not specified on the command line, the missing separators are taken from the separators specified for the current user. If the same separator is specified for both the user and the command line and their values do not match, the use and value of the separator will be set from the command line if the user of the infobase has the right to set session parameters that store the use and value of the separator. Otherwise, the user will not be authenticated.
When debugging a client running from the configurator, it is possible to specify separators in the startup settings dialog in the configurator. When specifying user on whose behalf it is necessary to perform start-up, selection of users are available. The list is limited to the specified separators values that separate authentication.
In the process of user authentication information database, the values of session parameters that are referenced properties Using separator and Separator value of the metadata object Overall attribute are set to the values specified for the user. By the time any code is executed in the 1C:Enterprise language (including the session module), the values of these session parameters will be set, and the event handler `SessionParametersSetting` will not be called when they are accessed.
It should be remembered that the definition of the separator value by its passed ID is performed without taking into account the use of other separators. That is, the definition of the separator value does not depend on the order of setting the values (and flag of usage) of other separators. If the system cannot uniquely determine the separator value by its ID (for example, there are two dictionary elements with the same code), an exception will be thrown.
If the usage flag and the separator value (see below) that are specified in the start line are different from the values that are specified for the user, then:

- if the user has rights to change session parameters that store the value and the flag of separator usage, these values are set from the start command line;
- if no such rights exist, authentication fails.

25.4.2. Client application command line

The command-line parameter is used to specify the values of separators Z.

Parameter syntax:

```
/Z "[<Flag>][<Value>][,[<Flag>][<Value>]]"
```

Description:

Each <Flag><Value> pair is responsible for setting one separator. Skip of separators in not possible. The order of collocation of the <Flag><Value> pairs corresponds to the order of separators in the configuration window. If the order of the shared attributes changes (for example, the list of shared attributes is sorted), it is necessary to change the location of the values in Z parameter. The <Flag><Value> pairs are separated by commas (",").

Parameters:

<Flag>

The flag of usage of this separator. Can take «+» (by default) and «-» values. If the separator value is preceded by a «+» flag (or nothing is specified), the specified value will be written to the session parameter that stores the value of the shared attribute, and the session parameter that stores the use of the shared attribute will be set to the True value.

If the separator value is preceded by a «-» flag, it means that the separator is not used in this session (the session parameter that is responsible for using the separator will be set to False). But if the value is specified, it will be written to the session parameter that stores the shared attribute value.

<Value>

Specifies the value to be set in the session parameter that stores the value of the appropriate separator. The separator value is set in text form and has the following format (depending on the separator type):

- For Boolean type, the values of 0 (False) or 1 (True) are specified.
- For the Digit type, a digit in canonical form is specified.

- For the `Date` type, a date is specified in yyyymmddhhiiss format, where:
  - yyyy - year (4 digits);
  - mm - a number of month (2 digits);
  - dd - a number of day (2 digits);
  - hh -a number of hour (2 digits);
  - ii - a number of minute (2 digits);
  - ss - seconds (2 digits).

  When setting the parameter, it is necessary to specify all the components of the date, i.e. the parameter value must consist of 14 digits.
- A text itself is specified for the `Line` type.
- For reference types, the text representation of the standard attribute `Code` or `Number` is specified. Codes and numbers must be unique among all values of the corresponding object of configuration. It is not allowed to use directories with code length equal to 0, and documents with numerators. If the link is a type of the shared attribute which is the separator, the code or the number of the predefined element is not allowed, if the predefined elements in this infobase have not yet been created for the object forming the separator type.

If the «+» or «-» character must be located at the beginning of the parameter value, it doubles. If the text line describing the separator value must contain the character «,», this character is doubled.

If the value specified in the command-line parameter can correspond to different types of values at the same time, (for example, value «1» can be a `True` value, «1» number, «1» directory code and the «1» document number), the conversion rule is determined based on the type of the appropriate separator.

If only a («+» or «-») flag is specified on the command line, but no value is specified, the default value of the appropriate type is used as the separator value.

Example:

```
/Z " 0,+001,+,---3"
```

In this example:

- the first separator is not used, but its value is specified: «0»;
- the second separator is used and its value is equal to «001»;
- the third separator is used and its value will be the default value for the type specified as the shared attribute type;
- the fourth separator is not used, but the «-3» will be its value.

25.4.3. Web client or thin client connected through a web server

There are two ways to specify separators values when running a web client (or a thin client connected through a web server) : in the address (command) line of the run with Z parameter and in the default.vrd file.

25.4.3.1. Using Z parameter

Separators values can be specified in the address line using the following Z parameter. The format of the parameter for the web client is similar to the command line for running the client application.

Example:

```
http://localhost/infobase?Z=" FirstSeparator,+"
```

Symbols, impermissible in URL (RFC 1738, http://tools.ietf.org/html/rfc1738.html), are converted to UTF-8 and encoded in accordance with section 2.2. URL Character Encoding Issues of RFC 1738 standard (using "%" and two hex symbols).

25.4.3.2. In the file default.vrd

It is possible to configure access to the infobase in such a way that the specific values of the separators will be set by the address of the infobase, and not by using command-line parameter Z or the connection string to the infobase. To do this, configure the <zones> element of the default.vrd file accordingly. It is also possible to prevent the values of objects associated with the data separation mechanism from being changed if the infobase is accessed using a web client or a thin client connected via a web server (secure session). Such an opportunity should be used if you need a guarantee that access to an infobase over the Internet will not be able to access another data area.

25.4.4. External connection and connection string

In the connection string and when the external connection is started, the following Zn parameter is used. The format of the parameter is similar to the command-line for running the client application.

Example:

```
Srvr=localhost;Ref=Demo82Srv;Zn=" FirstSeparator,+,---3"
```

Also it is possible to specify this parameter in the default.vrd file (in the connection string with infobase) when publishing to a web server.
There is another way to set the separators value in the default.vrd file.

25.5. Safe mode of data separation

When using various routine operations performed within sessions with unused separators values, the program code of the application can be executed, «reliability» of which, in general, can not be checked. "Reliability" means job strictly within the borders set, without abuse. For example, the called code should only work within the separators values that the caller has set for it, without attempting to navigate to other data areas. Such an attempt may lead to unauthorized access to the data of the "foreign" data area.
To prevent attempts to change the values of separators (and, as a consequence, the data area in which the program code works), it is possible to establish a safe mode of data separation. To do this, `SetDataSeparationSafeMode()` method is used. `safe="true"` для элемента `zone` файла default.vrd. If secure data separation mode is enabled by using the default.vrd file, then it is impossible to disable it using the method `SetDataSeparationSafeMode()` .
The following is prohibited in secure mode of data separation:

- Disabling the use of separator, unless the separation is conditionally disabled.
- Changing the used separator value, unless the separation is conditionally disabled.
- Modifying the objects managing the conditional separation:
  - Specified for the separator itself
  - Specified for the objects included in the separator.

Secure data separation mode can be set for each separator separately. The number of calls of the safe mode of data separation should match the number of shutdowns. However, if a secure data separation mode (once or more) is enabled within a procedure or function, but it is not disabled, the system will automatically shut down as many times as there are incomplete calls in the procedure or function being left.

If in a procedure or function, the safe data separation mode for the selected separator is turned off more times than it was turned on, an exception will be thrown.
The application of this mechanism may look as follows (there is an independent separator `Subscriber` in the system):

```
For Each SeparatorValue From SeparatorsValuesList Do

  // "go to" necessary area
  SetSeparatorValue(SeparatorValue);

  // set the secure mode of data separation
  SetDataSeparationSafeMode("Subcriber", True);

  // call the necessary programm code
  Process the data of area();

  // turn off secure mode of data separation
  SetDataSeparationSafeMode("Subscriber", False);

EndDo
```

Check the status of the secure mode of data separation by using `DataSeparationSafeMode()` method.

25.6. Deleting a data area

During the operation of the system, a situation may arise when the data area becomes unclaimed. For example, a subscriber ceases to use the services of the service (each subscriber works in one data area), or there is a reorganization of the group of companies (each company in the group works in its own area).
It is also possible that separators are removed from the system. But if the separators include the configuration objects with the predefined data, then it will be impossible to remove the separator, until predefined data will be deleted.
Deletion of data area is implemented with a method of global context `EraseInfoBaseData()` or with a help of the parameter of the command-line `/EraseData` of the batch mode of the Configurator running. For the method of 1C:Enterprise language, deleted region is determined by the values of session parameters that define the values of the separators in the session that the call comes from a method:

- If any separator is not used in the current session, then **all infobase data** will be deleted. After deletion, the state of the infobase becomes the

same as after creating a new, blank infobase and loading the configuration into it.

- If at least one separator is used in the current session, the **fully separated data** of the current data area is deleted. If two or more separators are used simultaneously (i.e. they store their values and conditional separation data in similar configuration objects), `DeleteInfobaseData()` will consider a configuration object as being part of a current area, if it is separated by at least one of the said separators using similar configuration objects to store their parameters. Deleting the data area causes the area to be in the same state as after the first setting of current set of values for the used separators before executing operations with data.

TIP. Immediately after executing the method, it is recommended to change the values of separators that define the data area.

To call the method, the user must have the `Administration right` and the ability to obtain exclusive access to the infobase. Exclusive access must be set by explicitly calling the global context method `SetExclusiveMode()`.

When deleting a data area by using the batch run mode of the Configurator, the deleted region is determined by the parameter `/Z` of the running command-line. If not all separators is specified, the area is determined similarly to the call of the `EraseInfoBaseData()` method (see above). The system tries to automatically obtain exclusive access to the infobase. In case of an error in obtaining exclusive access, the execution of the batch mode of running the Configurator fails.

The following data will be deleted during the deletion:

- From tables associated with configuration objects that are part of separators (including predefined data);
- Storage settings
- User work history;
- A list of infobase users;
- Configuration extension;
- The time zone of the infobase - if no independent separators are defined in the configuration, the value of this parameter becomes undefined, otherwise - the value is deleted.
- If no separator is used in the session, the following infobase settings are filled with default values:
  - Data lock timeout;
  - Password minimal length;
  - Password complexity validation;

    - Enable full-text search.
- When compatibility mode is set to version 8.3.2 or later, the predefined data is returned to the values defined in the configurator or to the default values. When compatibility mode is disabled, predefined data will be created at the first time of access to them;
- The value of element attributes `ThisNode` of exchange plan will be set to the default values;
- The values of unshared constants receive their default values, and the values of separated constants are removed.
- Registration data of infobase in the interaction service.
- For data history, the behavior of the method depends on the session in which the method is called:
    - Unshared infobase or any separators is not used in the session:
        - All existing versions in the entire infobase are deleted.
        - All settings of the history data are deleted. The versioning settings are aligned with the settings in the metadata.
    - The session uses both independent and shared separators:
        - The data history for completely separated data is deleted.
        - Versioning settings and metadata history are not deleted.
    - In the session, only independent separators are used:
        - The data history for completely separated data is deleted.
        - Settings for history data usage for the current combination of separators are deleted.
        - The metadata history for the current combination of separators is deleted.

25.7. Features of system objects behavior

Chapter 26. Data history

26.1. General information

Data history - it is a special mechanism which allows to store the data of objects of configuration in the database, organized by a timeline. As an equal term, the term of **data versioning** also will be used. The base stores a **version** - data, which existed in the object at the moment of change as well as metadata state at the moment of change. Механизм позволяет настраивать хранение истории данных для объекта целиком и более тонко управлять работой механизма для каждого реквизита (включая табличные части). The arbitrary data, which content is determined by the developer, can be added to each version when versioning the data. The mechanism also allows the actions which shall take place after the data version is actually saved to a database table (version post-processing). Post-processing can be used when a version saving fact starts one or several actions. The fact that the action shall be performed after the successful saving of both the object and its next version is the principal difference.
There is a possibility to control the enabling and disabling of history switching on and off both from the Configurator and with the help of the built-in language. Thus, it becomes possible to include change history storage only for the data that is needed by specific users, without the need to change the configuration.
In addition to storing the actual changes, the data history mechanism allows to perform the following operations:

- write the data version for the supported object;
- write additional data absent from the actual data of versioned object;
- get the data of a certain version (including the additional data if it is set);
- delete the data of a particular version;
- obtain the difference between two elements of the data history;
- and other useful features.

Data history is supported for the following objects:

- general attrubutes;
- constants;
- exchange plans;
- catalogues;
- documents;

- charts of characteristic types;
- chart of accounts;
- chart of calculation types
- business processes;
- tasks;
- information register.

Dealing with data history is regulated by access rights and logged in the event log.
Data history is not transferred through mechanisms of distributed infobase.

26.2. Mechanism structure

Data history is stored in special tables of the same infobase for which versioning is set. In addition to the data itself, the data history also stores metadata of previous versions of versioned objects. Version of metadata of the object is created at the time of change of the object's metadata. The moment of creation of the metadata version has nothing to do with changing the data of the versioned object. The data history does not store the types of attributes, but only the actual values of these attributes. It should also be remembered that from the point of view of the system, the following pairs of actions are the same:

- enable versioning for attribute and create new attribute;
- disable versioning of attribute and delete the attribute.

In addition to the versioned object changed data storage, the versioning mechanism allows specifying a special comment and additional data, which can be used in the subsequent views and analysis of changed data, for each version
The process of forming a version consists of several steps:

1. The system records the fact of changing an object. This occurs either automatically or by calling a method in the program interface. An information about changing an object is transfered to a special queue (queue of versions). The arbitrary data required in the subsequent work can be added to the versioned data at this point (`WriteDataHistory.AddAdditionalData()` versioned object method).
2. The data is transferred from the queue of versions to the database tables actually storing the version data. This operation is performed with the help of `DataHistory.UpdateHistory()` method. There is a possibility to update the data history asynchronously, for example, by using a regular task (similar to updating the full-text search index). In other words, the data version becomes "real" only after it is moved from the queue of versions to the tables of history storing. In this case, several consecutive changes to the data object (no matter how these changes are made) are recorded separately

and transferred separately. So, all changes to data objects are recorded in the data history, regardless of the frequency of history updates and the number of consecutive changes to one object. Each processed data version will be transferred to a special queue of versions requiring processing after the version saving (post-processing queue). The necessity of such transfer is defined by the versioned object property `WriteDataHistory.ProcessAfterWriting`.

3. The post-processing queue processing begins after the versions processing queue processing completion. This processing can be performed by both the method `DataHistory.UpdateData()` and the special method `DataHistory.ProcessAfterVersionSaving()`. The processing itself shall be performed in the handlers `AfterWriteDataHistoryVersionsProcessing()` of the versioned object manager's modules.
4. After the version post-processing completion, this version must be deleted from the post-processing queue with the help of `DataHistory.DeleteFromProcessingAfterWriteVersion()`. If it is not done this version will be constantly transferred to the event handlers `AfterWriteDataHistoryVersionsProcessing()`.
5. The post-processing queue can be cleared by `DataHistory.DeleteFromProcessingAfterWriteVersion()` method.

The first two steps of the description above are compulsory and always performed in the process of data version formation. The subsequent steps are used only when the developer wants to use the post-processing queue.

Also remember that the data history formation can be conventionally divided into two methods: the usual method described in this section and an accelerated method. In the second case, it is possible to initiate the whole process of a data version writing but only for one version of one data object. This processing method will be described in more details separately.

Versioning can be set in two ways:

1. in configurator;
2. using 1C:Enterprise language

With a help of configurator, it makes sense to set data history in case when developer of application is sure that the data history for any object will be required in any options of the system operation or some application logic is tied to this history.

If the data history is used to monitor the work with the infobase and is used to answer the question "why so?", it is logical to implement enabling of data history using the built-in language so that the user of the application can choose the ob-

jects and attributes for which versioning is required. It is also should be remembered that enabling data history from the 1C:Enterprise language does not require removing the configuration from support.
To enable the use of data history for an object in the configurator, it is necessary to set the property Data history for this object in the value Usage. After that, it is necessary to create a regular task that will update the history of the data in turn of changes. A similar property is presented in subordinate configuration objects: attributes (standard and usual), tabular parts and their attributes, resources of information registers.
Data history property is set in the following values:

- catalogues, documents, chart of characteristic types, charts of accounts, business process, tasks, information register - value Do not use;
- standard attributes - value Use;
- objects attributes - value Use;
- attributes of tabular parts - value Use;
- measurements of information registers - Use value is always used (with out possibility of disabling);
- information register resources- value Use.

Thus, for the type-forming configuration objects, a logging should be enabled forcibly, and all the attributes of such objects automatically fall into the history and it is necessary to forcibly exclude these attributes from the history.
Program work with the data history will be discussed in the next section.

26.3. Mechanism usage

26.3.1. General information

The global context property of the `DataHistory` is used to refer to the data history. The data history manager is available through this property. Therefore, the reference to the methods of the program interface will look as follows: `DataHistory.Name of CalledMethod()`. For simplicity, the name of the global context property will be omitted in the following text.

26.3.2. Managing the use of data history

The state of the object from the point of view of the mechanism of data history can be divided into two parts:

1. data history settings made in the configurator;

2. data history settings made in the mode of "1C:Enterprise".

For access to each part of these settings it is necessary to use different ways. To get the settings made in the configurator, metadata access should be used (object `Metadata`). To get the necessary information, the `DataHistory` property of metadata objects should be analyzed. Thus, to get the flag for enable the data history for the `Goods` catalog, the following expression can be used:

```
ObjectEnabled = Metadata.Catalogs.Goods.DataHistory;
```

Similar constructions should be used to get information about the attributes of the objects.

At the same time, to get information on what changes have been made in the "1C:Enterprise" mode, it is necessary to use the method `GetSettings()`. In this case, as a parameter, it is necessary to specify the configuration object whose settings must be gotten. The method returns an object of type`DataHistorySettings`. If the method has returned the `Undefined` value, i means that no changes, regarding settings made in the configurator, were made with the data history.

If known that the configuration object is involved in versioning, the `GetMetadata()` method can be used to get information about all attributes, values of which are stored in the data history. The result includes both attributes, versioning of which is set in the configurator, and attributes, versioning of which is set in the "1C:Enterprise" mode.

Setting the versioning system settings is performed using the `SetSettings()`. The versioning settings are determined by the object `DataHistorySettings`, that is passed as a method parameter. In this object, it is necessary to specify the history usage for both object as a whole (property `DataHistorySettings.Usage`), and object attributes (property `DataHistorySettings.FieldsUse`). The `DataHistorySettings.FieldsUse` property is a match in which the key is the name of the attribute, and the value is the - flag of the use of the attribute in the data history. If it is necessary to specify the attribute of the tabular part, it is specified through a point after the name of the tabular part: `TabularSection.Attribute`.

The following features should be considered when set the settings: for attributes listed in the `DataHistorySettings` object, the flag of participation of the attribute in the data history is set. For other attributes of the object, participation in the versioning will be adjusted to the settings specified in the configurator.

Also, remember that calling the `SetSettings()` method, where the second parameter is set to `Undefined` value, will cause the versioning system settings to be set to match the settings in the configuration.

26.3.3. Write a version

26.3.3.1. Usual processing

There are two ways to write data changes:

1. Automatically, by "1C:Enterprise" system itself. The main method to use the data history mechanism.
2. With a help of 1C:Enterprise language, using the `WriteVersion()` method. This method is used when you want to transfer the data history to the system store from some other store. When using this method, the writing is carried out directly to tables that store data history without an intermediate queue.

In the first case, writing is performed in three stages:

1. At the first stage, the system commits the need to generate a version for any object for which versioning is enabled. This commit is performed automatically whenever a versioned object is written. At this stage, the handler can additionally perform several actions:
   - Specify the additional data to be saved together with the data of the versioned object. It is done with the help of the method `WriteDataHistory.AddAdditionalData()` to add the additional versioned data and the method `WriteDataHistory.AddAdditionalDataFieldRepresentation()` to set the additional data representation (for the display in the user interface).
   - Specify the necessity of the written version participation in the post-processing queue. The versioned object property `WriteDataHistory.ProccessAfterWriteData` is used for it.
   - Specify for the platform that this version shall be processed in the accelerated mode. The versioned object property `WriteDataHistory.UpdateHistoryImmediatelyAfterWrite` is used for it.
2. To generate the actual version in the history, the developer must explicitly call the `UpdateHistory()` method. Only call of this method will lead to the actual change in the tables of the infobase responsible for storing the history, and the generated version can be obtained for analysis. It is recommended to place the history update in the scheduled job, which will be triggered with a certain periodicity. The frequency of the scheduled job is determined by the number of versioned objects, the intensity of their changes and how often it is necessary to obtain access to the historical data.

The platform forms the post-processing queue after the modification of the tables responsible for the history storage.

3. The data is processed upon recording data history version. This processing can be performed by both `UpdateHistory()` method and `ProcessAfterVersionSaving()` method of the data history manager.

The multi-stage method of history formation is implemented with the purpose to minimize the load on the database during the versions list formation with a large amount of changed data and also to divide the performance of different mechanisms associated with the data versions formation.

26.3.3.2. Accelerated processing

In some cases, it is required to form the data history (and to perform the actions associated with it) at once, without waiting for the regular task on the data history update to be performed. It can be done using the following methods:

- To check the box Update data history upon recording. In this case, the accelerated processing will be applied to any object of this type and any object writing. It is not recommended to check this box for the types of meta-data which presuppose a lot of objects and their intensive change.
- To use the property `WriteDataHistory.UpdateHistoryImmediatelyAfterrite` upon an object writing. In this case, the accelerated processing will be applied only to exclusive cases, when some conditions, which can be checked upon the data object writing, are met.
- To use the embedded language method `UpdateHistory()` with the transfer as a parameter of the required object.

In any of the cases specified above, the whole cycle of history processing will be performed but instead of processing the whole queue of versions one change of the information base object will be processed.

26.3.3.3. Dada adding to a version

The platform allows adding additional data to the own data of the saved object during data versions writing. To do it, use `WriteDataHistory.AddAdditionalData()` method for the versioned object. Three values are used as the parameter for this method:

- `Name` -additional data "attribute" name.
- `Value` -additional data "attribute" value. If `ValuesTable` is transferred as `Value` parameter it means that a tabular part is added to the additional date. In this case, the values table structure determines the tabular part structure.

- `Presentation` -contains the presentation of the additional data "attribute". Either a uniform presentation for all languages or an "attribute" local presentation can be set. In the first case (one presentation in all languages), the presentation is set by a usual line. In the second case (local presentation), the presentation is set by a correlation, where the `Key` determines the language code and the `Value` -determines the presentation itself.

The presentation for the additional data can also be specified separately from the data adding. Then, you should use the method `WriteDataHistory.AddAdditionalDataPresentation()`.

26.3.4. Processing after history writing

The platform allows performing the actions which shall take place only after the object is docked in the data history. A situation when it is necessary to change the sale price if the purchase price is changed can be viewed as an example.

The general scheme of work has been described earlier. This section will describe the technical peculiarities of processing after the data version is written. A version can be added to the post-processing queue in the following way:

- By specifying the property Process after data history version is written in the object configuration properties. In this case, the writing of the objects of this type will result in each version of this data automatic getting in the post-processing queue.
- To use `WriteDataHistory.ExecuteAfterWriteVersions` property when writing an object. In this case, the data version gets in the post-processing queue only if some conditions, which can be checked when writing an object data, are met.

As has been said above, own data history is formed by `UpdateHistory()` method of the data history manager. If `ExecuteAfterWriteVersions` parameter of this method is set in the value `True` then, after the platform completes the data history formation, the data versions for which `WriteDataHistory.ExecuteAfterWriteVersions` property is set in the value `True` will be placed in the post-processing queue. Then the platform initiates this queue processing by calling `ExecuteAfterWriteVersions()` method.

The following happens during this method execution:

- A collection of versions of an object of one type is formed (not more than 1,000 versions).
- `AfterWriteDataHistoryVersionsProcessing` event handler is called in the manager module of the relative configuration object.

Each processed object needs to be clearly deleted from the post-processing queue in `AfterWriteDataHistoryVersionsProcessing` event handler. It can be done with the help of `DeleteFromProcessingAfterWriteVersion()` method. Otherwise, the data version will remain in the post-processing queue.
The post-processing queue can be cleared with the help of `DeleteFromProcessingAfterWriteVersion()` method. Do not forget that the deletion from the post-processing queue takes place over time. The data versions created more than 20 days ago (from the clearance moment) will be deleted from the post-processing queue in "default" mode. `Date` parameter of `DeleteFromProcessingAfterWriteVersion()` method allows changing the clearance interval.

26.3.5. Getting a version list

During operation, the system can generate a sufficiently large number of objects versions. However, during the analysis of the history of all the created version may not be required. At the same time, it will be necessary to select object versions created by specific user at specific date interval, and so on. This filter can be done using `SelectVersions()` method.
The `SelectVersions()` method parameters specify criteria of version filter, which data are needed to be returned, how to organize the returned data, and the maximum number of records that can be obtained by querying the data history.
Filter criteria (parameter `Function` filter `SelectVersions()`) describes conditions that must match the versions that will be returned as a result of the function. Full list of filter elements is given in the Syntax Assistant. A version is considered to be eligible if it meets all the filter criteria specified in the parameter (conditions are combined "by AND"). If the parameter allows to specify several conditions (by combining them into an array), the version will match the filter if the value in the version matches to at least one value from the filter parameter (several conditions for one value are combined "by OR"). An exception to this is the filter by parameters `FieldValues` and `FieldsValuesChanging`, functioning of which will be described in detail below. If a string is passed as a filter value, then, as a rule, comparison with this string is usually not performed on full equality, but on occurrence, the - version is considered to satisfy the condition if the value in the version includes the value from the filter.
When filling in the filter criteria, it is useful to remember about following features:

- To filter, it is necessary to specify either the `Metadata` parameter, or the `Data` parameter. If both parameters are specified, or both of these parameters are empty, an exception will be thrown- when trying to retrieve the

version list. The exception will not be generated if the metadata of the `Data` parameter matches the value of the `Metadata` parameter.

If the data history is obtained for a specific data object, it is recommended to update the history by the required object using the `UpdateHistory(ElementLink)` method before getting the list of versions. Here the `ElementLink` - is that data element for which the version list must be acquired. In this case, the version list for one object will be forcibly updated, and the `VersionsSelecton()` method will operate the full list of versions.

- For filter by user, the user ID of infobase, instead of `InfobaseUser` object is used . The user ID can be obtained by using the `InfobaseUser.UUID()` method. In addition to selecting by user ID, it is also possible to use filters by name and full name of the user who written the version. Such filter, for example, can be used if the user who write the version has already been deleted, and it is not possible to obtain his UUID.
- `FieldsValuesChanging` allows to filter only those versions, where changing of values of specified in the parameter fields had been made. This filter parameter does not allow to specify specific values, but allows to determine the fact of the change itself.
- If there is a need to filter versions where specific conditions must be imposed on the field values, it is desirable to use the `FieldValues` parameter. This parameter contains an array of structures of a special kind. The operation of this parameter is described in more detail below.

`FieldValues` parameter contains an array, which contains elements of `Structure` type of the following components:

- `Field` - key contains the name of the checked attribute of the object in the quality of value. If it is necessary to specify the attribute of the tabular section, in this case the name is specified through a dot on behalf of the tabular section, for example, `Prices.Price`.
- `ValueBeforeModification` - key, as a value, contains the value of the attribute, whose name is specified in the structure element with the `Field` key, which was before the version was created.
- `ValueAfterModification` - key, as a value, contains the value of the attribute, whose name is specified in the structure element with the `Field` key, which was in the new version of the object.
- The structure must always contain the element with the `Field` key and at least one element that describes the value for that element.
- If in the array which is passed as the `FieldsValuesModification` parameter, two structures with the same value of the key `Field` are passed,

then such elements will be combined "by OR". However, unlike conditions of filter will be combined "by AND".

- Filter allows to specify checks only on the condition of equality of two values. Other types of conditions are not supported.

Let's consider an example of operation with filter. Some document `Note` will be used as an example, which contains attributes `Contractor`, `Discount` and tabular section `Components` with attributes `Goods`, `Price` and `Quantity`. If it is necessary to retrieve all versions of the document in which the attribute `Discount` is changing, then the filter will look like the following:

```
Filter = New Structure;
Filter.Insert("Data", Link to Note);
FieldList = New Array;
FieldList.Insert("Discount");
Filter.Insert("FieldValuesModification", FieldList);
Versions = DataHistory.SelectVersions(Filter);
```

If it is necessary to get all versions of the document where the `Discount` attribute became, for example, 20%, the following filter must be used:

```
Filter = New Structure;
Filter.Insert("Data", Link to Note);
DiscountModification = New Structure;
DiscountModification.Insert("Field", "Discount")
DiscountModification.Insert("ValueAfterModification", 20);
FieldsModification = New Array;
FieldsModification.Insert(DiscountModification);
Filter.Insert("FieldValues", FieldsModification);
Versions = DataHistory.SelectVersions(Filter);
```

In the above example it should be understood that the result of this query will return the list of all versions of the document, where the value of the `Discount` field when recording of the object was equal to 20. The resulting list is not a list of versions in which the value of the attribute has been changed! In order to get a list of versions where the discount value has become equal to 20 (but it doesn't matter, what this discount was before the modification), it is necessary to generate the following filter:

```
Filter = New Structure;
Filter.Insert("Data", Link to Note);
DiscountModification = New Structure;
DiscountModification.Insert("Field", "Discount")
DiscountModification.Insert("ValueAfterModification", 20);
FieldsModification = New Array;
FieldsModification.Insert(DiscountModification);
Filter.Insert("FieldValues", FieldsModification);
FieldList = New Array;
FieldList.Insert("Discount");
Filter.Insert("FieldValuesModification", FieldList);
Versions = DataHistory.SelectVersions(Filter);
```

In case where necessary to get only those versions in which attribute `Discount` was changed from 10% to 20%, the following filter must be used:

```
Filter = New Structure;
Filter.Insert("Data", Link to Note);
DiscountModification = New Structure;
DiscountModification.Insert("Field", "Discount")
DiscountModification.Insert("ValueBeforeModification", 10);
DiscountModification.Insert("ValueAfterModification", 20);
FieldsModification = New Array;
FieldsModification.Insert(DiscountModification);
Filter.Insert("FieldValues", FieldsModification);
Versions = DataHistory.SelectVersions(Filter);
```

Also the system allows to implement more complex queries to the history. For example, it is necessary to get those versions of the document `Note`, where price of specific item will become equal to 15 or 35 money unit:

```
Filter = New Structure;
Filter.Insert("Data", Link to Note);
Condition1 = New Structure;
Condition1.Insert("Field", "Components.Price")
Condition1.Insert("ValueAfterModification", 15);
Condition2 = New Structure;
Condition2.Insert("Field", "Components.Price")
Condition2.Insert("Value afterModification", 35);
Condition3 = New Structure;
Condition3.Insert("Field", "Components.Item")
Condition3.Insert("ValueAfterModification", LinkToItem);
FieldsModification = New Array;
FieldsModifications.Insert(Condition1);
FieldsModifications.Insert(Condition2);
FieldsModifications.Insert(Condition3);
Filter.Insert("FieldValues", FieldsModification);
FieldList = New Array;
FieldList.Insert("Components.Price");
Filter.Insert("FieldValuesModification", FieldList);
Versions = DataHistory.SelectVersions(Filter);
```

In the example listed above, the following condition is generated:

1. The history of the document which is determined be the variable value `LinkToNote` is considered.
2. Only those versions are selected from history in which:
   - there is a line in the tabular section, in which the item from variable `LinkToItem` is specified;
   - in the same line, the attribure value `Price` became equal to 15 **or** 35;
   - in the same line, the attribute `Price` was changed prior to writing of version.

26.3.6. Getting version data

The main purpose of data history - is a possibility to get data of specific version. The system of versioning the "1C:Enterprise" platform provides several different ways of getting this information.
`GenerateFromVersion()` method allows immediately to get the data object of the required version by its link or write key. To get data, it is necessary to have a reference (or write key) to the object whose version must be obtained, as well as the version number itself. As a result, the object of corresponding type, which can be written again to the data base will be generated.

Example:

```
// Version recovery
ReferenceToObject = GetReferenceToObject();
VersionNumber = DetermineRequiredVersionNumber();
Object = DataHistory.GenerateFromVersion(ReferenceToObject,
VersionNumber);
Object.Write();
```

If an attempt is made to receive the version data that corresponds to the deletion of the object-, the object filled with the data that was in the object at the time of deletion will be received.
The developer can interfere with the process of object formation based on the data version. For this, it is possible to use handler of the event `GenerateFromDataHistoryVersionProcessing()` of the object module and register set of information register. This handler is called after calling of `GenerateFromVersion()` method and at the time of interactive transition to version.
When creating a form, the developer also has an opportunity to understand tha `Number Of Version Of Transition To Data History Version` If the value of this parameter differs from `Undefined`, then a form is opened, in which the data corresponds to the version number specified in this parameter.
`GenerateFromVersion()` method for subordinated information register generates record set based on the write key. Generated record set contains all

record set strings which can be obtained from the data base by the specified key. Therefore, when write such record set, all record set and not only one record, which is recovered from the history will be updated. This means that if there is a change in the read data from another session between the version data being received and the data being written, the data entered from the other session will be lost. To prevent such loss, it is necessary to impose a lock on the changed data.
If it is not necessary to get the version data directly into the object of required type, it is possible to use `GetVersionData` method, which returns data of the required version as a quality of structure, where the attribute name is the key, and the attribute value - is the value

Example:

```
// Getting version data
ReferenceToObject = GetReferenceToObject();
VersionNumber = DetermineRequiredVersionNumber();
VersionData = DataHistory.GetVersionData(ReferenceToObject,
VersionNumber);
Report(VersionData.VersionedAttribute);
```

In addition to getting specific version data, the data history allows to get the list of versioned attributes (and their representations) for each version of the object. This feature allows to determine the differences between the current set of versioned attributes and the set of versioned attributes of the required version. The components of versioned attributes allows to get the `GetMetadata()` method. It is necessary to understand that the structure with the list of attributes will be received, and not the object of metadata. With this method, it will be impossible to compare the properties of versioned attributes. However, the result of this method will allow to understand what differences in the structure of versioned attributes exist between metadata and versioning settings that were at the time of creating a version with a certain number and the current metadata structure and versioning settings.

26.3.7. Version comments

Any version of the data can have a comment that allows to comment on the version in some way. To edit a comment, use the `WriteComment ()` method. The maximum comment size is 1024 characters.
The system also provides the ability to specify a comment directly during the recording of the data object. For it, use `WriteDataHistory.VersionComment` property when writing a versioned object. The comment will be formed for the version which will be created in the result of the current object writing.

26.3.8. Version deletion

If necessary, a deletion of one, more, or all versions in the infobase can be required. To do this, there is the `DeleteVersions()` method and the method of global context `EraseInfoBaseData`.

`DeleteVersions()` method provides the possibility to run the following actions:

1. Delete all historical data for any configuration object. To do this, call the method that specifies the configuration object as a parameter: `DeleteVersions(Metadata.Catalogs.Items)`.
2. Delete data for any configuration object before the specified date: `DeleteVersions(Metadata.Catalogs.Items, BegOfYear(CurrentDate()))`.
3. Delete the version set of any data object: `DeleteVersions(ReferenceToObject, StartOfRange, IntervalCompletion)`.
4. Delete all versions of any data object before the specified date: `DeleteVersions(ReferenceToObject, BegOfYear(CurrentDate()))`.

26.3.9. Identifying differences between versions

A standard task when working with data history is to get the difference between versions. `GetVersionDiff()` is a method for this. It should be understood that the term "difference" does not imply any calculation, including any analysis of changes on intermediate versions (if any). The difference between versions assumes that the system will return the versioned attribute for two specified versions, and the developer will interpret the result. When comparing versions is not possible attempt type casting. So, if the type of the attribute value has changed it means that the attribute value before and after will be different. To distinguish between versions, there is a need to specify required object and two versions: version **after** modification (version lying to the right on the timeline) and version **before** modification (version lying to the left on the timeline). If the version number is not specified before the change, a comparison will be made between the specified version and the previous version.

When forming the result, the system relies on the metadata structure, which was at the time of the version creation after the change (the second parameter of the method). This implies the following feature: if there was any versioned attribute in the version before the change, and there is no such attribute in the version after

the change, the difference between the versions will not contain any information on the deleted attribute. When considering the reverse situation (in the version before there is no attribute, and in the version after the attribute is appeared), the default value for the corresponding type will act as the value before the change.
The result of the method execution is a structure where the key is the name of the versioned attribute or tabular section, and the value is the structure (for attribute) or an array of structures (for the table part) with information about the values before and after the change.
The value describing the attributes represents structure with the following components of elements:

- Key `ValueAfterModification`, value - attribute value, which was in the older version.
- Key `ValueBeforeModification`, value - attribute value, which was in the earlier version. If in the earlier version the attribute were missing or versioning was disabled for it, the structure element with the key `ValueBeforeModification` will be absent.

If the versioned attribute has a reference type, the information about the reference is transmitted by a structure that contains two elements: with the `Presentation` - key, the presentation that the reference object had at the time the version was generated and `Reference` -the actual value of the reference.
The value describing the tabular section is an array where each element describes the changes in the tabular section row. Each row is a structure with the following elements:

- Key `RowNumberInVersionAfterModification`. The value describes the row number in the tabular section for the later version.
- Key `RowNumberInVersionBeforeModification`. The value describes the row number in the tabular section for the earlier version.
- Key `KindOfRowModification`. The value describes how the row of the tabular section is modified. Depending on this value, the structure of the components located in the element `Field` is changed.
- Key `Fields`. The value will be the structure that describes the changes made. The content of the structure depends on the type of row change:
  - Row creation. For each attribute of the tabular section participating in versioning, the individual element will be created. The key of the element will be the name of the attribute, and the value -, the value set in the attribute.
  - Row modification. For each attribute of the tabular section participating in versioning, the individual element will be created. The key element will be the name of the attribute, and the value - is the structure described above (change the attribute of the object).

- Deleting a row. For each attribute of the tabular section participating in versioning, the individual element will be created. The element key will be the name of the attribute and the value - is the value that the attribute had before the delete operation.
- Drag of the row. This operation is not reflected in the field list in no way.

26.3.10. Data history and extension

The data history can be used for native and adopted objects of configuration extension, while:

- For native configuration object:
  - History is maintained as for the native configuration object;
  - History data will be deleted after the extension is completely removed from the infobase.
- For adopted object of extension:
  - Use in the data history can be configured only for attributes added in the extension. The actual history is determined by the settings of the borrowed object.
  - Deleting an extension does not clear the history (similar to the behavior of the data history without the extension if the versioned object attribute is deleted).

Chapter 27. Database copies mechanism

27.1. General information

NOTE. Available only for CORP licenses.

Database copies mechanism allows setting the system so that a copy for the required data will be created and updated in other physical databases. Several database copies may exist at the same time. The system allows setting what configuration data will be duplicated in each database copy. You can save in a copy either a single configuration object or all configuration objects supported by this mechanism which exist as of the moment you start to setup this copy. The system allows using these copies when performing a query or in the data composition system operation. For duplication and mechanism operation purposes, primarily, "table" is used. The term "table" means a database table and not 1C:Enterprise configuration system object. For instance, an accumulation register consists of several database tables. To successfully use a copying mechanism, all database tables will be duplicated which are related to a specific register.
At the same time, the database copying mechanism does not constitute the active database backup tool. The created copy is not a sound information base and the user can't work with it as with a usual database.

IMPORTANT! The database copying mechanism does not support the operation in the information base file variant.

The copying mechanism is used for the Data Accelerator service operation (as a compulsory item of Data Accelerator) and also can be used to transfer the data used for analytical reports drafting to the external (in respect of the main base) database server to decrease the load on the database server serving the main database.
Both the embedded mechanism of 1C:Enterprise system and the external services (which is determined at each copy setting stage) can be used for the databases synchronization. The DBMS tools for creating external copies can be used only if all the following conditions are met:

1. The created copy will be used **only** with the help of the **database copying** mechanism.

2. **All tables** of the main database are included in the copy (**full copy** is created).

The embedded replication features of 1C:Enterprise system shall be used in all other cases.

The database copying mechanism does not support data partitioning. The tables marked as copied data must not be included in any partitioner.

The following DBMS can be used as the DBMS used for the database copying: Microsoft SQL Server, Oracle Database, PostgreSQL. The server cluster DBMS type can differ from a database copy DBMS type. Version, bitness and DBMS provider can be different.

27.2. Mechanism structure

Actually, when creating a database copy, the tables being the copies of the relative tables of the information base objects are created in the copy database. A copy tables content is determined with the help of `Content` property of the object describing a certain database copy (`DatabasesCopiesManager` object). In the process of operation, the system tracks the changes in the transferred data and, on command, transfers these changes to each database copy. Data transfer to the registered database copies is performed using a special method of the program interface (`DatabasesCopies.Update()`). There is a possibility to update the database copies asynchronously, for example, by using a regular task (similar to updating the full-text search index). In other words, a database copy becomes a "real" copy only after the changes registered in the transferred tables are transferred into it.

When a database copy is updated by selecting `DatabaseCopy.Update()`, internally the following dissimilar (as to their content) operations are performed by the platform:

1. Data is initially exported to a new copy table. For that purpose, one or more system background jobs are used. The said export is made upon each modification of copy data structure (adding or deleting an attribute or an object itself).
2. Current data is exported to an existing copy table. This action is performed without using background jobs. In this scenario, all successfully completed actions performed with data stored in the main database are exported. Obviously, modified data is saved and exported for those objects only which are marked as placed in the said copy. If the scope of modifications is large or update interval is long, - the said export (and obviously execution of `DatabaseCopy.Update()`) can be time-consuming.

The update process described above is used if the standard data replication type is selected for a certain database copy (`DatabaseCopyManager.ReplicationType` property). If an external type of database data replication is selected for a database copy then:

- 1C:Enterprise system performs no actions on the creation of tables in the copy.
- 1C:Enterprise system performs no actions on the data synchronization between the main database and its copy. `DatabaseCopies.Update()` method use is not required.
- The personnel setting the database copies parameters are fully responsible for the creation of tables of the required structure as well as for the data replication mechanism and regularity.
- In this case, 1C:Enterprise only transfers the data obtainment query to the database copy instead of DBMS of the main database. Any inconsistency in the data structure between the copy and the main base will result in a query error.
- 1C:Enterprise considers that data in this copy is always up-to-date.

As has already been stated, the system uses the database copies when working with `Query` object in the data composition system. The following conditions must be met so that the data obtainment is performed in the database copy:

1. All tables used in the query or in all data sources of the composition system must be in one copy. The term "all" means the following tables used in the query (directly or indirectly):
   - Tables and attributes used as query sources.
   - Tables and attributes from which any data in the query is obtained (for example, in the case of address put "through a dot").
   - Tables and attributes engaged in the conditions of restriction of access to the data (if any).
2. The copy data relevance meets the developer's requirements.
3. If the table contains a query for which the selection in the copy is set the condition in the query must correspond to the selection to the copy. If the query contains no condition but the copy contains the selection for this table such query will be transferred to the database or copy in which the selection for this table is set.

Tables selection for data accelerator external processing, which is located on ITS disk in the section of the developer's methodological support (https://its.1c.ru/db/metod8dev#content:5951:hdoc), can be used to simplify the selection of tables which must be located in the database copy.

The required data relevance setting is managed with the help of `Query.RequiredDataRelevance` property and

RequiredDataRelevance data composition output parameter. This parameter may have the following values:

- Relevant -in this case, the data will be obtained only from the main database.
- Any -in this case, the data from the copy containing all tables necessary for the query execution will be used and these tables contain the data of required relevance (RequiredTimeDataRelevance property). If there is no required database copy the query will be performed in the main database.
- Auto -the platform automatically decides on the work with a copy. In this case, Relevant value will be used for Query object and Any -value will be used for the data composition system.

If the required data relevance is set in the value Any or Auto the degree of the irrelevance of the data used for the information retraction can be specified. For example, the data of the last day can be sacrificed for a sophisticated analytical report without losing the accuracy and adequacy of the result. In this case, the beginning of the current day can be specified with the help of Query.RequiredTimeDataRelevance property (or RequiredTimeDataRelevance data composition output parameter).

The platform allows obtaining the information on the data source and its relevance after the data retraction (and -during the data composition for the data composition system):

- QueryResult object provides the following properties:
  - RelevantData -allows determining whether the data obtained by the query is relevant or irrelevant.
  - RelevanceTime -contains the date and time of the last data update in the copy. For the latest data, contains the time of the query initiation.
  - DatabaseCopy -contains the name of the database copy used for the data extraction.
- The composition system expression language contains RelevantData(), DataRelevanceTime(), and DatabaseCopy() functions. The purpose of these functions fully corresponds to the query result properties of the same name.

The output to the report of the information on the information source and retrieved date relevance can be managed using the data composition system. OutputDataRelevance and OutputDatabaseCopy parameters are intended for it. However, it should be noted that if a data composition schema has nested schemas, each data composition schema can either be part of a main data base or any of its copies which is in compliance with requirements specified in the query being executed.

If several copies are set in the information base and each of these copies meets the executed query requirements the system will randomly select a copy from the list of suitable. There is no copy selection mechanism in the system in this case.

27.3. Data accelerator

NOTE. Available only for CORP licenses.

Data accelerator -is a special system of database management in which the database is fully located the production server core (in-memory DB) of 1C:Enterprise servers cluster. The Data accelerator operates only in cooperation with the database copying mechanism. The Data accelerator operates only when the cluster is started under the management of a 64-bit operating system. The Data accelerator does not support operation under the management of a 32-bit operating system.

The Data accelerator mechanism is optimized to execute the analytical queries processing large data quantities in DBMS and returns a small number of entries as the result. It is the recommended scenario of the mechanism use. When the Data accelerator is used in other scenarios it may take the Date accelerator longer to execute the query mechanism as compared to the use in the main database.

The general scheme of the Data accelerator operation is as follows:

1. The production server on which the Data accelerator will be executed is set.
2. The databases copying with the set property Built-in Data accelerator is performed. The built-in synchronization of 1C:Enterprise is always used when the Data accelerator is used.
3. The list of tables to be in the copy is set.

TIP. It is not recommended to place the whole business database in database copy intended for the work with the Data accelerator. The list of tables necessary for the acceleration of one or several reports critical for work is recommended to be placed in the database copy.

4. The database copy initial update is started.
5. After the update completion, the reports can be used by the Data accelerator.

According to the mechanism description, all data of the Data accelerator is stored in the server core. It means that the production server restart results in the database physical destruction. It means that it is necessary to perform the initial update of the database copy used for the Data accelerator after the production server restart.

Do not also forget that the application developer is responsible for the setting of the update process for the data of the database copy intended for the work with the Data accelerator.
The Data accelerator mechanism has its own restrictions in addition to the database copying mechanism restrictions. The following is not supported:

- The attributes storing the data exceeding 4 KB are not supported. The empty values will be returned in the case of an attempt to get such attributes.

27.4. Specifics and limitations

A query can't be executed on a database copy if any of the following conditions is met during the query execution:

- There is an open transaction in the current session.
- A temporary tables manager is set in `Query` object.

`DataVersion` property of the data located in the database copy may mismatch with the value of this property in the main database. Only the relevant data must be used to obtain the correct value for such property (`RequiredDataRelevance` property is set in `Relevant` value).
Do not forget the following specific feature when setting the list of tables in which the copies will be placed with the help of embedded language:

- The new tables will be added to the copy after the object of the type `DatabaseCopyManager` is written and `DatabaseCopies.Update()` method is executed.
- At the same time, the tables will be deleted from the copy immediately after the object of the type `DatabaseCopyManager` is written without waiting for the update of the information in the database copy.

The following access right is necessary for the database copying mechanism operation:

- In the information base being the data source:
  - `Administration` for the database copies list management.
  - `DataAdministration` for the management of the properties of the tables included in the database copy.
  - The following access rights are necessary for each configuration object synchronized with the external database: `Read`, `Add`, `Change`, and `Delete`. In this connection, it is recommended to create a separate user on whose behalf the data will be synchronized and who will have the necessary access rights.
- The user on whose behalf the platform connects to DBMS in which the copy data is stored must have the following access rights:

- Create, change, and delete tables in the database.
- Read, create, change, and delete the data in all tables of the database being a copy.

27.5. Copy content management

Copy content is managed as follows:

1. Create a database copy (select `DatabaseCopy.Add()`). Alternatively, you can get details of an existing database copy (select `DatabaseCopy.Find()`).
2. So far as a database copy so selected is concerned, tables and fields are set up using `DatabaseCopyManager`.
3. Further, new copy content settings are saved in the main database (select `DatabaseCopyManager.Write()`).
4. To update data in a database copy (including copy table structure), use `DatabaseCopy.Update()` and specify the database copy being set up as a parameter.

Tables being part of a database copy are managed by `DatabaseCopyManager.Content`. However, in addition to tables 1C:Enterprise enables you to manage their fields (table structure) which are saved in a copy. The following configuration object elements can be managed:

- For reference configuration objects:
  - Object attributes.
  - Tabular sections and tabular section attributes.
  - Standard and common attributes intended to separate data are always available in a copy.
- For registers:
  - Resources and register attributes.
  - Register dimensions are always available in a copy.

The table structure can be managed as follows:

1. The first method is to use dedicated properties describing behavior of all attributes of a specific kind. This method is recommended to be used as a basic method.
2. The other method is to set up display of specific object attributes in a copy only.

Dedicated properties available for `DatabaseCopyManager` and `DatabaseCopyContentElement` include the following:

- `UseValuesStorages`. This property manages the way attributes of `ValueStorage` kind are used in a copy.

- `UseStringsOfUnlimitedLength`. This property manages the way attributes of StringOfUnlimitedLength kind are used in a copy.
- `UseOtherFields`. This property managers the way attributes other than those of StringOfUnlimitedLength or `ValueStorage` kinds are used in a copy.
- `UseTabularSections`. This property defines, whether tabular sections are available in a database copy, or not. Availability of tabular sections attributes in a copy is managed by• `UseValesStorages`, `UseStringsOfUnlimitedLength` and `UseOtherFields`.

A database copy content element (object of `DatabaseCopyContentElement` kind) has `Fields` property. This property is intended to manage availability of specific configuration object attributes in a copy on a attribute-by-attribute basis. For that purpose, use `DatabaseCopyContentElementField.Use` property.

To manage content, use `UseDatabaseCopyContentElementField` system enumeration. This enumeration consists of the following three values:

- `Use` - an attribute (or tabular section) is used in a copy.
- `Do not use` - an attribute (or tabular section) is not used in a copy.
- `Auto` - use of an attribute (or tabular section) in a copy is defined by the field parent or system global-level settings.

By default, each attribute which is part of a configuration object and marked as used in the database copy content, is assigned `Use` property set to `Auto` (however, it can be modified). The actual usage scenario is defined by the following algorithm:

- If `Use` property for a specific attribute is set to any value, other than `Auto`, - the said value is used. If Use property is set to `Auto`, -to define usage scenario object of `DatabaseCopyContentElement` kind is analyzed.
- Depending on the field for which usage is verified, the following properties of `DatabaseCopyContentElement` are analyzed:
  - If a field is a tabular section - `DatabaseCopyContentElement.UseTabularSections` is analyzed.
  - If a field is not a tabular section, the following properties are analyzed (depending on the field kind):
    - If`ValueStorage` - `DatabaseCopyContentElement.UseValuesStorages` is analyzed.

    - If string of unlimited length - `DatabaseCopyContentElement.UseOfStringsOfUnlimitedLength` is analyzed.
    - If other kinds - `DatabaseCopyContentElement.UseOtherFields` is analyzed.

  If value of a property being analyzed is different from `Auto` -, the said value is used. If a property being analyzed is set to `Auto`, `DatabaseCopyManager` is analyzed. Analysis rules are described below.

- If in `DatabaseCopyManager` object values of `Use...` properties are different from `Auto` -, the said values are used. If these properties are set to `Auto`, the following algorithm applies:
  - If a database copy with standard replication is used, the following rules apply:
    - So far as tabular sections and attributes of `ValuesStorage` are concerned, `Do not use` value is used.
    - So far as attributes of other kinds are concerned, `Use` value is used.
  - If a database copy with external synchronization is used, - all attributes are assigned `Use`.

Therefore, to determine whether an attribute has to be made available in a database copy (or not), the following objects chain is used (from the first to the last one):

1. `DatabaseCopyContentElementField.Use`;
2. further`DatabaseCopyContentElement`;
3. subsequently`DatabaseCopyManager`;
4. further default values set by the platform.

As soon as a specific usage value is defined at any step, -search is aborted.

By way of example only, let's add a catalog to a copy (so that the entire catalog is added, except attributes of `ValuesStorage` kind). It is assumed that a database copy with `TestCopy` name has been already created and is now available for use.

```
Copy = DatabaseCopies.Find("TestCopy");
CopyElement = Copy.Content.Add(Metadata.Catalogs.Goods);
CopyElement.UseOtherFields = UseDatabaseCopyContentElementField.Use;
CopyElement.UseStringsOfUnlimitedLength =
UseDatabaseCopyContentElementField.Use;
CopyElement.UseTabularSections =
UseDatabaseCopyContentElementField.Use;
CopyElement.UseValuesStorage =
UseDatabaseCopyContentElementField.DoNotUse;
Copy.Write();
DatabaseCopies.Update(Copy, False);
```

If you need to specify, whether attributes are available in a copy, or not, the example will be somewhat different. In the example below `Goods` catalog will be added to a copy. The said catalog will have `SKU` attribute available in a copy. It is assumed that a database copy with `TestCopy` name has been already created and is now available for use.

```
Copy = DatabaseCopies.Find("TestCopy");
CopyElement = Copy.Content.Add(Metadata.Catalogs.Goods);
CopyField = CopyElement.Fields.Find("SKU");
CopyField.Use = UseDatabaseCopyContentElementField.Use;
Copy.Write();
DatabaseCopies.Update(Copy, False);
```

To determine the actual status of configuration object attribute usage, select `DatabaseCopyContentElementField.GetCurrentUsage()` method in the database copy. This method takes into account all distinctive features of the way the actual usage value is retrieved in.

27.6. Copy management

Database copies management standard function is intended for the database copies management in 1C:Enterprise.

Chapter 28. Global Search

28.1. General information

Global search -is the mechanism which allows both searching in the information base data and any other actions necessary in the application solution.
By default, the global search searches in various areas of information base, makes arithmetic calculations, and allows following the navigation links (both to the configuration objects and different resources on the Internet).
The developer may add a search mechanism in a way that the input of certain data in the search string the application solution will return the action banned by default. For example, a simple specification of bar-code symbols will result in the search of this line using all types of search. At the same time, if `bc` (the text of the type `bc 12345678` will be in the search string) is typed in front of the bar-code the search will be performed only by the attribute `Bar-code` in the catalog of `Goods`, which is preliminarily indexed.
Let's consider the global search mechanism arrangement and operation scheme.
The global search includes:

1. The global search plan describing what an where will be executed.
2. The embedded language method implementing the necessary actions.
3. The event which allows changing the global search plan depending on what the user is typing in the global search field.
4. The search mechanism itself which:
   - Performs the actions included by the developed in the global search plan.
   - Forms the search result.
5. The event which allows processing in a certain way the selection of any search result formed during the preceding steps.

From the point of view of user actions and the developer's reaction to them, the search looks as follows:

1. The global search plan by default and search reference information are set. It is done once when the system is started.
2. The user inputs the information in the global search field.
3. The developer can process this input and modify the search plan which will be used in this certain search.

4. The search results are displayed as and when obtained. The result display depends on the value of priority of each of the used search types.
5. When selecting from the search results the system allows processing this selection for the special implementation in the required cases.

Let's consider the arrangement of the described tools in more details.

28.2. Global search plan

The global search plan is accessible through `GlobalSearch` global context property (the object of `GlobalSearchManager` type). Therefore, the reference to the methods of the program interface will look as follows: `GlobalSearch.NameOfCalledMethod()`. For simplification, the global context property name will be omitted (except, evidently, the examples in embedded language).

All main actions can be performed with the help of the manager:

- To perform a search (`BeginSearch()` method).
- To set and get the global search plan by default (`SetPlan()`/`GetPlan()` methods).
- To set and get the search execution description (`SetDescription()`/`GetDescription()` methods).
- To set and get the text to be displayed in the global search field (`SetInputHelp()`/`GetInputHelp()` methods).

The global search plan determines where and how the search mechanism will search. `GetPlan()` method returns an object of `GlobalSearchPlan` type, which consists of `GlobalSearchItem` objects. Each global search item describes one "elementary" action which the platform will perform. Action can be:

- Standard. It is the action performed by the platform.
- Set using 1C:Enterprise language. It is the action implemented by the application solution developer.

The standard actions are described by the system enumeration `ConventionalGlobalSearch`. The following standard types are provided:

- `AllFunctions`. Regular search conditions. The search by configuration objects representations is made. The search is made only if AllFunctions dialog is enabled in the settings. The same text as displayed in All functions window is used as the representation. The list of accessible objects (in which the search is executed) also fully corresponds to the list of objects in All functions dialog.
  The localized representation of the standard functions is used for the search in the standard functions.

The search result returns not more than 10 found values.
On search result choice, `OnGlobalSearchResultChoice` handler is not called and the relative meta-data object form is opened.
The search is executed on the server in the background job.

- `Expression`. An attempt to calculate the search string as an expression is made. The search is executed in the client.
- `GlobalStandardCommands`. Regular search conditions. The search is made by Service and Settings menu commands. The search is executed in the client. The following search strings are processed:
  - all functions - to open All functions dialog box (Service and Settings- All Functions), if this option is enabled in the client application settings.
  - favorites -for the user's favorites opening.
  - edit scale- for the interface scale editing form opening (Service and Settings- Edit scale).
  - history -for the user work history opening.
  - calendar- to open calendar form (Service and Settings- Calendar).
  - calculator - to open calculator form (Service and Settings - Calculator).
  - home page setting - to open home page setting form (Service and Settings- Settings- Home Page Settings).
  - panel customization - to open panels customization form (Service and Settings- Settings - Panels Customization).
  - sections panel customization - to open sections panel customization form (Service and Settings- Settings - Sections Panels Customization).
  - about program - to open About program form (Service and Settings- About program).
  - notifications -for Notification center opening.
  - parameters - to open client application parameters editing form (Service and Settings - Settings- Parameters).

  On search result choice, `OnGlobalSearchResultChoice` handler is not called but the relative command is performed.
- `Data`. The search in the data is executed with the help of the full-text search. Not more than 30 values are returned. The search is executed on the server in the background job.
- `UserWorkFavorites`. Regular search conditions. The search is executed in the client.

- `UserWorkHistory`. Regular search conditions. The search is executed in the client.
- `FunctionMenu`. Regular search conditions. All commands containing each sub-line in any position in their representation are returned. The search is performed not only by the commands presentations, but also by the commands and subsystems groups presentations (except the group See also). If one (or several) out of searched sub-lines are contained in the command group name (except for the group See also) or subsystem then the search in the commands of this group (subsystem) is executed without taking this sub-line(s) into account. The search is executed also in the functions which display is currently disabled by the user (or developer) but the user can enable the display of these functions. The search is executed on the server in the background job.
- `URL`. If the search string begins with one of the sub-lines: http://, https://, e1c://, e1cib/, e1ccs/ then the search will result in a link. Execution in the client.
- `CollaborationSystemConversation`. The search string is divided into sub-lines. Space symbol acts as the practitioner. The conversations and users containing each of sub-lines in any position in their representation (conversation title, user name) are returned. Not more than 15 values are returned. The search runs on the collaboration system server.
- `CollaborationSystemMessage`. The search string is divided into sub-lines. Space symbol acts as the practitioner. All messages containing each sub-line in any position in their text are returned. The search is executed only in the messages of non-context conversations. Not more than 15 values are returned. The search runs on the collaboration system server.
- `Help`. The search in the help is executed analogously to the search string in the help window. Not more than 10 values are returned. The search is executed on the server in the background job

The phrase "search standard conditions" in the list of the standard types of search means the following:

- The search string is divided into sub-lines. Space acts as the partitioner in such division.
- The search is executed by the beginning of the word.
- If the sub-line contains letters and digits then:
  - a sequence of letters is a separate sub-line;
  - a sequence of digits is a separate sub-line.
- In a sub-line consisting of digits, leading zeros can be both significant and insignificant.

- In case of a hierarchy, an item is found if one of the following conditions is met:
  - it contains all searched sub-lines;
  - it contains at least one searched sub-line and its parents contain the remaining sub-lines.

By default, the global search plan consists of the following types of search and the order of output of the result of each type of search:

- `Expression`. Order: 10.
- `URL`. Order: 10.
- `UserWorkFavorites`. Order: 20.
- `UserWorkHistory`. Order: 30.
- `GlobalStandardCommands`. Order: 40.
- `AllFunctions`. Order: 50.
- `FunctionMenu`. Order: 60.
- `CollaborationSystemConversation`. Order: 70.
- `CollaborationSystemMessage`. Order: 80.
- `Data`. Order: 90.
- `Help`. Order: 100.

To create a new search type, add the description of the embedded language method which will be "responsible" for the new type of search to the global search plan. Let's suppose that the following code in the embedded language is present in the application module (in `OnStart` event handler):

```
Procedure OnStart()
  GlobalSearchPlan = GlobalSearch.GetPlan();
  GlobalSearchPlan.Add("SearchExecutionMethod",
"ServerGlobalSeachModule", True, True, 5);
  GlobalSearch.SetPlan(GlobalSearchPlan);
EndProcedure
```

This code means the following:

- The global search text plan is obtained at first (the global search plan by default which includes all standard types of search will be obtained).
- Then, a new type of plan is added to this search plan (`GlobalSearchPlan.Add(...)` line).
- The changed global search plan becomes the global search plan by default.

The method executing the search operation has the following characteristics:

- This method can be executed both in the client and on the server.
- The method executed on the server can be executed synchronously or asynchronously (background job).
- Some additional parameters can be transferred to the method.

- The method operation result has a certain (set) position among all results of the global search operation. Actual -values of sorting.

So:

- "How" we search -means how the search string will be interpreted in each search type and what platform mechanisms will be used to execute the search.
- "Where" we search -means where the search is executed: in the client or on the server.

28.3. Search execution and search result formation

After the user launches the search, for example, by pressing ENTER in the global search string (in the application main window), the system performs the following actions:

1. Analyses the global search plan.
2. The client types of search and the server types of search are divided from this plant.
3. The sever types of search are executed first. All server types of search present in the used global search plan are collected in one asynchronous call. On server side:
   - The synchronous types of search are executed consecutively.
   - The background jobs for the background types of search are started.
   - The management is returned to the client with the results of the synchronous types of search.

   It can be stated that all server types of search are asynchronous from the point of view of the client application. In other words, the client application does not wait for the server work completion to begin executing the client types of search.
4. All client types of search are executed (all these types of search are synchronous).
5. The search result to be displayed to the user is formed. The completion of the server types of search is awaited simultaneously. The value of the order for each type of search is used to build the search results list.

It shall also be pointed out that the search is launched not only be pressing ENTER but also as and when the user types if the user does not type for some time. When the search string is changed, a new search is launched ad all background jobs executing the preceding search are canceled and launched with the new search string.

The search result displayed to the user is formed on the basis of `Order` property value of each item of the global search plan. The smaller the value of this property, the higher the result of this type of search will be in the search results.
The search result is formed dynamically, as and when the search results are obtained, and the following rules are applied:

- The search result construction begins after any type of search with the minimal order is completed (for the used global search plan). The results with the bigger order are not displayed until all types of search with smaller order are completed. Each new search result is added to the preceding one.
- After all search results with the current order value are displayed, the search results for the next (ascending) order value are displayed. The process is repeated until all search results are performed or until a new search is initiated, or until any item of the search results already displayed to the user is selected.
- The search result is output without any reordering (including after the completion of all types of search). Within the framework of one type of search, the ordering is determined by the search type itself.

Do not forget that the client application interface does not react to user actions during the synchronous types of search execution.
Thus, the following recommendations on the search types formation (and their order in the global search) can be formulated:

- All synchronous types of the search must return a result within the minimum time.
- It is desirable to set all synchronous types of search with the small order value as compared to asynchronous types of search.
- It is better to set the asynchronous types of search with biggest order number.
- The order of search types execution within one order value is undefined.
- If there are several types of synchronous search with one order value but a situation when a search type can take longer time is possible it is better to increase the order value of such type of search.

28.4. Search string processing and new type of search

The operation scheme described until now refers to the global search operation by default. But the question remains: how to implement own search type and how to integrate it in the global search mechanism? This section is intended to answer this question.

The system calls `OnGlobalSearch` event handler during the search string input. This event is analogous to `AutoComplete` form output field event. The line input by the user (`SearchString` handler parameter) and the global search plan to be used in this search (`GlobalSearchPlan`) are transferred to the handler.

This handler is intended to modify the used plan so that to include the types of search depending on the data input by the user. At the same time, the types of search already existing in the transferred global search plan can be both left and deleted from this plan. If there are any types of search that must be present always -these types of the search shall be set using `SetPlan()` method. It is recommended to do upon the system start. Also, do not forget not to execute any search in this handler because it has no place for the search results! Only the global search plan to be used once for a certain search string is formed in the handler.

So, if we want to allow the creation of any objects by the user by pressing `+` it can be done the following way:

```
Procedure OnGlobalSearch(SearchString, GlobalSearchPlan)
  If StrStartsWith(MSearchString, "+") Then
    GlobalSearchPlan.Clear();
    GlobalSearchPlan.Add("CreateGSCommand", "GeneralModuleClient",
False);
  EndIf;
EndProcedure
```

The configuration shall also contain the client non-global general module `GeneralModuleClient` with the following code in the embedded language:

```
Procedure CreateGSCommand(SearchString, SearchResults,
AdditionalParameters) Export
  SearchResults.Add("+Picture", "Picture", PictureLib.CreateListItem);
  SearchResults.Add("+Basket", "Basket", PictureLib.CreateListItem);
  SearchResult.Add("+Box", "Box", PictureLib.CreateListItem);
  SearchResults.Add("+Dog", "Dog", PictureLib.CreateListItem);
EndProcedure
```

An attempt to select any of the displayed lines will result in the message that the URL has a wrong type. The following section is dedicated to this problem solution.

There is one problem in the implemented example: if the text `+2+2` is input in the search string it will result in no expression calculation. The reason is obvious: we have deleted all types of search by default which were in the global search plan. Now, there are two ways to solve the problem:

1. To compare the search string for absolute equality: `SearchString = "+"`.
2. Add expression calculation to the global search plan.

Let's use the second option. To do it, add the text to `OnGlobalSearch` handler so that it looks as follows:

```
Procedure OnGlobalSearch(SearchString, SearchPlan)
  If StrStartsWith(MSearchString, "+") Then
    SearchPlan.Clear();
    SearchPlan.Add(ConventionalGlobalSearh.Expression);
    SearchPlan.Add("CreateGSCommand", "GeneralModuleClient", False);
  EndIf;
EndProcedure
```

Now, the input of the expression `+2+2` will give the required result: the expression calculation and the creation of the picture, basket, box, and dog.
Thus, analyzing the text input by the user we can randomly form the global search plan. A unique global search plan can be formed for each unique input by the user. Some additional parameters, which can accelerate the search or provide the method with additional parameters absent from the global search string, can be transferred to each method. These parameters can be unique for each search string.

28.5. Search result processing

Each search type returns a set of search results consisting of the objects of `GlobalSearchResultItem` type to the client. Each item describes the search result (and representation). `SearchType` property allows specifying what type of search will provide the given result. The platform fills this property in but does not use it. The developer can act "depending on the circumstances".
If the search result returns a value "known to" the system nothing is needed to be done additionally. For example, having returned a link to an information object the system will automatically open the form for this link view.
But if the value returned by the user type of search requires additional processing it is necessary to use `OnGlobalSearchResultChoice` event handler of the client application module.
The selected search result item (`ResultItem` handler parameter) and the standard processing necessity attribute are transferred to this event handler.
Now, it is necessary to analyze the selected result and to perform the necessary action:

```
Procedure OnGlobalSearchResultChoice(ResultItem, StandardProcessing)
  StandardProcessing = False;
  If ResultItem.Value = "+Picture" Then
    // action
  OtherwiseIf ResultItem.Value = "+Basket" Then
    // action
  OtherwiseIf ResuktItem.Value = "+Box+ Then
    // action
  OtherwiseIf ResultItem.Value = "+Dog" Then
    // action
  EndIf;
EndProcedure
```

Any data structure executed in the client application can act a result item value. The data in this structure must also be available in the client.

If upon execution of server search an error is returned, `ErrorDetails` will be sent as a result item value (`GlobalSearchResultItem.Value`).

Whenever results are displayed with `ErrorDetails` object specified as their value, in a list of results `SpecificTextColor` style color is used as text and image color (for template images). If the said result has no representation assigned, Search Error is displayed. If the said result has no image assigned, `DialogStop` image is displayed.

When you select a result with a search error, `OnGlobalSearchResultChoice` handler is not displayed. Instead, a standard dialog box with an error message is displayed.

28.6. Other specifics

It may be necessary to redefine the help displayed in the global search field during the global search setting. The writing Search and the key combination for this field actuation are displayed there by default. Nevertheless, the developer can install another text, which will be more informative for the user, there. There is a couple of functions for it: `GetInputHelp()`\`SetInputHelp()`.

Thus, if the following example is written in `OnStart` event handler:

```
Procedure OnStart()
  GlobalSearch.SetInputHelp("What is registered in luggage?");
EndProcedure
```

The search help setting option is another useful specific feature of the program interface.

The search help is displayed either instead of the search history (if no search has been run yet) or after clicking the hyper-link What To Search For in the search history if the search has already been run.

GetDescription()/SetDescription() methods are intended to redefine the text displayed when clicking this hyper-link.
The addition to OnSatrt can be made (an example is given above) so that to get the following text:

```
Procedure OnStart()
  GlobalSearch.SetInputHelp("What is registered in luggage?");
  GlobalSearchDescriptionArray = New Array;
  GlobalSearchDescriptionArray.Add("Input");
  GlobalSearchDescriptionArray,Add(New FormattedString("+", New
Font(,,24, True)));
  GlobalSearcdescriptionArray.Add(" for quick creation of objects");
  GlobalSearchDescriptionArray.Add(Symbols.LF);
  GlobalSearchDescriptionArray.Add(" or arithmetic expression for
result obtainment");
  GlobalSearch.SetDescription(New
FormattedString(GlobalSearchDescriptionArray));
EndProcedure
```

Developing an application solution, it may be necessary to call the global search from the embedded language so as if it has been done by the user. For example, to run the search by the search string formed in the embedded language in accordance with some rules.
BeginSearch() method can be used in this case. The method gets a search string for input, places this string in the global search field, and launches the search. The result is displayed in the same manner as in the case of interactive input. Search everywhere command accessible in the context menu of some form items can be an example of such use. This command is available for a form table, spreadsheet document form field extension, and for a formatted document form field extension. This command places the selected (in the management item) text in the global search field and launches the search.

Chapter 29. Development for mobile devices

29.1. General information

The system of programs "1C:Enterprise" allows to develop application that operate on mobile devices (smartphones, tablet computers). The developed applications can function on mobile devices that run Android (http://www.android.com/), iOS (http://www.apple.com/ios/) and Windows (https://www.microsoft.com/windows/) operating systems. So far as Windows OS is concerned, devices with touch screens are supported.
Depending on the task, the developer can develop an application for the mobile platform or an application that functions in the mobile client. There are significant differences between these development options:

- **Application on a mobile platform** is an analogue of a thin client working with a file information base located on the same computer as the client application itself. The application on the mobile platform works with the infobase located on the mobile device. It follows from the above that the client and server of application operate on a mobile device. Presence of the infobase on the mobile device means that to synchronize data between the mobile device and the main system (if any) it is necessary to implement the necessary tools.
  In addition, the application on the mobile platform - is a separate technology, which should be used to develop a completely new solution or significantly improve the current one.
- **Mobile client** is an analogue of a thin client for an ordinary computer that uses a web server to access the infobase. In other words, there is no infobase on the mobile device when the mobile client is working. In the mobile client, the client code is executed on the mobile device and the server code is executed on the application server. At the same time, as it is obvious, the application server does not function on a mobile device. Since there is no infobase on the mobile device, it means that the mobile client works only if there is a connection between the mobile device and the web server. The mobile client cannot work offline.

In order for the application to work in the mobile client, it is necessary not to rewrite it (as in the case of the application on the mobile platform), but to adapt it. When adapting, it should be remembered that the mobile client has a specific set of features and limitations that should be taken into account when finalizing the application. An essential feature of the mobile client is that it supports the technology of configuration extensions.

- **Mobile client with offline mode** - is a mobile client which supports operation in offline mode, i.e. without connection to an infobase. For an application to be able to support an offline mode, an application which can be run by a mobile client needs to be updated heavily. For an application which supports an offline mode, select metadata which is available in the said mode and implement offline forms and infobase data synchronization options . So far as development complexity is concerned, a mobile client which supports offline mode is placed somewhat in between a mobile client and a mobile application.

Mobile application interface - interface analog Taxi. In general, the interfaces of the mobile platform application and the mobile client look similar. However, the operation of the interface in each of the possible variants has its own specifics, which should be considered during the development of the application.

The following terms will also be used in this chapter:

- **Mobile platform**.
- **Mobile version "1C:Enterprise"**. This term will be used when it is necessary to briefly describe all the opportunities provided by 1C for the development of applications that work on mobile devices. Thus, the following two phrases will be equivalent:
  - Mobile platform and mobile client provide access to work with short messages (SMS).
  - Mobile version "1C:Enterprise" provides access to work with short messages (SMS).
- **Mobile application** - will be used when it is necessary to describe an executable file that can be executed on a mobile device. In a very close approximation, it is possible to assume that the mobile application is .apk-file for Android OS .ipa-file for iOS and .appx-file for Windows.

A developer tool is provided for development: developer's mobile platform or developer's mobile client (mobile version of "1C:Enterprise" for developer). This version has a number of differences from the mobile application, which is distributed through the application store: it cannot be hosted in the

application store, it has all the possible permissions that an application solution might require, and it provides some advanced features regarding the mobile application.
To distribute a mobile application in both cases, it is necessary to use the mobile application builder, which is a specialized tool that facilitates the creation of a package that can be hosted in the application store of the appropriate mobile operating system provider.
This chapter is devoted to the features of the development of applications running on mobile devices. At the same time, the features common to the application on the mobile platform and the mobile client are described in one section. In the case when described features common to the mobile platform and mobile client, the term "mobile version "1C:Enterprise" will be used. Features specific to only one development option will be described in its section. Unless otherwise stated, the term "mobile application" shall mean any application that can operate on a mobile device (mobile platform application or mobile client).
A developer dealing with applications for mobile devices is highly recommended to understand the background of mechanisms used at the documentation level to the extent having regard to a specific mobile operating system. Respective resources for developers are available online:

- OS Android: https://developer.android.com/.
- OS iOS: https://developer.apple.com/.
- Windows OS: https://developer.microsoft.com/ru-ru/windows.

29.2. Application interface

29.2.1. General information

The interface of the mobile version "1C:Enterprise" is analogous to the Taxi interface, with the ability to switch between Windows using the main menu of the application. Navigation in the applicaition depends on which form is currently active:

- A form that does not have an owner and does not store data.
  - Example: list form, processing form.
  - The form is opened by the command from the function menu - in the left part of the window title there is a button to open the main menu of the application. The main menu of the application is accessible by a touch on the left side of the device screen or a special button.

- The form is opened by other commands or from the 1C:Enterprise language - in the left part of the window title there is a button to return to the previous window (<). When closing of such form, the form that was active at the time when the command to open the closed form was invoked will be activated. The main menu of the application is accessible by a touch on the left side of the device screen.

- Form with an established owner.
  - Example: selection form, object form, opened from the list.
    - When such windows are opened sequentially, a chain of open windows is formed, in which only the most recently opened form is available to the user. At this, closing the current form activates the previous form of the chain, i.e. the form from which the closed form was opened. It should be understood that the return may not occur in the form in which the user was before switching to the closed form.
    - The left part of the window title contains the button to return to the previous window (<). The current window will be closed at this.
    - The main menu of the application is accessible by a touch on the left side of the device screen.
  - A form that does not have an owner and saves data.
    - Example: the form of object creation, called from the main menu of the application or using the 1C:Enterprise language.
    - When closing of such form, the form that was active at the time when the command to open the closed form was invoked will be activated.
    - The left part of the window title contains the button to return to the previous window (<). The current window will be closed at this.
    - The main menu of the application is accessible by a touch on the left side of the device screen.
  - Forms that block the interface.
    - Example: a form, for which the property Mode of window opening is set in value Block the owner window or Block the hole interface.
    - The window title contains the button associated with the form command:
      - In the presence of the saved data (flag of the attribute Persistent data) - standard button Cancel.
      - If there are no persistent data - standard button Close.

    - If the form contains one of commands Cancel, Close, No -this command is displayed.
  - The main application menu is unavailable for a form that blocks the entire interface. For a form that blocks the owner window, the main menu of the application is available by a touch on the left side of the device.

When starting the mobile application, the full-screen image specified by the application developer is displayed. At a certain point in time, the logo and copyright of the company "1C" appear at the bottom of the screen, overlapping part of the screen saver:

- In portrait orientation, - about 10% of the screen area.
- In landscape orientation, - about 15% of the screen area.

29.2.2. Global command interface

The interface of the mobile version "1C:Enterprise" is focused on the fact that one form is displayed on the screen at one time. Under it is released the maximum free space on the screen. Therefore, the commands that generate the command interface of the main section (navigation and action panels) on the personal computer are collected in the main menu of the application. The main application menu also contains a command to return to the home page, a list of sections (the commands of each section are displayed as a submenu), as well as commands to display information about the program and jump to the list of applications. The main menu of the application is opened by a touch on the left side of the screen (if available) or by a special button ≡ located on the left side of the window title.

There is a possibility to configure the role visibility of the command interface.

29.2.3. Home page

The home page supports displaying a different number of forms, depending on the option you are using:

- The mobile platform - supports displaying of only one form.
- The mobile client - displays all available home page forms (the form selection algorithm and the display method are described below).
- An offline mobile client - is similar to a mobile client with due regard to offline mode distinctive features (see below).

The displayed forms are selected according to the following rules:

- From the list of forms specified in the settings of the home page area, forms that are not available according to access rights are excluded.
- Then, the forms that are disabled by functional options are excluded from the remaining list.
- Displaying the home page forms:
  - Mobile platform - from the remaining forms (as a result of the previous steps), the first visible form is selected, while the desktop forms are traversed from top to bottom, first the left column, then the right one (if the location of the home page forms is set to two columns).
  - Mobile client - the forms selected for displaying are arranged in the form of bookmarks, where the bookmark switch is located at the top. When running the application, two tabs are displayed: the first visible form and a special tab, clicking on which leads to the loading of all other forms of the home page. After loading all forms, a separate tab of the home page will be created for each form.
  - Behaviour of an offline mobile client- in an ordinary mode is similar to that of a mobile client. If a mobile client is in an offline mode or Expect bad connection check box is checked in the application settings, when you select home page forms those which are unavailable in the offline configuration are omitted. As such, to make user operations independent from the mode, it is recommended to select an initial home page form which 1) has applied sense; and 2) is available in the offline configuration.

    If when home page is generated in offline configuration no forms are found, the home page displays a form with its content being dependent upon the mode of operation:
    - Offline:
      - Header: Offline mode.
      - Form text: Offline features are available.
    - Expect bad connection check box is checked:
      - Header: Connection is not established.
      - Form text: Offline features are available.
    - When connection with the main server is established:
      - Header: Connection is established.
      - Press Start… button in the form to load home page forms.

In addition to the form, the home page can contain the command interface of the main section. The components of the command interface of the main section on the mobile version is similar to that on the platform for a personal computer (except for the group see also). The main section command interface is displayed as a matrix of buttons (each button is associated with one

command), grouped by pages. The image on the button is obtained from the associated command. If a command does not have a custom image, the default image for that group of commands is used.
The commands are displayed so as to fill the working area of the window as much as possible. Groups of commands are placed on the pages so as to avoid transfer between pages, i.e. if after one group of commands the second does not fit completely in the remaining space, it starts with a new page. If all commands do not fit on the screen, they are placed on several pages that can be scrolled.

29.2.4. User message

The display of messages to the user depends on how many unseen messages are currently available. The first unseen message is displayed as a dialog box with the message text and the button OK. After the second message arrives, the All messages button appears in the advanced dialog box. The All messages button has a signature with the full number of incoming messages.
Clicking the All messages button opens the messages panel, which contains all unread messages. If the OK button is pressed in the dialog box or the message panel is closed, the list of messages is cleared. Clicking on a message activates the form element if the message is associated with one. To re-open the message list, it is necessary to re-call the action leading to the opening of the list.

29.3. Mobile version "1C:Enterprise"

29.3.1. General information

This section describes the mechanisms that are used in the same way in both the mobile platform application and the mobile client.

29.3.2. File operations

All applications on mobile devices operate within the border of a special mechanism that ensures the safe execution of programs (sandbox). Within the border of this mechanism, applications are given limited access to the file system of the mobile device. When developing a mobile application, there may be a need for access to the file system, for example, to store intermediate

data or to record documents generated in the system for later exchange with other computers.

Special functions are used to access these file system locations:

- `TempFilesDir()` - returns the temporary files directory for this application. If necessary, the directory can be cleaned up by the operating system.
- `UserDataWorkDir()` - returns the directory where it is recommended to store the user's "long-term" data.
- `DocumentsDir ()` - a document directory that are intended to be exchanged with an external (in relation to a mobile application) environment.
- `MobileDeviceLibraryDir()` - returns the library directory of the mobile device. The library type is determined by the method parameter. The following types of libraries are available: audio files, video files and pictures.

Regardless of the actual capabilities of the mobile OS, it is not recommended to use any other directories of the file system of the mobile device for security reasons, as well as the requirements of mobile OS developers for applications.

If necessary to use a file located on a mobile device in some way in the application, then for file selection it is possible to use the `FileDialog` object. This dialog box supports operation only in the mode of selecting an existing file (`FileDialogMode.Open`) and does not support the operation mode `FileDialogMode.Save`. If necessary to save the file on the mobile device, the interactive option of the method `GetFile()` should be used.

On a mobile device, the name of a file in the library may not match how the file is displayed by system applications. However, the file view cannot be used to identify the file, pass as parameters to file manipulation methods, and so on. The view is intended solely for display to the user. To obtain a view, use the `File.GetMobileDeviceLibraryFilePresentation()` method .

The following example shows how to select and display an arbitrary directory on a mobile device:

```
Dialog box = New FileDialog(FileDialogMode).ChooseDirectory);
Dialog box. Directory =
MobileDeviceLibraryDir(MobileDeviceLibraryDirType.Audio));
If Dialog box. Select() Then
  Report ("The directory is selected: " + Dialog box. Directory);
EndIf;
```

It should be remembered that in the directory of the libraries of the mobile device (in general) it is impossible to write file in the usual way. Writing is only possible with methods `GetFile()` or `GetFiles()`.

As a result, it is possible to formulate the following recommendations for working with files on the mobile device:

1. If the path to the directory (or file) starts with a line `content:`, then to write to this directory it is only possible to use `GetFile()` or `GetFiles()` methods. There is also a possibility to use these paths in the following object constructors: `File`, `BinaryData`, `Image`, `MobileDeviceApplicationRun` and as parameters of the following methods: `RunApp()`, `GotoURL()`, `FileCopy()`, `MoveFile()`, `DeleteFiles()`, `FindFiles()`.
2. If the path to the directory (or file) does not start with the string `content:`, then it is possible to use all the features of the mobile platform to work with files to work with such files and paths. Keep in mind that mobile operating systems work with different file systems. Therefore, the path separators and masks of all files are different on different mobile operating systems.
3. In order to get the mask of all files and the file path separator in a universal way that considers the operating system used and the place of execution of the code, the special methods `GetAllFilesMask()` and `GetPathSeparator()` should be used. For unification with the application code for a personal computer, it is possible to use methods that return the specified parameters separately for the client and server parts of the application: `GetClientAllFilesMask()`/`Servers()` and `GetClientPathSeparator()`/`Servers()`.

When displaying information from mobile device libraries, it may be necessary to show smaller copies (preview, view image) rather than full-fledged images. As a result, the probability of getting a memory shortage error when working with a mobile application that actively uses pictures or video files from mobile device libraries is reduced. In order to get a picture of the presentation for a file located in the library of a mobile device, use the `File.GetMobileDeviceLibraryFileThumbnail()` method. When using this method, it should be remembered that to get a picture of the view, the path to the file being processed must belong to the scheme `content` (begin with the line `content:`).

29.3.3. Work with pictures

For mobile devices, a regular picture with variants is used, which is used by the platform for a personal computer.

For iOS devices, the following pictures from the set are used:

- conventional screens - picture mdpi;
- Retina screens - pictures xdpi.

For Android devices, pictures are used that correspond to the screen resolution of the device used.

29.3.4. Forms

29.3.4.1. General information

For those mechanisms or configuration objects that are not supported by the mobile version of "1C:Enterprise", the properties of form elements associated with this mechanism or object will be unavailable. For example, the Shortcut key property or drag-and-drop events.

The principles and rules of forming a form for the mobile version of "1C:Enterprise" basically coincide with those for a thin client, however, there are some features that should be considered when developing forms of a mobile application:

- Smartphone screens are mainly oriented to vertical position (portrait orientation), so the form is significantly limited in its width and slightly limited in height.
- Mobile devices do not accept horizontal scrolling of the form, since the scroll bars on mobile devices are not displayed constantly, so the user may not notice that the form (report or table) shows only a piece of information. In this regard, when forming forms, the developer is obliged to make efforts to ensure that all elements of the form fit in its width, for example, using groups with the Horizontal if possible grouping. If such efforts are not undertaken, the form will be limited in operation, since the horizontal scroll bar in the forms on the mobile platform is not supported.

29.3.4.2. Available form elements

In the form of a mobile application, it is possible to use the following form elements:

- Fields of the following types:
  - Entry field;
  - Label field;
  - Switch field;
  - Picture field;
  - Check box field;
  - Indicator field;
  - Slider field;

    - Chart field;
    - Field of spreadsheet document.
- Button;
- Table;
- Decoration: inscription and picture;
- Groups of all kinds.

29.3.4.3. Placement of form elements

Groups, including the form itself, and most of the form elements, are displayed in the form of lists, as is customary in mobile interfaces. Form elements are conventionally divided into three types:

1. Necessarily are located in the line of the list. This group includes the following elements:
    - Entry field;
    - Label field;
    - Switch field;
    - Button;
    - Chart field.
2. Can be located in the row of the list. The group includes decoration.
3. Cannot be placed in the row of the list. This group includes all other elements.

The phrase "display list" means the following:

- The group has a background that is different from the color of the form.
- A row list is either a form element or a group with horizontal grouping of elements.
- Elements located in a group with horizontal grouping are arranged in one line.
- Elements located in a group with a vertical grouping are located each on its own line.
- If there are elements in the group with vertical grouping that cannot be located in the row of the list, a gap is displayed in the list.
- If the group has a display title, then an indent is generated before the group.
- Visual line separators are horizontal lines, the display of which cannot be controlled.

The peculiarity of the behavior of type 2 elements is that they are located in a row of the list only if they are grouped with type 1 elements. Elements of type 2 themselves are not listed in the list.

Many form elements do not have borders (input fields and buttons whose border color is set to Auto). In this regard, the alignment of the input fields on the form (as on the platform for a personal computer) has no practical

meaning. Instead, the algorithm uses the alignment of values in the input fields. Depending on the presence and location of the heading of the input field, the information in it is aligned in different ways:

- The field header is missing or is not displayed on the left, the numerical values are aligned to the right, and the rest - to the left.
- The field header is on the left:
  - In the fields with text editing, - the text is aligned to the left.
  - Numeric values are aligned to the right.
  - The alignment of fields of other types depends on whether there is a text editing field in the group with these fields or not. If there is such a field, the alignment of the other fields is carried out in the same way as the alignment of this field. Otherwise - to the right.

The title of the form is displayed only in the window title and is not duplicated on the form itself.

As mentioned above, the horizontal scrolling of the form does not apply on mobile devices. At the same time, vertical scrolling of a form is quite acceptable. Based on these prerequisites, the mobile version of "1C:Enterprise" tries to automatically rebuild the form so that it fits on the screen of the mobile device. In the process of adaptation, a width of the form items is reduced for those items whose width in the configuration is set so that it exceeds the actual screen width on a mobile device. The width for such elements is generated in such a way that the element fits (in width) in the current orientation of the mobile device without horizontal scrolling. In addition to changing width, the system can attempt to reduce a number of the displayed form items in order to improve the visual appearance of a form. To control the need for such a restructuring, the property of the The option of folding form elements in importance (`CollapseItemsByImportance`) form is intended. The `Auto` value is treated as `Use`.

To make the mobile version easier to adapt the form, the form elements (and columns of tables) have the Importance when displaying (`DisplayImportance`) property. By changing a value of this property, a developer can achieve the desired kind of form on the device screen. A value of this property is handled by the mobile client as follows:

- More important items are given more space on a form.
- If the more important item is located below the less important ones and these less important items occupy more than three rows of a form, then the less important items are combined and placed in a collapsible group. It is important to understand that here and further a height is important but not a number of items. In other words, the three rows of the form grid can occupy both three items with a height of one row each as well as one form item occupying three rows of height.

- If the first or last of the collapsible items is the command bar, then it does not collapse as it can refer to an important item.
- If the more important item is located within the hierarchy of groups or pages and at the same time in different places of the hierarchy, the less important items are located above it occupying more than three rows, a collapsible group is formed in each level of hierarchy.
- If there are several elements of higher importance, then less important elements between them are placed in a collapsible group, if less important elements occupy more than three lines.
- Less important items that are below the last more important item are not collapsed.
- If the items of different importance are also located inside a collapsed group, the algorithm is applied recursively to this group.
- The heading of a coagulated group is formed according to the following algorithm:
  - If an already existing group is being collapsed, then the heading of this group is used.
  - If a single item with a specified heading falls into a collapsible group, then the heading of this item is used.
  - If none of the above items is appropriate - Heading is formed by combining through "," all the headings of the items included in the collapsible group.

If for the form item the `DisplayImportance` property is set to `Auto`, then the actual value of the property will be determined by the following rules:

- Value `VeryHigh` for the form item displaying data from the main form attribute (and the subordinate atributes).
- A value `High` if the item is stretched vertically and is one of the following items:
  - A form table displaying a dynamic list;
  - A field of a tabular document;
  - A field of chart of any kind and type;
  - formatted document field;
  - An HTML document field.
- If an item is a command bar, then:
  - A value is `High` if the source of commands for the command bar is the controlled form itself.
  - The value of importance of an item that is the source of commands for the command bar.

    - The importance value of the most important item in the same group as the command bar provided that no command source is set for the command bar.
- The value `Standard` for all the remaining items.

29.3.4.4. Features of displaying form elements

29.3.4.4.1. General peculiarities

The form does not display an empty mark for both the elements located on the form and the elements located in the table.
Field headers are displayed without the ":" characters at the end of the text.

29.3.4.4.2. Forms

The `CurrentItem` property allows to get and set the current form element. However, setting the current form element does not lead to the transition of this element to the editing mode and is not displayed visually. To put a form element in edit mode, use the `BeginEditingItem()` method.
If at the opening of the form any actions are performed, then the fact that until the completion of all the form handlers, the screen of the mobile device will not be updated, should be considered.

29.3.4.4.3. Input fields

To facilitate work with the input field using touch screens of mobile devices, in all fields, except for text and text input fields, where auto complete is used, editing the value directly in the input field is not supposed. The input field behaves like a button, the title of which displays the representation of the current value, and pressing this button opens a specialized editor. In case an auto complete is used in the input field, it is possible to clear the value using the Clear button located in the left part of the window title.
A long press on the input field leads to the appearance of a context menu, which consists of standard system commands and commands that are added to this menu during development.
To display the clear, open and adjust buttons, set the corresponding property of the input field in the Yes value. In addition, there is an additional opportunity to control the visibility of the cleaning and opening buttons, which are displayed directly in the input field. This feature is provided through the `AutoShowClearButton` and `AutoShowOpenButton` properties. These properties are processed only if the corresponding property of the input field is set to Yes. The following options of behavior are possible (the values

of the AutoShowClearButton and AutoShowOpenButton properties):

- Only for completed - in this case, the button will be displayed only if the value in the field differs from the default value for the type used or if the data displayed by the input field is of a composite type.
- Always - clear button is always displayed.
- Auto - behavior depends on the property:
  - The property AutoShowClearButton - is similar to the value Only for completed;
  - The property AutoShowOpenButton - is similar to the value Always.

The selection button is always not displayed, at this, the StartChoice event is triggered when clicking on the input field itself. If a separate form is used to edit the field value, then a special marker (>) is placed next to the field, clicking on which will open this form. The location and appearance of editors is different on the tablet and phone.

The multi-line input field always works in text input mode. This input field does not support value selection and auto complete. If the input field is associated with data other than the Line type, then the value of the MultiLine check box is ignored for such an input field. A multiline input field can be displayed in three different ways:

1. Automatic adjustment of the height of the field to the height of the text (to avoid scrolling). Adjustment occurs when the field occupies the entire width of the form, the field height is not specified (the Height property), and the field can stretch vertically (the VerticalStretch property).
2. The field has a specified height and vertical scrolling inside the field is allowed. The field is displayed in this way in the case when the field height is specified (the Height property) and the field cannot stretch vertically (the VerticalStretch property).
3. The field occupies the maximum possible place on the form. A field is displayed in this way when the field height is specified (the Height property) and the field can stretch vertically (the VerticalStretch property).

The value of the Input hint property is not displayed in the multiline input field.

For the mobile platform, additional properties of the input field are also relevant, which will allow more efficient use of the application. These properties allow to further customize the input field (only if the input field displays data in the type of Line) for implementation of some specific tasks:

- The `AutoCorrectionOnTextInput` - property allows to enable automatic correction of the entered text. Behavior depends on the settings of the virtual keyboard used. The value Auto means that automatic correction works for multiline input fields, but not for normal (single-line) - ones.
- The `SpellCheckingOnTextInput`- property allows to enable spell checking when entering text. At operation under the Android OS, this property is ignored, since this feature is controlled by the system and depends on other settings.
  At operation under Windows OS, this property and the `AutoCapitalizationOnTextInput` property are always enabled at the same time. When setting the property `SpellCheckingOnTextInput` into the Use value, the operating system will automatically turn on the register changing at the beginning of the sentence when entering text. This change will be made regardless of the value of the `AutoCapitalizationOnTextInput` property. The value of the `AutoCapitalizationOnTextInput` property will be used when running under other operating systems.
  The Auto value means that the automatic check works for multiline input fields, but not for normal (single-line) - fields.
- The `AutoCapitalizationOnTextInput` - property allows to manage the automatic change of the register of characters when entering text. The behavior depends on the operating system used and the settings of the virtual keyboard used.
  At operation under Windows OS, this property and the `AutoCapitalizationOnTextInput` property are always enabled at the same time. When setting the property `SpellCheckingOnTextInput` into the Use value, the operating system will automatically turn on the register changing at the beginning of the sentence when entering text. This change will be made regardless of the value of the `AutoCapitalizationOnTextInput` property. The value of the `AutoCapitalizationOnTextInput` property will be used when running under other operating systems.
  Can take values:
  - No - automatic changing of the register is not performed.
  - Words - the first letter of each word is automatically capitalized.
  - Sentences - the first letter of each sentence is automatically capitalized.
  - All characters - all entering characters are automatically capitalized.

- Auto - for multiline input fields is similar to the Sentences value, and for ordinary (single line) - No.

- The `SpecialTextInputMode` - property allows to specify a special type of virtual keyboard when typing in the input field:
  - No - the virtual keyboard is used by default.
  - Digits and punctuation - the virtual keyboard to enter digits and punctuation is used.
  - URL - the virtual keyboard to enter URL is used.
  - Email - the virtual keyboard to enter an email address is used. There will be a symbol @ on the keyboard.
  - Phone number - the virtual keyboard to enter a phone number that contains numbers and keys * and # is used.
  - Digits - the on-screen keyboard to enter digits is used.
  - Auto - is used similar to the No value.

  When operated on tablets running Android OS, all types of virtual keyboards, except for the keyboard of the Phone number type, can look the same, due to the large amount of free space on the screen.

When developing forms that involve different types of information (phone numbers, email addresses, etc.), there may be a need to name, in a special way, the enter complete button on the virtual keyboard used. This should help the user understand what action will be performed after pressing this button. To do this, there is a property of the input field `OnScreenKeyboardReturnKeyText`. The value of this property indicates the text that will be displayed on the button of the input completion. When operated under the Android OS, when specifying the value Continue, Next will be displayed, and when specifying the value Connect - Enter. On Windows, this property will be ignored. Text management on input buttons is not supported.

29.3.4.4.4. Check box field

A check box can only display two states. When running on OS iOS, the check box is always displayed as a tumbler.

29.3.4.4.5. Switch field

View switch field Switch is always displayed in a separate vertical group (the number of columns is not adjustable). View switch field Switch cannot be combined with other elements on the same line.

View switch field Tumbler is always displayed in one line (the number of columns is not adjustable).

29.3.4.4.6. Button

In the case when the button is located in a vertical group, it stretches to the width of the parent group. The text inside the button is aligned to the left edge of the button, except for the case when the color of the button border is different from the default - in this case, the text is aligned to the center of the button.

29.3.4.4.7. Group

When displaying groups of a managed form, only indenting for groups with the Normal selection and Strong selection display modes is supported.
When displaying a collapsible group (the Behavior group property is set to Collapsible value), there are a number of features:

- The header of the collapsible group is displayed as a clickable line;
- The collapsible group is always displayed with the Picture display variant and its (display option) setting is ignored.

The group of the Pages view allows to implement several options for the tabbed panel. The view of the panel (and its behavior) depends on the value of the property Display Tabs:

- Tabs on the top.
  - Tabs are displayed as a tumbler. If the header text does not fit in the space provided, then it is shortened so that the ... character is displayed at the end of the text. Pictures in the headers are not displayed.
  - Pages are located inside the group under the list of tabs.
  - The active tab is displayed by the header inversion.
- Tabs from the bottom.
  - Tabs are displayed as a panel of buttons. If the header text does not fit in the space provided, then it is shortened so that the ... character is displayed at the end of the text. The pictures in the headings are displayed, while the heading text is always displayed under the picture.
  - Pages are located within the group above the list of tabs.
  - If there is not enough space on the screen to display the headers of all the pages of the group, then the More button is added to the end of the list. Clicking this button opens a list of headings for all pages of the group.
- Tabs from the left.
  - Tabs are displayed as a list. Clicking on the tab opens a new window with the contents of the tab, which replaces the current form (with a

list of tabs). To return to the list of tabs, use a special button (<) located in the upper left part of the page header.

- Scrolling.
  - Tabs are displayed as points below the pages. Text and pictures are not displayed. Switching between pages is done by a left-to-right or right-to-left touch in the pages area.
  - The pages are located inside the group above the points describing the number of tabs.
  - In the personal computer platform, this value is interpreted as the Tabs on the left value .
- Tabs on the right.
  - Do not supported by mobile platform.

29.3.4.4.8. Tables

Operation with the current line

The behavior of the current row in the table is determined by the table property Using the current row (`CurrentRowUse`):

- Value `Selection`. In this case, properties `CurrentRow`, `CurrentItem`, `CurrentData` are determined only during implementation:
  - context command handler;
  - event of activation of the row or cell;
  - row edit events.

  The rest of the time, these table properties are set to `Undefined` value. For a form command that requires data from the current row, it is necessary to specify explicitly the need for the form command property Use the current row. In addition, it is necessary to specify the table from which the command will be used by using the Used table property.

  In the list, the current row is visually determined briefly, while clicking on the row.

  If the current row is set from the 1C:Enterprise language, the set value is retained until the top-level 1C:Enterprise language method (called by the system from the interface, not from another 1C:Enterprise language method) is completed.

  This value is recommended for tables that are not logically related to any other data (including tables).
- `SelectionPresentation` value . In this case, the current row always exists if the table contains at least one row.

In the list, the current row is always visually determined.
This value is recommended for tables that are logically related to some other data (including tables).

- `SelectionPresentationAndChoice` value. In this case, the current row always exists if the table contains at least one row.
  In the list, the current row is always visually determined. Also on the right side of the row there is a button that performs two actions simultaneously: activating of the row and its selection.
- `Auto` value. This value is treated as a `Choice` value on both the mobile platform and the mobile client.

The `CurrentParent` property is defined in the following cases:

- in hierarchical lists - always;
- in the tree view, - only in cases where the `CurrentRow` property is defined.

Displaying table columns

The table in the mobile platform never has horizontal scrolling. In this regard:

- Column fixing in the table is ignored.
- The value of the `HorizontalScrollBar` property is ignored.
- The display of columns that do not fit in the visible part of the table depends on the value of the managed form table property Behavior at horizontal compression (`BehaviorOnHorizontalCompression`) and from the Importance when displaying property (`DisplayImportance`) of columns of this table:
  - The value Hide elements by importance - in this case, columns that do not fit on the form will be hidden as the property value Importance when displaying of the column decreases. Thus, the columns with the lowest importance (Very low) will be hidden first, then with the next importance (Low) and etc. If there are several columns with the same importance, the hiding is done from right to left. In other words, the more to the left the column is located in the table, the higher are its chances to be displayed on the form (other things being equal).
  - The value of Transfer elements by importance - in this case, the columns are hidden as the importance decreases (similar to the behavior of Hide elements by importance), and the values of hidden columns are displayed on a separate row, in small print, comma-separated and color-interleaved.
  - The value of Auto depends on where the configuration is used:

- on mobile platform: is treated as Hiding elements by importance.
- on mobile client: is treated as Transfer elements by importance.

Various modes of the table

The table in the mobile application can be displayed in three modes:

- main mode;
- multiple row selection mode;
- row sort mode.

The main mode is similar to the mode of displaying the table on the platform for a personal computer, with the exception of displaying the current and selected rows. In the mobile platform, these rows are not displayed. The exception is the short-term highlighting of the current row of the table when the form is opened. The form table automatically adjusts to the size of the content if the following conditions are met simultaneously:

- the table covers the entire width of the form;
- docked table height is not set;
- the table is stretched vertically;
- the object that displays the table contains less than 100 rows of data. If there are more than 100 rows -, scrolling will be available.

The multiline row selection mode is characterized by a special column with check boxes, with the help of which the selection is made. The column is always displayed first. In this mode, contextual table commands are available and act on the entire selection at once.

The row sorting mode is characterized by the presence of a special column with a marker for which it is possible to drag a row. This column is always displayed last. In this mode, the table context commands are not available.

Multiple selection mode in the table (the `SelectionMode` property is set to `Multiple` value) on the mobile platform is not visually displayed. A flag of its existence is a special command Select multiple, which is available in the command bar associated with the table. When this command is selected, the table switches to the multi-row selection mode.

If the multiple selection mode is set (the `Multiselect` property is set to `True` value), then the table is immediately in the multi-row selection mode, while there is no command to switch to this mode in the command bar.

If the table is available to change the order of rows (for example, the document tabular section), then to activate this mode, it is necessary to perform a special command Permutation of rows, which is available in the command

bar associated with the table. In this case, the table goes into the row sorting mode, and the selected rows are reset (if any).
For the table, only two row entry modes are supported (the RowInputMode property): at the end of the list and at the end of the window. The row selection mode is always used - Row (the RowSelectionMode property). The actual value of the property is ignored.

Context menu

The form table supports the use of the context menu, which opens with a touch from the right border of the table or by long pressing the row.
Filling the command bar and context menu has important differences from the platform for the personal computer. On a platform for a personal computer, the question of commands distribution between the command bar and the context menu - is a question of form design. On the mobile platform (due to the absence of the current row) it is a question of the functional model. If the command uses the properties of the table associated with the current row, then this command should be located in the context menu. If such a command is located by the developer in another place, then the system can forcibly move it to the context menu. The exception to this rule is the case when a command can do its operation both with and without the current row. As an example, a command Create of the dynamic list, displaying the data of a hierarchical directory can be given. Off-text use of the command involves creating an element in the root directory, and contextual - in the current group.
In order to indicate to the system that the command is operated with the data of the current row, the property of the command of the form CurrentRowUse is intended. In order to associate a form command with a table, whose data it must process, the property of the command AssociatedTable is used. Using this property means that it is impossible to implement a command on the mobile platform that operates simultaneously with data from more than one table. To control the placement of a command, there is a property of the command Display In The Context Menu:

- Auto - the command that uses the data of the current row is always located in the context menu. A command that does not use the data of the current row - is always located in the command bar of the table.
- Only In The Context Menu - the command is always located in the context menu.
- Advanced In Context Menu - the command is located in the context menu and in the command bar of the table.
- No - the command is located only in the command bar of the table.

Search string

Managing the location of the search line is also different from the platform for the personal computer. Depending on the value of the table property SearchStringLocation, the following behavior is possible:

- Auto - the search line is available by pressing the button on the command bar (if the table displays a dynamic list and this dynamic list is the main attribute of the form) or appears when touching down the table header (in other cases);
- Top, Command bar - the search line is located between the table command bar and the table header;
- Drag from the top - the search line appears when touching down the table header;
- Form header - the search line is located in the form header;
- Bottom - no search line;
- No - no search line.

If two or more form tables have the SearchStringLocation property set to Form header value, then the actual behavior will be similar to if each of these tables has this property set to Top.

Height of the table on the form

When scrolling the table associated with a dynamic list, translucent buttons appear in its upper and lower parts. Clicking on these buttons leads to the beginning or end of the list (respectively).

To control the height of the table of the managed form, use the Option of height control property (HeightControlVariant):

- In form rows - in this case, the table height is set using the Height property of the managed form table. If the value of this property is equal to 0, then the height of the table will be equal to 8 units for a table displaying a dynamic list and 4 units - in all other cases.
- In the table rows - in this case, the height of the table is set using the property HeightInTableRows of the table of the managed form. If the value of this property is equal to 0, then the height of the table will be 7 rows for a table displaying a dynamic list and 3 rows - in all other cases.
- By content - if the table is linked to a dynamic list, then the table height is set by the property value HeightInTableRows. If the value of this property is equal to 0, then the height of the table is determined by the value of the HeightInTableRows property.

If the table is not associated with a dynamic list, then all rows of the table are fully displayed, but not less than 3 and not more than 100 rows. When adding the 101st row, the table is compressed to the height specified in the `HeightInTableRows` property. If this property is set to 0, then the `Height` property is used to determine the height. At the same time, the contents of the table begin to scroll vertically. When deleting the 101st row, the table is stretched to a height, enough to accommodate 100 rows of the table.

Value Based on content is recommended to be used for tables displaying a small number of rows.

- Auto - is similar to the value Based on content.

Data update

Updating the data displayed by the table can be done in the style adopted on mobile devices: by trying to pull the list down or up when the first or last (respectively) line of displayed data is reached (pull to refresh). To control this feature, use the table property Refresh request (`RefreshRequest`):

- No - update can be performed only through the command of the command bar of the table;
- Any other value means that the update can be initiated by trying to pull a list in the appropriate direction. At the same time in the table area outside the data animation is displayed.

When updating the list, a table event handler will be called `RefreshRequestProcessing`. In that case, if the table is adapted to the content, then the Handler for the Table `RefreshRequestProcessing` will not be called.

If the dynamic list associated with the table does not contain displayed data, then No data to display is displayed inside the table. The table that is not associated with the dynamic list does not display any text when there is no data to display.

Display rows of tables

Instead of editing data in the row of the table associated with the tabular section or the table of values, the mobile platform provides a special system form. This form contains all columns of the current table row displayed on the parent form. When using this form, be aware of the following features:

- The columns for which the View only property is set are displayed in the system form also without the possibility of editing;

- In the system form, the fields of type picture and inscription are not displayed;
- The system form is subject to conditional appearance of the form;
- All event handlers are called;
- The system form is not editable in the configurator and cannot be modified from the 1C:Enterprise language.

After returning from edit mode, the edited line is briefly highlighted.

29.3.4.5. Dynamic list

The dynamic list used in the mobile platform has the following features:

- Management of dynamic list settings is supported using the 1C:Enterprise language. For interactive customization, it is necessary to implement independently the appropriate configuration forms.
- Interactive list display is not supported;
- It is possible to sort by clicking on the title. It is not possible to sort by several columns in this way.

29.3.4.6. Form panels

The form navigation panel is implemented using a special menu that is displayed at the bottom of the form as a switch. Pressing the switch button opens the corresponding form.

Command bar form:

- In the top display mode:
  - The command bar is located in the form header, to the right of the text.
  - If the command bar contains standard command pairs OK-Cancel and Yes-No, then the negative action button is transferred to the left side of the form header, and the positive one is displayed in the command bar of the form. In the More menu there are all commands except the command with negative action, which was transferred to the form header.
  - If the command bar does not contain the above pairs of commands, but contains the standard form command Cancel, No and Close, then the button with such a command is transferred to the panel in the header on the left.
  - The command, which is located on the left side of the form header, will be displayed with a special symbol (<), but will perform its original function.

- When a button is displayed in the command bar of the form, the property Display of this button is considered. The command properties associated with this button are ignored when defining the display mode of the button.
- If the command bar contains a default button, then it is always displayed in the header first.
- If there is enough free space on the right side of the header to display the remaining commands, - they are displayed there. Otherwise, the More button is displayed and the system tries to make sure that the More button will not be the only one: if the command displayed on the right side of the header is not yet defined, then it will be the first command from the list of the remaining commands, and the rest will be located in the More menu.

- In the display mode below:
  - Docked when scrolling form.
  - When the keyboard appears, is not displayed.

The list of standard commands does not include commands for direct deletion (for objects of a reference type), calling for help, displaying a list, changing a form. Standard commands Record and close and Post and close are displayed in the command bars with the title Done.
For applications running on OS iOS, it must be remembered that the system mechanisms for closing a form do not ask the user whether to save data when the form with modified data is closed.
The command bar associated with the table is located above or below the table and does not exceed the width of the table itself. When scrolling the table, the command bar always remains in the visible area.

29.3.5. Special features of the mobile device

29.3.5.1. Operation with telephony

Mobile version of "1C:Enterprise" provides a set of features that allow to make calls from mobile applications. Calls can be made with or without user confirmation. The system also provides access to the call log and the ability to handle events that occur when receiving or making phone calls. The application can determine the capabilities for operation with telephony. The methods `NumberDialingSupported()`, `CallLogSupported()` and `CallsHandlingSupported()` of the object `TelephonyTools` are intended for this. Call processing is possible only on Android OS.
Dialing is done as follows:

```
If TelephonyTools. NumberDialingSupported() Then
  TelephonyTools. DialNumber(PhoneNumber, DirectCall);
EndIf;
```

The `DirectCall` parameter is responsible for how the subscriber's call will be made:

- The parameter is equal to the value `True` - in this case, when the method is executed, the number starts to be dialed automatically. A standard call program interface appears on the screen.
- The parameter is equal to the value `False` - in this case, when the method is executed, the call handling application will be opened, in which the specified number will be set. The user will only press the start call button.

Mobile devices operating under the control of the Android OS, provide access to the call log. It is possible to iterate over the log records with the installation of the necessary filter. To operate with the log, it is necessary to use the method `GetCallLog()` of the object `TelephonyTools`.

The following example demonstrates getting a list of all missed calls on a mobile device.

```
If TelephonyTools.CallLogSupported() Then
  Log = TelephonyTools.GetCallLog();
  Filter = New DataCompositionFilter;
  NewItem = Filter. Elements.
Add(Type("DataCompositionFilterItem"));
  NewItem.LeftValue = New DataCompositionField("CallType");
  NewItem.ComparisonType = DataCompositionComparisonType.Equal;
  NewItem.RightValue = CallLogCallType.Missed;
  NewItem. Use = True;
  Log Records = Log.FindRecords(Filter);
  For Each Record From LogRecords Do
    // call list processing
  EndDo;
EndIf;
```

At handling with the call log, it should be remembered that the filter of the log is performed using the `DataCompositionFilter` object.

To process call events (on devices running on Android OS), it is possible to connect/disconnect a call handler. After the call event handler is registered in the system, upon the occurrence of an event, the registered handler is called to which the call information is passed. The handler does not allow to manage the actual call, for this purpose, the standard operating system program is used.

Using this mechanism, it is possible to implement, for example, mobile application own call log or implement a reminder of a missed call.

```
&OnClient
Procedure for Managing Call Handlers (Connect Handler)
  If NO TelephonyTools.CallsHandlingSupported() Then
    // call processing is not supported
    Return;
  EndIf;
  Handler = New NotifyDescription("CallsProcessing", ThisObject,
"Parameters");
  If ProcessCallEvents Then
    TelephonyTools. AttachCallsHandler(Handler);
  Else
    TelephonyTools.DetachCallsHandler();
  EndIf;
EndProcedure

Calls Handler Procedure (PhoneNumber, Date, Event Option, CallType,
Additional Parameters) Export
  // calls events processing
EndProcedure
```

29.3.5.2. Operation with messages (SMS and MMS)

Mobile version of "1C:Enterprise" provides a set of tools to operate with messages (SMS and MMS). Provides the ability to send and receive messages. Access to the messages list is determined by the mobile operating system used. Sending messages is possible both interactively and in fully program mode (non-interactive mode).

Mobile version of "1C:Enterprise" provides the means to determine the capabilities of the used mobile OS. For this purpose, the global context object `TelephonyTools` has methods `SMSSendingSupported()`, `SMSReceivingSupported()` and `MMSSendingSupported()`. Depending on the mobile operating system used, the following restrictions apply on operation with messages:

- For OS iOS:
  - Subscription to receive SMS/MMS is unavailable;
  - Non-interactive sending of SMS/MMS is unavailable.
- For OS Android:
  - Non-interactive sending of MMS is unavailable.
- On Windows OS:
  - Subscription to receive SMS/MMS is unavailable;
  - Non-interactive sending of SMS/MMS is unavailable.

The message in “1C:Enterprise” is represented by the `SMSMessage` object. Sending a message is as follows:

```
&OnClient
Procedure Send SMS (To, Text, Send Interactively)
  SMS = New SMSMessage();
  SMS. Text = Text;
  Line Recipients = StrReplace(To, ",", Symbols.LF);
  For Counter=1 To StrLineCount(RowRecipients) Do
    SMS. Recipients. Add (TrimAll(StrGetLine(Line Recipients,
Counter)));
  EndDo;
  TelephonyTools.SendSMS(SMS, SendInteractively);
EndProcedure
```

As a value of the `To` parameter, several phone numbers may be passed, separated by the symbol «,». The `Send Interactively` parameter determines how the message will be sent:

- The parameter is set to `True` value, the - message will be sent immediately;
- The parameter is set to `False` value, - will open the messaging application, which is specified as the default application for the mobile OS being used. In this application, the form for sending a message with filled fields will be opened. In this case, the user must independently press the button for sending a message in an open form.

In order for an SMS message to be converted into an MMS message, it is necessary to add attachments to messages. This is done using the `SMSMessage. Attachments` property. In this case, the MIME type of the passed data should be specified as the value of the `EncType` property of the `MMSAttachment` object, for example, for a jpeg file this will be image/jpeg.

If the mobile OS allows, the mobile version of "1C:Enterprise" allows to subscribe to a notification about receiving SMS messages.

```
&OnClient
Procedure Attach the Message Receive Handler()
  MessagesRecipient = New NotifyDescription("MessageReceiving",
ThisObject);
  TelephonyTools.AttachSMSMessageHandler(MessagesRecipient);
EndProcedure

&OnClient
Procedure MessageReceiving(Message, AdditionalParameters) Export
  Inform(Messages.Sender + " - " + Message.Text);
EndProcedure
```

When an incoming message is received, a notification handler will be called to process the received message. Messages received using the installed notification handler will also be available from the system messaging application.

To check the accessibility of the message log using the `TelephonyTools.SMSLogSupported()` method. If the mobile OS being used provides access to the message log, then it becomes possible to receive messages in accordance with the imposed filter.
The following example demonstrates getting a list of all unread messages on a mobile device.

```
If TelephonyTools.SMSLogSupported() Then
  Log = TelephonyTools.GetSMSLog();
  Filter = New DataCompositionFilter;
  NewItem = Filter. Elements.
Add(Type("DataCompositionFilterItem"));
  NewItem.LeftValue = New DataCompositionField("Read");
  NewItem.ComparisonType = DataCompositionComparisonType.Equal;
  NewItem.RightValue = False;
  NewItem. Use = True;
  Log Records = Log.FindRecords(Filter);
  For Each Record From LogRecords Do
    // messages list processing
  EndDo;
EndIf;
```

At operation with the SMS log, it should be remembered that the filter of the log is performed using the `DataCompositionFilter` object.
When filter from the `PhoneNumbers` field, some features should be considered:

- It is permissible to specify fragments of numbers only if the string is used as the right value of the filter element, and the type of comparison should be `DataCompositionComparisonType.Contain` or `DataCompositionComparisonType.NotContains`. Spaces, brackets and hyphens in the number are not ignored.
- Indication of names from the address book (list of contacts) is not supported.
- If an array of numbers is specified as the filter value, the message is considered to satisfy the condition only if the same number of numbers and all numbers coincide with each other in the message and filter.

As a result of search, a list of messages is generated, ordered in such a way that the newest messages are at the top of the list.

29.3.5.3. Location tools

29.3.5.3.1. General information

A large number of mobile devices supports the determination of the coordinates along which the device is located. It also provides the ability to convert

the received coordinates into the address and vice versa. For access to these functions, **provider** of location is intended. Each provider is described by a name and a specific set of parameters (features). A specific location provider may not support all features.

Overall scheme of operation with tools of location is as follows:

1. The required provider is selected. In each case, it is necessary to choose the provider that is most suitable for the implementation of the task assigned. In general, it is worth choosing a provider that provides the minimum power consumption and maximum accuracy in determining the coordinates.

Example:

```
// Define the provider that does not lead to spending
// funds and uses to determine the coordinates
// cellular network
RequiredProvider = Undefined;
Providers = LocationTools.GetProviders();
For Each Provider From Providers Do
  If NO Provider. Paid and Provider. UsesCellNetwork Then
    RequiredProvider = Provider;
    Break;
  EndIf;
EndDo;
If RequiredProvider = Undefined Then
  Inform("Required provider not found");
Else
  Inform("Found provider: " + RequiredProvider.Name);
EndIf;
```

   If necessary to get the most energy-efficient provider or the most accurate provider, it is possible to use the methods of `GetMostPowerSavingProvider()` or `GetMostAccurateProvider()` (respectively).

2. The coordinates are determined using the selected location provider. Two methods are used to obtain the coordinates. The `GetLastLocation()` method returns the latest location data received by the selected provider. Among the properties of the `LocationData` object (which returns the `GetLastLocation()` method) there is the `Date` property, which describes when the location data was received. If this data was received very long ago, then it is necessary to call the `UpdateLocation()` method in order for the location provider to redefine the location data. One of the parameters of the method determines the timeout for determining the location. In case the method is completed on time-out, the actual determination of the

coordinates in this case depends on the specifics of the implementation of the particular version of the mobile operating system.

Example:

```
// If the location on this device has never been performed
// or completed more than an hour ago - update the location and
// receive specific coordinates
Coordinates = LocationTools.GetLastLocation(ProviderName);
If Coordinates = Undefined Or CurrentDate()-
ToLocalTime(Coordinates.Date) > 3600 Then
  LocationTools.UpdateLocation(ProviderName, 60);
  Coordinates = LocationTools.GetLastLocation(ProviderName);
EndIf;
```

3. If necessary, a subscription to change the coordinates is performed. After the need for a subscription has disappeared, - it is possible to turn off the notification of changes in coordinates.

After receiving the location data, it is possible to perform several auxiliary operations related to the received location:

- Get the address by coordinates. To do this, the `GetAddressByLocation()` method is intended.
- Get location coordinates by address. To do this, the `GetLocationByAddress()` method is intended.
- Show a map showing the coordinates of one or more points. To do this, the `ShowOnMap()` method is intended. This feature is not available on all mobile OS. The `MapRepresentationSupported()` method allows to determine whether one or more points are displayed on the map. To display points on the map in each mobile OS, a branded application of developer of the operating system will be used.

NOTE 1. For using methods of converting coordinates to address and vice versa, the mobile device must have access to the Internet.
NOTE 2. The operation of the location mechanism in the background is not supported on devices running on OS iOS.

29.3.5.3.2. Getting the key for Google Maps

IMPORTANT! Description of the developer console is as of the writing of the documentation.

In order for the mobile application to deal with location under the Android OS, it is necessary to get a special key for dealing with Google Maps. To do this, follow these steps:

1. It is necessary to enter the developer console (Google Developers Console, https://console.developers.google.com/project).
2. It is necessary to create a project (Create project button).
3. It is necessary to enable the use of the Google Maps service. To do this, select the application, and then use the menu command Products and Services - API Manager (menu is available in the upper left corner of the page, next to the Google APIs label). On the Library page it is necessary to **enter** in the input field Google Maps Android API. In the list, below the input field, select the Google Maps Android API item, and on the page that opens, click the Enable hyperlink (at the top of the page, to the right of the Google Maps Android API label).
4. It is necessary to obtain a software interface key (API Key). To do this, select the Accounting data menu item on the left side of the page. In section Accounting Data it is necessary to press the Create accounting data button and select API Key in the drop-down menu. After creating the key, click the Close button in the resulting dialog box.
5. It is necessary to create a mobile application group in the mobile application builder (after setting up the mobile application builder setting) and copy the contents of the Key acquisition parameter for operation with Google maps field.
6. In the Key restriction switch, select Android application. Then click the Add resource button: package name and checksum. The copied data must be inserted into the SHA-1 certificate checksum field. Then, at the end of the inserted line, select the full application ID and transfer it to the Package name field. At this, the checksum should not end with a symbol ";".
7. After the formation of the key is completed, it will be displayed in the API Key field. The resulting value must be entered in the Key for operating with Google maps field, which is in the form of a mobile application group. Then necessary to save the mobile application group.

29.3.5.4. Operation with multimedia

29.3.5.4.1. Record

Mobile devices have various multimedia features: take a picture, make an audio - or video recording. For various mobile devices, these features may or may not be available. To determine the possibility of performing a particular operation, special functions are provided: `PhotoSupported()`,

VideoRecordingSupported(),
AudioRecordingSupported().
If the mobile device supports this or that feature, it is possible to use the global context method, which activates the necessary system application to perform the desired action: MakePhoto(), MakeVideorecording(), MakeAudiorecording(). For mobile applications on any mobile operating system, the sound is recorded in stereo, the bitrate is 128 kbps, and the sample rate is 44 kHz. MultimediaTools.MakeAudioRecording() method returns in MultimediaData object where the file extension is set to m4a and the MIME type - is set to audio/mp4.
Running the program in the 1C:Enterprise language will wait for the requested action.
Due to the fact that mobile devices may have two cameras (front and rear), the platform provides the mobile application with the opportunity to operate with cameras by choice. To do this, use the DeviceCameraType system enumeration when determining the features of a mobile device and when performing an action.

```
Front = MultimediaTools.PhotoSupported(DeviceCameraType.Front);
Rear = MultimediaTools.PhotoSupported(DeviceCameraType.Rear);
```

The above program code allows to determine which of the two cameras of the device supports photography. Using these methods, it is possible to determine the presence of a particular camera on the device. When performing a photo - or video recording, it is also possible to specify the camera used. If the camera type is specified as DeviceCameraType.Auto, then it will be processed on devices in the following way:

- The device has two cameras - the rear camera will be used;
- The device has one camera - this camera will be used (depending on availability).

In addition, when trying to take a picture or take a video, there is a possibility to specify some additional parameters that affect the result of the action.
When trying to take a picture, it is possible (except for the camera type) to additionally indicate:

- What resolution is required to shoot. If a resolution that is not supported by the device is specified, the nearest appropriate resolution will be selected. The list of supported resolutions can be obtained using the GetSupportedCameraResolutions() method of the Cameras () of the MultimediaTools object. For iOS devices, this parameter is ignored.
- What is the quality of the resulting snapshot. Actually determines the degree of compression of jpeg file. 100 - means maximum quality.

- Which snapshot should be taken as a result: color or black and white. For devices running on iOS, an indication of the need to get a black and white snapshot does not affect the camera interface (but the snapshot itself will be black and white).
- The initial mode of usage of the flash when opening the camera interface. To do this, use the `LightingType` parameter of the `MakePhoto()` method. It is possible to forcibly turn on, turn off the flash or let the device determine the flash mode automatically.
- `Mark` parameter allows managing the insertion of the date and time of the photo-taking and free text in the right bottom corner of the photo. `PhotoStamp` object shall be transferred as the parameter value for it.

In order to speed up the acquisition of snapshot data when running on Android OS, it is recommended to perform the following actions:

- In the event that the application does not require a front camera, force the rear camera as the main camera (parameter `CameraType` of the method `MultimediaTools.MakePhoto()`). If automatic camera detection mode is set - it is not recommended to switch to the front camera of the device in the camera interface.
- When photographing, it is advisable to keep the device horizontal (wide side down) and the camera on the left.
- Do not enable photograph conversion to monochrome (parameter `Monochrome`of the method `MultimediaTools.MakePhoto()`) without obvious need.
- If possible, reduce the resolution and quality of the resulting photo to a reasonable value (parameter `resolution` and `Quality` of the method `MultimediaTools.MakePhoto()`). For example, setting the `Quality` parameter to 50%, and the `Resolution` parameter to 50% of the maximum camera resolution, can reduce the time taken to take a photo by 1.5 times.

When trying to perform video shooting, in addition to specifying the type of camera, it is possible to specify the quality of the resulting video. Devices running on OS iOS ignore this parameter.

29.3.5.4.2. Playback

The mobile version of "1C:Enterprise" provides the ability to play audio notifications, play audio files and perform speech synthesis. The ability to perform speech synthesis depends on the language in which the synthesis is intended to perform.

To play a sound notification (short sound or vibration), the method `MultimediaTools.PlaySoundAlert()` is intended. It is possible to

play either the standard sound of the system or the sound file, the name of which is passed as a parameter `Sound` of this method. This sound file must be part of the mobile application. However, there is no possibility to attach resources (which include playable files) on the developer's mobile platform. In order for the specified file to be attached to a mobile application, it is necessary to use the features of the mobile application builder. Because of this, it is possible to check the use of sound notifications from files only in the assembled application. The ability to use vibration when playing a sound notification is controlled by the `Vibration` parameter of the `PlaySoundAlert()` method. If the file name with the sound notification is specified without an extension, then the following extensions (depending on the OS) are used by default:

- OS iOS - extension.caf;
- OS Android - extension .mp3;
- OC Windows - extension .mp3.

Playback notifications cannot be paused or stopped. It is always played utterly and completely.

Calling the global context method`Signal()` is equivalent to calling the method `MultimediaTools.PlaySoundAlert(SoundAlert.Default)`.

If necessary to play a sound file, then it is necessary to use the method `MultimediaTools.PlayAudio ()`. The source of playback can be the `MultimediaData` object (the result of the `MakeAudioRecording()` method), the `BinaryData` object (which contains the audio file of the required format directly) or the file name (`String` type). Playback of the audio file included in the mobile application is not supported. In the event that an application should handle the playback stop state, - it is necessary to specify a notification handler (in the `StopPlaybackHandler` parameter) that will be called if playback is stopped or interrupted.

It is possible to stop playing an audio file using the `MultimediaTools.StopAudioPlayback()` method. If the audio file is not currently playing, - nothing will happen.

From the point of view of the ability to play various formats of audio files, it is recommended to rely on the MIME file type. Playback and recording of audio files is only possible if these files have the following MIME types (depending on the mobile OS used):

- OS iOS - `audio/mp4`;
- OS Android - `audio/3gpp`;
- OS Windows - `audio/aac`.

In addition, it is possible to play audio files with the following MIME types:

- MP3 format - `audio/mpeg`;
- AAC format - `audio/aac`;
- WAVE format - `audio/vnd.wave` (for OS Android -starting from version 4.1 and higher).

Playing audio files whose MIME types differ from those listed above is not supported.

The following is a simple example in which there are two buttons on a form, clicking on one of which (the `PlayAudio()` handler) selects the file being played and starts playing it, and pressing the second button (`Stop()` handler) - stops playback.

```
&OnClient
PlayAudio Procedure(Command)
  Dialog = New FileDialog(FileDialogMode.Opening);
  If Dialog box. Select() Then
    Data = Dialog box.FullNameOfTheFile;
    MultimediaTools.PlayAudio(Data);
    File = New File(Dialog box.FullNameOfTheFile);
    Report ("" + File.GetMobileDeviceLibraryFilePresentation() + "
with duration of " + MultimediaTools.GetAudioDuration(Data) + "
seconds");
  EndIf;
EndProcedure

&OnClient
Stop Procedure(Command)
  MultimediaTools.Stop StopAudioPlayback();
EndProcedure
```

On a mobile platform, it is possible to use a speech synthesizer built into the operating system. The availability of synthesizer for a specific language should be checked using the `MultimediaTools.TextPlaybackSupported()` method. If a speech synthesizer is available, - it is possible to play the text using the `MultimediaTools.PlayText()` method. If it is necessary to process the interruption of speech synthesis, it is possible to set the appropriate handler using the `StopPlaybackHandler` parameter. It is possible to interrupt text playback using the `MultimediaTools.StopTextPlayback()` method.

29.3.5.5. Barcode scanning

A mobile device with a camera allows to perform barcode scanning functions. To access this feature, the platform provides a special interface.

NOTE. To effectively scan barcodes, it is desirable that the camera of the mobile device has the ability to automatically focus the camera lens on sharpness (autofocus). Otherwise, scanning barcodes will be difficult.

The general scheme of operation with a barcode scanner is as follows:

1. The ability to operate with the scanning mechanism is determined. To do this, it is necessary to use the `BarcodeScanningSupported()` method of the Global Context `MultimediaTools` object.
2. In the event that scanning is supported, the method `ShowBarcodeScanning()` of the `MultimediaTools` object opens a special barcode scanning interface. During the scan operation, a special handler is called that contains information about the scan result.
3. It is possible to close the scan interface either manually, using the standard Android interface button, or from the 1C:Enterprise language, using the `CloseBarcodeScanning` method of the `MultimediaTools()`object.

Let's consider an example of operation with this mechanism. For the demonstration, a simple form will be used, on which the button for opening the scan interface and the flag of closing this interface after a single scan is located. The scanned barcode will be displayed as a message.

NOTE. This example is not a completed tool. It is intended to demonstrate the operation with the barcode scanning mechanism.

It is necessary to create in the form the `CloseInterface` attribute of the `Boolean` type and place on the form an element that displays this attribute. It is also necessary to create a form command. `OpenScanInterface`, which should also be placed on the form. The form module will have the following type:

```
// Form command handler OpenScanningInterface
&OnClient
Procedure OpenScanningInterface(Command)
  ScanningHandler = New NotifyDescription("ScanHandle",
ThisObject);
  CloseHandler = New NotifyDescription("InterfaceClosureHandling",
ThisObject);
  MultimediaTools.ShowBarcodeScanning("Point the camera at the
barcode", ScanningHandler, CloseHandler);
EndProcedure

&OnClient
Procedure ScanHandling(Barcode, Result, Message,
AdditionalParameters) Export
  If Result Then
    Text = "" + Barcode;
  Else
    Text = "Error: " + Message;
  EndIf;
  If CloseInterface Then
    MultimediaTools.CloseBarcodeScanning();
  EndIf;
  Report(Text);
EndProcedure

&OnClient
Procedure InterfaceClosureHandling(AdditionalParameters) Export
  Report("ScanningInterfaceCloses");
EndProcedure

&OnClient
Procedure OnOpen(Cancel)
  If NO MultimediaTools.BarcodeScanningSupported() Then
    CallException "Barcodes scanning is not supported";
  EndIf;
  CloseInterface = False;
EndProcedure
```

When opening interface, two descriptions of notification are passed to the `ShowBarcodeScanning()` method. The first notification is triggered (`<ScanningHandler>` parameter) in the case when the system has successfully recognized the barcode, the second notification (`<Closure-Handler>`parameter) is triggered when the window intended for scanning the barcode is closed. Closure can be both software and interactive. An example of software closure is given in the `ScanHandling()` notification handler, and interactive closure can be performed by clicking the Back button on the scan interface.

When open the scan form, it is possible to specify which barcodes the system will recognize: linear (for example, EAN), two-dimensional (for example,

QR code) or all. This is set using the `<BarcodeType>` parameter of the `ShowBarcodeScanning()` method. It should be remembered that the indication at necessity to recognize all types of bar codes reduces scanning performance, since the system will spend more time searching for different bar codes in the image.

When scanning barcodes using the camera of a mobile device, the following barcodes will be recognized, depending on the operating system of the mobile device:

- OS Android:
  - Linear barcodes:
    - UPC-A;
    - UPC-E;
    - EAN-8;
    - EAN-13;
    - Code39;
    - Code93;
    - Code128;
    - RSS-14;
    - RSS Expanded;
    - ITF;
    - Codabar.
  - Two-dimensional barcodes:
    - QR Code;
    - Data Matrix.
- OS iOS:
  - Linear barcodes:
    - UPC-A;
    - UPC-E;
    - EAN-8;
    - EAN-13;
    - Code39;
    - Code93;
    - Code128;
    - RSS-14;
    - RSS Expanded;
    - ITF;
    - Codabar;
    - PDF417.
  - Two-dimensional barcodes:
    - QR Code;
    - DataMatrix;

    - Aztec Code;
    - MaxiCode.
- Windows OS:
  - Linear barcodes:
    - UPC-A;
    - UPC-E;
    - EAN-8;
    - EAN-13;
    - Code39;
    - Code93;
    - Code128;
    - ITF;
    - Codabar;
    - RSS-14;
    - RSS Expanded;
    - PDF417.
  - Two-dimensional barcodes:
    - QR Code;
    - DataMatrix;
    - Aztec Code;
    - MaxiCode.

29.3.5.6. Operation with email

The mobile version of "1C:Enterprise" provides an application developer with several ways to operate with email. One method allows to implement a full-fledged email client on the "1C:Enterprise" platform (`InternetMail` object and related objects), and another method allows only sending email messages using the default email application installed on the system (`MailTools` object).

When sending email messages of the `InternetMailTextType.HTML` type, using the `MailTools` object (running the Android OS), the message text is converted into Spannable format. The email sent is displayed by most email clients.

When using email messages texts with `InternetMailTextType.HTML` type (running on OS Android), it is not recommended to use the following features:

- cascading tables of styles;
- embedded tables;
- embedded images;
- fonts of different sizes.

It is recommended to check the display of emails on the maximum number of devices in order to minimize the dependence on implementation differences on different operating systems and in different versions of mail programs.
It should be kept in mind that the text type of email messages `InternetMailTextType.RichText` is not supported by the `MailTools` object.

29.3.5.7. Dealing with contacts

The contact list - is a database that stores information about the subjects with whom the owner of the mobile device communicates. The contact description contains the contact identification data, addresses, phone numbers and other information necessary to communicate with the subject. Also, accounts of various services can be registered on the mobile device. An account is a set of information that a user uses to identify himself in any service. For example, this is a username and password. There is always an account on the mobile device that describes the device itself (local account).
Each record in the contact list is associated with single account. The mobile operating system can synchronize the list of contacts with the account data (depending on the OS settings). If the contact is associated with a local account, then the information is not synchronized and all data is located only on the mobile device.
The main information about the contact is stored in the objects of `ContactData` type. Objects of this type contain all information about the contact, but do not contain the connection of the contact with the account. There are several data types in the contact description that can contain more than one value. Such data are phone numbers, addresses, email addresses and others. For storage of such data, the arrays of objects of the `ContactDataItem` type are intended. To store the list of identifiers in instant messaging programs, an array of `ContactDataItemInstantMessaging` objects is intended.
The accounts registered on this mobile device for contact synchronization are described using the `ContactAccount` object.
Relationship between the contact account and the actual elements of the contacts list forms the `AccountContactData` object. In addition to information about the contact and account, this object also contains ID by which synchronization between the list of contacts on the mobile device and the account is performed. It can be noted that the `GlobalContactKey` property of the `AccountContactData`object is an analogue of the `Link` property for "1C:Enterprise" applied objects.

To identify a contact on the mobile device, there is an object LocalContactKey. All operations with contacts (creation, modification, deletion, search) are performed using this object. To perform operations with contacts, use the contact manager (the object of the same name).
The following example shows creation of the elements of the contacts list in the local account:

```
&OnClient
Procedure CreateContactElement(Full name, Email, Telephone)

  ContactData = New ContactData;
  ContactData.Name = Name;
  ContactData.Patronymic = Patronymic;
  ContactData.Surname = Surname;
  EmailElement = New
ContactDataItem(ContactDataEmailAddressType.Office, Email);
  ContactData.Emails.Add(EmailElement);
  PhoneElement = New
ContactDataItem(ContactDataPhoneNumberType.Office, Phone);
  ContactData.Phone numbers.Add(PhoneNumbers);
  CM = New ContactManager;
  LocalContacts = CM.GetLocalContactAccount();
  Contact = New AccountContactData(ContactData, LocalContacts);
  CM.AddContact(Contact);

EndProcedure
```

29.3.5.8. Operation with calendar

There is a possibility on mobile devices to view the calendar and establishments various events in it. Events are characterized by some set of characteristics, such as the start and end date and time, description, employees, etc. Also, accounts of various services can be registered on the mobile device. An account is a set of information that a user uses to identify himself in any service. For example, this is a username and password. There is always an account on the mobile device that describes the device itself (local account).
There can be several calendars (with different names) on a mobile device. Each calendar is associated with any one account. The mobile operating system can synchronize calendar events with the account data (depending on the OS settings). If calendar events are associated with a local account, the information is not synchronized and all data is located only on the mobile device.
Basic calendar information is stored in the CalendarData object. Calendar events are described by the CalendarEventData object. These objects describe the characteristics of the calendar or calendar events, but do not describe which account the object is associated with.
Accounts registered on this mobile device for contacts synchronization are described using the CalendarAccount object.

The relationship between the calendar and event data and the accounts is made up by the `AccountCalendarData` and `AccountCalendarEventData` objects. These objects, in addition to information about the actual data and information about the account, store information about the value of the keys by which synchronization is performed between the mobile device and the service with which synchronization is performed. The `GlobalCalendarKey` is used to synchronize calendar data, and the `GlobalCalendarEventKey` is used to synchronize data about calendar events.
To identify these calendars and events on the mobile device, there are objects `LocalCalendarKey` and `LocalCalendarEventKey`. All operations with calendars and events (creation, modification, deletion, search) are performed using this object. To perform operations with calendars and events, use the calendar manager (the object of the same name).

29.3.5.9. Screen orientation

The platform provides information on some characteristics of the screen of the device on which the client application is running (personal computer or mobile device). The method `GetClientDisplaysInformation()` is intended for this. As a result, an array of structures describing the screens attached to the device is returned.
Each array element describes one screen attached to the device. `Height` and `Width` parameters describe the height and width of the screen in points, respectively. At the same time, for a mobile device, these parameters depend on the orientation of the device (if it is allowed in the settings), and the platforms for the personal computer always return the canonical parameters of the attached screens. Thus, on a mobile device, it is possible to analyze in which position the device is located and, based on this, make decisions about the transformation of the application interface. To facilitate orientation, there is an event `OnChangeDisplaySettings`, which can be processed in the managed application module and in the managed form module.
As an example, consider changing the display of a form displayed on the desktop. In this form, there are two lists that display some information. In the case of portrait orientation of the screen (the narrow side of the device - lower), the lists will be displayed one above the other, while in the landscape (wide side of the device - lower) orientation - lists are located side by side. To implement this behavior, in the form module located on the desktop, it is necessary to place an event handler `OnChangeDisplaySettings` of the following type:

```
&OnClient
Procedure OnChangeDisplaySettings()
  InformationArray = GetClientDisplaysInformation();
  If InformationArray[0].Width > InfaormationArray[0].Width Then
    ThisForm.Grouping = ChildFormItemsGroup.Horizontal;
  Else
    ThisForm.Grouping = ChildFormItemsGroup.Vertical;
  EndIf;
EndProcedure
```

29.3.5.10. Dealing with notifications

29.3.5.10.1. General information

During the operation of the information system, it is regularly necessary to notify the user of any event. An event can occur both on a schedule and in case of another event. The event itself may occur on a mobile device owned by the user, or in some other place of the information system.
Notifications are intended to notify the user about the occurrence of events. They can be created locally or sent via push services. Local notifications are used to inform about events that occur on the user's device. Typically, the source of push notifications is the central node of the information system. Using push notifications, it informs remote users that data important to users has changed. However, this is not the only scenario for using push notifications. Push notifications can send specialized services that inform about currency rates, sports scores, weather forecasts, etc.

29.3.5.10.2. Local notifications

Local notifications are used to notify on events that occur directly on the mobile device. Local notifications are instant and scheduled. They are used to inform about different events and, moreover, they are distinguished by the behavior on a mobile device. Instant notifications are used when it is necessary "here and now" to notify the user about occurrence of the event, for example, the end of the process. Such a process may be a synchronization operation with an external system. Scheduled notifications allow the user to be reminded of the need to perform an operation at the appointed time. Scheduled notification may be one-time or periodic. A one-time scheduled notification can be used, for example, if the system indicates that some counterparty needs to call back at a specific time. Repeated scheduled notification can be used, for example, in the case when some operation is repeated every day, regardless of external events.
To use local notifications, it is necessary to set the permission Local notifications. The mobile application builder adds the appropriate permission for

the collected mobile application. It should be noted that the "1C:Enterprise" mobile version for the developer has all the permissions to operate with local notifications. Therefore, after the development of the mechanism it is recommended to check its operation on the usual mobile version of "1C:Enterprise". To do this, use the mobile application builder and test the resulting application on one or more devices.

For operation with local notifications, the manager of delivered notifications is used (global context property `DeliverableNotifications`) and object `DeliverableNotification`. The object describes the notification itself (including its display parameters), and the manager controls the notification mechanism.

Let's consider creating local notifications in more detail.

```
&OnClient
Function CreateNotification(Title, Text, Data = 0, TriggeringDate,
RepetitionInterval = 0)

  Notification = New DeliverableNotification;
  Notification.Title = Title;
  Notification.Text = Text;
  Notification.Data = Data;
  Notification.FireDateUniversalTime = TrigerringDate;
  Notification.RepeatInterval = RepetitionInterval;
  Notification.SoundAlert = SoundAlert.Default;
  Return Notification;

EndFunction

&OnClient
Procedure SetNotification(Command)

  Notification = CreateNotification("Title", "Text", 2014,
'00010101');
  DeliverableNotifications.AddLocalNotification(Notification);

EndProcedure
```

When creating a notification, it is necessary to remember the following features:

- Date and time of the first display of the notification is set in the universal time format (UTC). To transfer from local time, it is possible to use the global context function `UniversalTime()`.
- If the `FireDateUniversalTime` property is set to an empty value (or not specified at all), the notification will be displayed immediately. This notification can notify the user that the mobile application has completed some time-consuming operation, for example, completed the data synchronization operation.

- In order to display notification regularly, it is necessary to use `RepeatInterval` property of `DeliverableNotification` object. As a result, it is possible to implement a simple option of the unconditional timer, which, after a specified time interval will generate a notification. The value that describes the notification repeat interval depends on the mobile operating system used. The features of specifying the interval are described in the syntax assistant. The system does not provide the ability to specify the date and time of the end of the notification. For Windows OS, the notification will be shown 5 times (including the very first display).
- If necessary, it is possible to set the sticker- number displayed in the notification blind (Android OS) or on top of the application icon (iOS OS). `Sticker` property is intended for this. When running on OS iOS, it is possible to set and retrieve the value of the sticker using methods `SetBadge()` and `GetBadge()` of manager of delivered notifications. At operation in Windows, the ability to create a sticker on the application icon is not supported.
- When a notification is displayed, it is possible to play a certain melody. To specify the played melody, the property `SoundAlert` of the object `DeliverableNotification` is intended. If value of system enumeration `SoundAlert.Default` is set as the value of this property, then the melody set for notifications in the used operating system will be played. If the name of the file is specified as the value of the `SoundAlert` property, then this file will be played. However, there is no possibility for the developer to attach resources (which include playable files) in the mobile version of "1C:Enterprise". In order to attach a file with notification to the mobile application, it is necessary to use features of the mobile application builder. Because of this, there is a possibility to check the usage of sound notifications from files only on the assembled mobile application.

In some cases, it may be necessary to delete installed (but not yet arrived) notifications. This may be necessary in cases where the action for reminding about which a notification has been set has already been completed. To delete a notification, it is possible to use the `CancelLocalNotifications()` method of the manager of the delivered notifications. It should be remembered that all local notifications that were set by this mobile application are canceled at once, and the response time of which has not yet come.

After the notification appears, clicking on it will launch (or activate) the application. However, this may not be enough to fully process the notification. In this case, using the notification handler is possible. This handler is attached

using the `AttachNotificationHandler()` method of the manager of delivered notifications.

```
&OnClient
Procedure OnOpen(Cancel)
  Handler = New NotifyDescription("NotificationHandler",
ThisObject);
  DeliverableNotifications.AttachNotificationHandler(Handler);
EndProcedure

&OnClient
Procedure NotificationsHandler(Notification, Local, Shown,
AdditionalParameters) Export
  // Important and necessary actions
EndProcedure
```

As a result, after the user of the mobile application clicks on the notification, the mobile application that created the notification will be activated, and the notification handler will be called in this application. Within the border of this handler, it is possible to implement the necessary notification processing. It should be remembered that the object that is passed in the `Notification` parameter contains real data only in the `Text` and `Data` properties. Other properties are filled with default values.
It should also be noted that the notification may not be displayed in the notification blind. To inform the mobile application about this fact, the parameter `Shown` is passed in the notification processing procedure. So, if the mobile application is running and active, then the local notification will not be displayed on the screen, but the notification handler will be called, and the `Shown` parameter will be set to `False`. In this case, the task of attracting the user's attention lies with the application developer.

29.3.5.10.3. Push-notification

General information

Push-notifications (further in this section, the terms "notification" and "push-notification" are used equally) are used to notify about events that have happened somewhere else. For example, in the corporate information system, some data that is important for mobile users of the system has changed.
At operation with push-notifications, several different systems are used:

1. Mobile application itself. Receives notifications and implements the response to them from the mobile device.
2. Application that deals with sending notifications (sender). The application created on the "1C:Enterprise" platform acts as the sender.

3. Service of notifications delivery. This service provides a transfer of notification from the sender to the recipient (to mobile application). This service can be:
   - Apple Push Notification Service (APNs, https://developer.apple.com/library/content/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/APNSOverview.html, in English language);
   - Firebase Cloud Messaging (FCM, https://firebase.google.com/docs/cloud-messaging/, in the English language);
   - Windows Push Notification Services (WNS, https://docs.microsoft.com/windows/uwp/controls-and-patterns/tiles-and-notifications-windows-push-notification-services--wns--overview, in the English language).
4. 1C Company has developed a specific auxiliary service for sending notifications. This service is designed to facilitate the implementation of sending notifications from the circulation of applications. The service allows to use all the services of sending notifications with which the mobile version of "1C:Enterprise" can operate. The service allows to isolate the recipients of notifications of one sender from other senders who operate with the same mobile application. In addition, the service allows to avoid the publication of confidential information, which can be directly prohibited by the policy of the service of sending notifications. The features of operating with this service will be described later in the corresponding sections. It should be understood that the service of the 1C company can not operate without using the actual services of sending notifications. Service is located at https://pushnotifications.1c.com.

In general, sending a notification is as follows:

- The sender generates a notification;
- The sender determines the list of recipients of the notification;
- The sender attaches to the notification service;
- The sender passes the service notification and the list of recipients;
- The service provides delivery of notifications to mobile devices;
- Mobile application on the device provides notification processing.

Push-notifications delivery service does not guarantee delivery of a notification to a mobile device.

To use notifications, it is necessary to set the permission Push-notifications. The mobile application builder adds the appropriate privilege to the mobile application being built. Operation with the mobile version of "1C:Enterprise"

for the developer has some features that will be discussed below. In any case it is recommended after development of a mechanism to check its operation on the usual mobile version of "1C:Enterprise". To do this, use the mobile application builder and check the performance of the resulting mobile application using all notification delivery services.
Processing a push notification is a little different from processing a local notification. It is obviously that the `Local` parameter will be set to `False` if the handler is called to process a push notification.

The scheme of operation with the services of sending notifications

To understand how to deal with push-notifications, it should be noted that several different roles are involved in the deployment and implementation of the system. In a number of cases, role performers can unite, but the number of roles remains unchanged:

- **Mobile application developer** - is developing a mobile application (notifications recipient). Implements all the necessary functionality of the application to receive notifications.
- **Developer of notifications sender** - is developing an information system that distributes notifications and a mechanism for obtaining and storing identifiers of notifications recipients.
- **User of mobile application** - uses the mobile application. Setting of mobile application is also performed by this user, so the settings of the mobile application should be as simple and understandable as possible even to an inexperienced user.
- **Sender's integrator** - installs and configures the system that sends notifications to the user's network.

When considering schemes of operation with a particular service, the description of actions will occur on behalf of the carrier of a particular role.
The scheme of operation contains a description of the actions to be performed by the representative of a particular role at the time of the execution of their actions. The order for description actions will always be the same (mobile application developer, sender developer, sender integrator, mobile application user), but this does not mean that the physical order will be exactly the same.
In addition to the actions described later in the section, the implementation of the following steps is considered mandatory, and further about them separately will not be mentioned:

- The implementation of the reaction to the notification - is performed by the developer of the mobile application;

- The implementation of sending notifications - is performed by the developer of the sender.

It is also considered that the mobile application and the sender "know" that they can interact with each other:

- The sender has some public program interface (exchange service) in order to get the notification recipient ID from the mobile application.
- The recipient (mobile application) can use the exchange service that the sender publishes to perform both actions.

Next will be considered the actions (by roles) that must be performed to operate with a particular service of sending notifications. Step-by-step instructions for performing specific steps will be given later in this section.

APNs

The mobile application developer must:

- Register for the mobile application the option of receiving push-notifications on the Apple website. At registration you should receive a specific **certificate**.
- Provide the received certificate to the sender of notifications.
- Implement receiving the notification recipient ID and delivering the resulting object to the notification sender.

The sender’s developer must:

- Implement a mechanism for entering and storing certificates to recipients of notifications.
- Implement a mechanism for obtaining and storing identifiers of recipients of notifications.

The sender integrator must:

- Upload the certificate of the mobile application to the sending system, which will receive notifications.
- The certificate must be obtained from the developer of the mobile application.

The user of the mobile application must:

- Provide the mobile application with the parameters of the exchange service that the sending application provides.
- Initiate the command to create a notification recipient ID and send it to the notification sender.

FCM

The mobile application developer must:

- Register for the mobile application the ability to receive push notifications in the Firebase service. When registering, you should get a **server key**.

- Provide the received server key to the sender of notifications.
- Implement receiving the notification recipient ID and delivering the resulting object to the notification sender.

The sender's developer must:

- Implement the mechanism for entering and storing the keys of the servers of notifications recipients.
- Implement a mechanism for obtaining and storing identifiers of recipients of notifications.

The sender integrator must:

- Download the key of the Firebase project server for the mobile application to the sending system, which will receive notifications.
- The server key must be obtained from the developer of the mobile application.

The user of the mobile application must:

- Provide the mobile application with the parameters of the exchange service that the sending application provides.
- Initiate the command to create a notification recipient ID and send it to the notification sender.

WNS

The mobile application developer must:

- Register for the mobile application the ability of receiving push-notifications on the Microsoft website. When registering, you should get Application secrets and Application security IDs.
- Provide the received Application secrets and Application security IDs to the notifications sender.
- Implement receiving the notification recipient ID and delivering the resulting object to the notification sender.

The sender's developer must:

- Implement a mechanism for entering and storing Application secrets and Application security IDs. This data is necessary to obtain an access marker, which, in turn, is used to send notifications.
- Implement getting, storing and updating access markers.
- Implement a mechanism for obtaining and storing identifiers of recipients of notifications.

The sender integrator must:

- Specify in the sending system the Application secrets and the Application security IDs that will receive notifications.
- Application secrets and Application security IDs must be obtained from the mobile application developer.

The user of the mobile application must:

- Provide the mobile application with the parameters of the exchange service that the sending application provides.
- Initiate the command to create a notification recipient ID and send it to the notification sender.

Service of 1C Company

The 1C Company service allows to somewhat simplify the process of sending and receiving notifications for circulation solutions.
The mobile application developer must:

- Choose which services will be used to deliver notifications to a mobile device. Then it is necessary to provide 1C Company service with artifacts that are necessary for sending notifications for the used services. It should be remembered that the artifacts provided can be confidential information.
- Implement receiving the notification recipient ID and delivering the resulting object to the notification sender.

The sender’s developer must:

- Implement the mechanism for entering and storing the application server key (Sender access key) for the current instance of the system that sends notifications. Registration of a specific instance of the sender on the website of the 1C Company service is performed by the sender's integrator.
- Implement a mechanism for obtaining and storing identifiers of recipients of notifications.

The sender integrator must:

- To register a specific instance of the sender on the 1C Company service website. The resulting server key of the registered application (Sender access key) is entered into the appropriate sender settings.

The user of the mobile application must:

- Provide the mobile application with the parameters of the exchange service that the sending application provides.
- Initiate the command to create a notification recipient ID and send it to the notification sender.

Operation with APNs service

To test the mechanism for sending notifications on OS iOS, you need to build your own mobile version of 1C:Enterprise for the developer. When building, you should use a provision profile for which is possible to operate with push-notifications. When creating a "own" mobile version of "1C:Enterprise" for

the developer, you must specify the correct ID of the mobile application, and the identifiers `com.e1c.mobile` and `com.e1c.mobile.ios` cannot be used, since they are reserved for use by 1C Company.
To establish a connection with APNs, a sending certificate of the mobile application of the notification recipient must be available on the sending computer. Certificates for APNs are of two kinds Apple Development IOS Push Services (developer certificate) and Apple Production IOS Push Services (public mobile application certificate). The developer certificate is designed to develop a mobile application and debug it. It should be used if the application is installed on a device using development tools. The certificate of the public mobile application should be used if the mobile application is distributed through the application store.
The mobile version of "1C:Enterprise" imposes the following requirements for the certificate used:

- Application certificate must be provided in .PEM format.
- The certificate must contain section Bag Attributes, in which the attribute friendlyName must be present. Under the attribute value friendlyName, the system distinguishes between a developer certificate and a public mobile application certificate. Certificate without attribute friendlyName will be rejected by the mobile version of "1C:Enterprise".

IMPORTANT! Description of the developer console is as of the writing of the documentation.

To prepare for sending notifications using APNs, the mobile application developer must perform the following actions:

1. To enter the developer console (iOS Dev Center, https://developer.apple.com/devcenter/ios/index.action). The user on behalf of whom the user is logging into the console must have administrator rights (in the developer console).
2. In the right part of the web browser window, select the menu item Certificates, Identifiers & Profiles in section iOS Developer Program.
3. If the mobile application that will receive notifications does not have App ID, then it must be created. The order of creation is described below. If App ID exist -, skip this item and go to the next item.
   1. In section iOS Apps select item Identifiers.
   2. Press the button + above the list iOS AppIDs. If the user has insufficient rights - the create button will be unavailable.
   3. In the Name field, the application description must be specified (in Latin letters). In section App ID Suffix select item Explicit App ID.

In the Bundle ID field, the full ID of the mobile application (available from the mobile application builder) should be specified. In the App Services section, Push Notifications item should be pointed.

4. To go to the next screen, click Continue.
5. Check all the specified parameters and click Submit button to confirm creation App ID.
6. Clicking the Done button completes the App ID creation process.

4. It is necessary to create a provision profile to build a mobile application that can receive notifications. This provision profile should be used in the future in the mobile application builder.
   1. You should select a menu item Provisioning profiles - All on the left of the screen Certificates, Identifiers & Profiles.
   2. You should press the button + above the list iOS Provisioning Profiles.
   3. Profile view is selected depending on what it is created for. If a provision profile is created for developing and debugging an application, then in the Development section, it is necessary to select iOS App Development item. If a provision profile is created for application distribution, then in the Distribution section, it is necessary to select App Store item. Depending on the type of provision profile chosen, the further steps of the assistant will vary.
      1. Provision profile is selected for developer and the Continue button is pressed.
         1. You should select App ID of application, for the assembly of which a provision profile is created, and press the button Continue.
         2. You should select certificates for those developers who can create applications in Xcode using this provision profile and then click Continue button.
         3. You should select the devices on which the application will operate, assembled with this provision profile and then click Continue button.
         4. You should enter a description of the profile being created (in Latin letters) and press the button Generate.
      2. Provision profile is selected, Continue button is pressed for distribution.
         1. You should select App ID of application, for the assembly of which a provision profile is created, and press the button Continue.

    2. You should select the certificate of the company-provider of the application and then click Continue button.
    3. You should enter a description of the profile being created (in Latin letters) and press the button Generate.
  3. Then you should download the created provision profile using the Download button for further use (in the Xcode system or the mobile application builder).
5. To create a mobile application certificate, you first need to generate a certificate request file. Mac computer is required to perform this operation. Do not close the web browser window with the developer console open while creating a certificate request.
   On a Mac:
   1. It is necessary to start the Keychain utility.
   2. Select the command Keychain - Certification Assistant - Request a certificate from a certification authority.
   3. In the opened dialog box you should fill in the fields, at the same time as the field value User's E-mail it is recommended to specify the e-mail address that was specified during registration in the iOS Developer Program program. As Request property value, you should specify Saved to hard drive.
   4. After pressing Continue button, the file of request of the certificate *.certSigningRequest will be generated.
6. Generate the certificate for the sender of notifications. To do this, return to the web browser window with an open developer console.
   1. You should select menu item Certificates - All on the left of the screen Certificates, Identifiers & Profiles.
   2. You should press + button above the list iOS Certificates.
   3. Depending on your needs, you should specify the item Apple Push Notification service SSL (Sandbox) (if you need a certificate for developing and debugging an application) or item Apple Push Notification service SSL (Production) (if you need a certificate for the distributed application). Then click Continue.
   4. You should specify App ID of the used application and press Continue.
   5. Click Continue button. This screen prompts you to create a certificate request file that has already been created in the previous step.
   6. You should download the file generated in the previous step (with extension .certSigningRequest) and press Generate button.
   7. After completion of the generation of the certificate, it should be downloaded from the website using the button Download.

7. The final step is to convert the certificate into the required formats. To do this, on the Mac where the certificate request file was generated, do the following:
    1. For conversion, use the program Keychain.
    2. The certificate obtained on the website is downloaded to Keychain.
    3. Certificate is being exported from Keychain in the format .P12.
    4. The resulting certificate is converted from the format .P12 into the .PEM format. To do this, run the following command in the macOS console:

```
openssl pkcs12  in pushcert.p12  out pushcert.pem  nodes  clcerts
```

The resulting certificates are required for:

- Certificate in format .P12 - is required if the notification is sent using the 1C Company service. The certificate should be used by the recipient's developer to register the application-recipient of notifications in the 1C Company service.
- Certificate in the .PEM format - is required if notifications will be sent directly through the APNs service, in which case the certificate must be provided to the sender's developer or the sender's integrator (if the interface of the notifier-sender program provides the option of specifying the certificate of the notifying program).

Operation with FCM service

IMPORTANT! Description of the developer console is as of the writing of the documentation.

Registration of new application for FCM

The following actions must be performed by the integrator of the sending application during deployment and settings of the sending application on the customer's computer network:

1. You should enter the developer console (Firebase Console, https://console.firebase.google.com).
2. You should create a project (Add project button). When creating a project, indicate the name of the project and the country where the organization is located. The project name should contain only Latin characters. After you finish creating the project, click the Continue button.
3. On the project page, click the Add Firebase to your Android application hyperlink. After that you should do the following:

1. In the Android package name field, specify the full application ID. With this ID, the application will be placed on the Google Play store. The same ID will be used to build the mobile application in the mobile application builder. After specifying the application ID, click the Register application button.
2. Download the proposed file google-services.json. This file will need to be specified in the mobile application builder. After downloading the file, click Continue button.
3. Click Finish button.

4. In the application settings, go to the Cloud Messaging tab.
5. The value located in the Server key field must be used as the authentication key for sending notifications (`AuthenticationData` parameter of the `Send()` method of the `DeliverableNotificationSendManager` object). It is recommended to treat this key as confidential information. Receiving this key by third parties will allow them to send push-notifications on your behalf.

The resulting server key (Server key) must be provided to the integrator of the sending application in order to be specified in the application settings.

Transition from using GCM to using FCM

The following actions must be performed by the integrator of the sending application during deployment and settings of the sending application on the customer's computer network:

1. You should enter the developer console (Firebase Console, https://console.firebase.google.com).
2. You should create a project (Add project button). In the Project name field, click the select button from the list and select the GCM application to be transferred to FCM. After that, specify the country of the organization. Then click the Add Firebase button to start the import process.
3. The remaining steps are similar to those described above.
4. You should also change the code in the 1C:Enterprise language so that sending notifications uses the FCM service.

Description of operation with GCM service

Description of operation with GCM service is given in the documentation for 1C:Enterprise, version 8.3.12 and earlier. (https://its.1c.ru/db/v8312doc#bookmark:dev:TI000001543).

Operation with WNS service

IMPORTANT! Description of the developer's portals is provided as of the writing of the documentation.

The following actions must be performed by the integrator of the sending application during deployment and settings of the sending application on the customer's computer network:

1. You should log into the Windows Development Center. (https://developer.microsoft.com/windows).
2. It is necessary to go to the information panel (Information panel button).
3. It is necessary to create a new application (Create a new application button). On the opened page, it is necessary to specify the name of the application that will be displayed in the application store. After specifying the name, you should check the availability of this name (Check the availability link) and then click the Reserve product name button.
4. Enter into the Application Registration Portal (https://apps.dev.microsoft.com).
5. You should find the created application and open its page.
6. On the application page, it is necessary to get the values specified in the Application secrets and the Package security IDs sections. It is recommended to treat this key as confidential information.
7. The received values should be specified as parameters for calling the `GetAccessToken()` method of the `DeliverableNotificationSendManager` object. As a parameter of the `ApplicationID` method, the value from the sectionApplication secrets is specified, and as the value of `ApplicationKey` parameter, the value from the Package security IDs section is specified. The resulting access marker will be used to send notifications.

Received Application secrets and Package security IDs must be provided to the sender of the sending application to be specified in the application settings.

When operating WNS, the value of the property `DeliverableNotificationSubscriberID.ApplicationID` contains Push Notification Channel. Push Notification Channel is a URI for receiving notifications by a mobile application. Sending notifications is done via a POST-request to the URI. The channel is tied to a specific application on a specific device. It should be remembered that the value of the Push Notification Channel may change over time. It is recommended that the mobile

application requests a new notification subscriber ID each time it is launched, and notify it to its service for sending notifications. Notwithstanding the foregoing, the developer should consider the value of the `ApplicationID` property as a string with unknown contents and should not change its contents during operation.

Operation with the 1C Company service

The mobile application developer must:

- Register with the notification service that the application will use. If the application uses multiple services, - you must register in all. Save the artifacts that each notification service provides.
- It is necessary to register on the service website https://pushnotifications.1c.com/signup. To register, it is necessary to have a valid mobile wireless radiotelephone number (mobile number). During registration, a registration confirmation code will be sent to it. Contact information is recommended to indicate the data of the responsible person of the mobile application developer.
- It is necessary to register a mobile application that will be the recipient of notifications. To do this, select the menu item Delivered notifications Mobile applications- and click the Register application button.
- It is necessary to attach the created mobile application to the used services. To do this, in the Your mobile applications list, click the Attach to APNS, GCM, FCM or WNS link. On the next screen, click the button corresponding to the service you are using and enter the parameters for operation with this service (the full mobile application ID is required for all notification services):
  - APNS service. + Attach to APNS button. Required artifacts: mobile application certificate in.P12 format (and its password).
  - GCM service. + Attach to GCM button. Required artifacts: not required.
  - WNS service. + Attach to WNS button. Required artifacts: Application secrets value acquired in the Windows application store.
  - FCM service. + Attach to FCM button. Required artifacts: project server key values (Server key) acquired from the Firebase developer Console.

The sender's developer must:

- No action is taken.

The sender integrator must:

- It is necessary to register on the service website https://pushnotifications.1c.com/signup. To register, it is necessary to have a valid mobile

wireless radiotelephone number (mobile number). During registration, a registration confirmation code will be sent to it. Contact information is recommended to specify information of the information system administrator.

- Register the sender of notifications. To do this, select the menu item Delivered notifications Senders of notifications- and then click the Register sender button. When registering, it is recommended to specify a list of IP-addresses from which it will be possible to send notifications using this service. If the addresses are not specified, - sending notifications is allowed from any computer that can log in to the service. It is important to understand that the sender's registration is not performed by the sender's developer, but by the sender's integrator at the time when the sender system instance is being set.
- As a result of registration, an access key will be generated. To save this value, click the Copy Access Key button or click the Edit button and copy the value from the Sender access key field. Acquired value must be specified in the settings of a specific instance of the application-sender. It will be used as an authentication key for sending notifications (`AuthenticationData` parameter of the `Send()` method of the `DeliverableNotificationSendManager` object). It is recommended to treat this key as confidential information. Receiving this key by third parties will allow them to send push-notifications on your behalf, for example, if specific values of IP-addresses were not specified when creating the sender.

The user of the mobile application must:

- Provide the mobile application with the parameters of the exchange service that the sending application provides.
- Initiate command execution to create a notification recipient ID and send it to the sender of the notification.

Notifications mailing

The actions for sending notifications are almost the same for different services. The differences are observed in the values of the parameters of the method of sending notifications. The actual sending takes place using the `Send()` method of the delivery manager of the notifications sent:

- (parameter `AuthenticationData` of the `Sent()` method) is used during notifications sending as a data of authentication:
  - APNs service - mobile application certificate acquired from the developer console. It is recommended to treat this key as confidential

information. To ensure that notifications are sent using APNs, outgoing connections on 2195 and 2196 IP-ports must be allowed on the computer with the sending system.

  - FCM service - project server key (Server key), registered in the developer console Firebase Cloud Messaging. It is recommended to treat this key as confidential information.
  - WNS service - value acquired from the use of the `GetAccessToken()` method.
  - 1C Company service - the key of the program interface of the sender (Key of the access of the sender), which was received during the registration of the sender on the website of the service. It is recommended to treat this key as confidential information.
  - If the 1C Company service is used to send notifications, then the `UseIntermediateService` parameter should be set to `True`.

- If sending is performed simultaneously using multiple services, then the authentication data can be a match, where the key is the type of service (the value of the `DeliverableNotificationSubscriberType` enumeration), and as the value, - those data that are needed by the corresponding service.

  It should be understood that it is impossible at the same time (in one call of the `Send()` method) to send notifications using the notification delivery service (APNs, FCM, WNS) and the 1C Company service. If in the `Send()` method, the value of the `UseIntermediateService` parameter is set to `True`, then the value of the `AuthenticationData` parameter can only be the Sender access key received from the 1C Company service.

- The list of recipients of notifications is described using the `Recipients` property of the `DeliverableNotification` object, which is the value of the first parameter of the `Send()` method of the delivery manager of the notifications being sent.

When operating with WNS service, remember that the value of the access marker (which is received using the `GetAccessToken()` method) is not a constant value. It must be periodically received repeatedly. To determine if this procedure needs to be performed, the `Send()` method has the `DeliverableNotificationSendIssuesInformation` returned parameter. If any problem occurs while sending notifications, the above parameter will contain a list of detected problems. If in the list of problem there is a problem of `AuthenticationDataError` type, this means that you need to re-get the access marker and try again to send a notification.

29.3.5.10.4. Features of mobile application notification processing

If the notification handler is not attached, the notification is stored in memory until the end of the session and will be passed to the notification handler after it is attached (in this session). If before the end of the session, the handler was not attached, - the notification is lost after the end of the session.
It is also necessary to consider the following features of operation with notifications in a mobile application:

- If the mobile application is active, the notification is immediately delivered to the mobile application, while the sticker and the sound notification are ignored.
- If the mobile application is not active at the time of receiving the notification, the notification is displayed by the operating system. If the user clicks on the notification, the corresponding application will be started. If the user has deleted the notification, - the application will not be informed of the existence of the notification. On OS iOS (version 7 and earlier), the notification is deleted from the notification center either by an explicit user action, or is supplanted by other notifications of this application.
- If the mobile application, at the time of receiving the notification, is running in the background and is not active, then operation with the notification is performed differently on different platforms:
  - In OS iOS, the notification will be passed to the mobile application only after it is selected by the user from the notification panel.
  - In OS Android, the notification is immediately passed to the mobile application and simultaneously displayed in the notification panel. After the user has selected a notification in the panel, the application is activated, but the notification is not re-passed to the application.
  - In Windows OS, the notification will be passed to the mobile application only after it is selected by the user in the notification center.
- In the event that several information bases are deployed in a mobile application, the user is prompted to switch if he is currently working with another infobase. If a specific infobase cannot be determined, - the corresponding diagnostic message is issued.

29.3.5.11. Dealing with advertising

29.3.5.11.1. General information

Mobile version of "1C:Enterprise" provides the option to display advertising information in a mobile application. If you need to play advertising, an application developer can customize the display of an advertising banner using the Google AdMob (https://www.google.com/admob/). This service is accessible working under the management of Android OS and iOS.
The advertisement can be of several kinds:

- Banner -a rectangular declaration occupying a part of the application main window. It can be updated automatically after some time.
- The full-screen banner mode -the full-screen declaration format used for display at the moment of going from one application form to another. Such declaration occupies the whole screen of a mobile device.
- A rewarded video ad -the advertisement format presupposing a reward for viewing.

This documentation addresses only issues related to the technical aspects of the mobile application with the advertising service.
The general scheme of managing advertising is as follows:

1. Register your application in the service. Get application identifier from the server.
2. Customize ad units in the service (if necessary).
3. Implement a program interface for operation with advertising, using the data received in the previous steps.

To manage the display of advertising information, the global context property `AdRepresentation` is intended. For simplification, the method prefix `AdRepresentation` will be omitted in the documentation. It is evident that such omissions are unacceptable in examples and real applications. After starting the mobile application, the system has the following default settings:

- Advertising in the mobile application is not used.
- No CIDs of any ad units are defined.
- The ad units are not loaded.
- Advertising information is not displayed on the screen.

In order for advertising information to become displayed in the application, the following actions are necessary:

1. Use of advertising in the mobile application must be allowed using the method `SetUsing(True);`.
2. Depending on the ad unit used, the unit CID shall be specified:
   - Banner- method `SetAdvertisingBannerID()`.

- Full-screen advertising banner -method `InstallFullScreenAdvertisingID()`.
- Rewarded video ad- method `SetRewardedVideoID()`.

3. The advertising information display is launched depending on the used ad unit:
   - Banner -method `SetAdvertisingBannerRepresentation()`. This method allows specifying the place of ad display or disabling the ad display.
   - Full-screen advertising banner -method `DisplayFullScreenAdvertising()`.
   - Rewarded video ad -method `ShowRewardedVideo()`.

   In the case of the full-screen ad or rewarded video display, the display completion notification handler can be set.
4. The time delays caused by the necessary ad unit loading are possible during the advertising information display. These delays can be shortened by loading the ad units in advance:
   - Banner -method `StartAdBannerDownloading()`.
   - Full-screen advertising banner -method `StartFullScreenAdvertisingDownloading()`.
   - Rewarded video ad -method `StartRewardedVideoDownloading()`.

   The relative handler will be called after the downloading completion. This handler allows determining the ad loading status and performing the necessary actions with the help of the method parameter.

   The preliminary downloading of the advertising information is not compulsory. The ad representation method call will automatically download the needed ad unit.
5. Rotation of the advertising information itself is provided by the advertising network.

`GetAdStatus()` method is intended to understand the status of a certain ad unit (by the ad unit ID).

The IDs obtained in the system web-interface shall be used for the ad ID setting.

The advertisement displayed in the mobile application shall pass AdMob advertising service rules (https://support.google.com/admob/answer/6128543). Special attention shall be given to the following sections of the Rules:

- Rules and recommendations for AdMob publishers;
- Rules for the rewarded resources.

29.3.5.11.2. Setting of advertising services

Google AdMob

IMPORTANT! The description of operating with the web interface is given as of the time of writing the documentation.

To customize ad units, follow these steps:

1. Sign up for Google AdMob.
2. Login as a registered user (https://apps.admob.com).
3. If the application is built using the mobile application builder and previously this application was tested, then nothing needs to be done. Ad units identifiers must be specified in the appropriate places in the application. In the mobile application, assembled with the help of the assembler, should display real advertising information.
4. Add a new application. To do it, select Applications -Add application command in the menu.
5. The adding of an application differs for the applications already published not published in the ad store. This selection is made when answering the question Have you already placed your application in Google Play or App Store?
   - The application is published in the store:
     - Answer Yes to the question.
     - Insert a fragment of the application name and press the button Find on the next page.
     - Select the required application from the list.
   - The application is not published in the store:
     - Answer No to the question.
     - The application name and the operating system under which management it operates shall be specified on the next page. If a mobile application (for which the ads display is set) will operate under the management of Android OS and iOS then two AdMob applications shall be created.
     - The necessary mobile application will be created after pressing Add button.
6. Application ID field value shall be remembered on Application settings page. This value needs to be specified to the mobile application builder.
7. After an application creation, go to Ad units page of this application.

8. During an ad block creation, it is necessary to specify what ad unit will be created: Banner, Inter-page advertisement (full-screen ad banner) or Rewarded (rewarded video). To do it, press Select button next to the required type of advertisement.
9. The name (property of the same name) is formed and ad unit other parameters (if necessary) are specified afterwards.
10. Create ad unit button pressing completes the creation operation. The button will be inactive if not all parameters necessary for the ad unit creation are set.
11. Two types of IDs will be presented on Ad unit created page. The first iD -is the application ID and the second ID shall be remembered for the use in the embedded language. The same ID can later be found in the ad unit properties as Ad unit ID field value.

Operation with ad units can be performed as follows:

1. Sign in to Google AdMob (https://apps.admob.com).
2. Select the required application from the list.
3. Specify Ad units item in the left part of the page afterwards.
4. Now, the required ad units can be created and set. The ad unit ID necessary for the use in the embedded language will be accessible in the properties of each ad unit.

Identifiers received when creating ad units should be used as parameters of the methods `AdRepresentation. Set the Advertising Banner ID()` and `AdRepresentation. Install the Full Screen Advertising ID()`.

29.3.5.11.3. Testing operation with advertising Google AdMob

The general scheme of testing the display of advertising information when using the Google AdMob can be represented as follows:

- Testing display of advertising information does not require publication of the mobile application in the application store.
- For testing, only ad unit identifiers are needed, for the creation of which (ad units) it is necessary to register the application being developed in the Google AdMob service.
- After registering the application in Google AdMob, you need to create ad units. Their identifiers will be used in the developed mobile application and during testing and in actual operation.

- While the application is running on a mobile developer platform, ad units display test content. After the application is assembled by the mobile application builder, - it will start displaying real advertising information.

29.3.5.12. Dealing with in-app purchases

29.3.5.12.1. General information

The mobile version of 1C:Enterprise provides an application developer with the capability to make various purchases within the mobile application (in-app purchase). The feature to operate with the following types of purchases:

1. Permanent purchase. It is used in all supported operating systems. It is purchased once for the user account of the mobile device. Applies to all mobile devices running under one account. For distribution, a purchase recovery mechanism is used.
2. Spendable purchase. Used in OS iOS and Windows. Such a purchase can be purchased any number of times. Information on acquiring of such a purchase is received on a mobile device only at the time of the request to start the purchase. In the Android OS, a spendable purchase is implemented in an application using platform tools (spending of a permanent purchase).
3. Subscription. It is used in all supported operating systems. Such a purchase is acquired as a permanent purchase, valid for a certain time. There is a feature to set an automatic renewal of the purchase upon expiration of the subscription. In this case, the subscription value will be written off from the payment instrument account linked to the user account. You can not change the cost of a subscription in the Android OS. In OS iOS, changing subscription costs will disable automatic renewal of subscriptions. Deleting the application through which the purchase was made does not disable automatic renewal of subscriptions. The subscriptions are managed with the help of Apple AppStore, Google Wallet or via Microsoft account. You cannot implement a subscription management interface in a mobile application.
   In the case of Windows OS use, the subscriptions will be accessible only if Windows 10 starting from 1607 version is installed on the mobile device. SDK Windows 10 of 14393 or earlier version shall be installed in the mobile application builder server.

Purchasing in the mobile application is carried out using external services:

- OS iOS: Apple In-App Purchase.
- OS Android: Google Play In-App Billing. The device must have the latest version of the Google Play application installed.

- Windows: Windows In-App Purchase.

Purchasing inside a mobile application, remember that the purchased objects are restricted by the owners of the application stores (the companies Apple, Google, and Microsoft).

In general, in-app purchases must fully comply with the requirements of service owners:

- OS iOS: https://developer.apple.com/app-store/review/guidelines/ (in English).
- OS Android: https://support.google.com/googleplay/android-developer/answer/6151557.
- Windows: https://docs.microsoft.com/en-us/legal/windows/agreements/store-policies#108-financial-transactions (in English).

As an example, the following types of content are permitted and prohibited:

- It is allowed to sell:
  - Functionality of mobile applications.
  - Services directly related to mobile applications.
- It is forbidden to sell:
  - Goods and services used outside of mobile applications.

29.3.5.12.2. Scheme of use

To use purchases, set the permission for the developed mobile application In-app purchases. The general scheme of operation with in-app purchases, in a mobile application, is as follows:

1. It is necessary to configure the operation of the in-app purchases services to interact with the developed mobile application. When setting up services, you may need to upload a developed mobile application to the application store, for which the option of using in-app purchases is enabled. At the same time, the downloadable mobile application may not contain purchasing functionality. The downloadable application is necessary for the operation of the purchasing services mechanisms. However, the full ID of the downloadable application must be exactly what it will be for the final version of the mobile application with the feature to make in-app purchases.
2. After setting up the services, you need to create the necessary purchases (in each of the services) and fix the identifiers and types of purchases.
3. In the mobile application, you need to check whether or not purchasing is available on this device. To do this, use the `InAppPurchases.Purchases are supported()` method.

4. If purchases are supported, then you should update the list of completed and available purchases using the `InAppPurchases.UpdatePurchaseInformation()` method. Without performing this operation, it will be impossible to make a purchase in the future.
   It is recommended that you update the purchase information regularly, as the list of purchases made may not only change on the current device.
5. To determine whether purchases are spent using the 1C:Enterprise language, use the `InAppPurchases.PurchaseConsumingSupported()` method. If the method returns the value `True` it means that the in-app purchases can be spent. Such option is provided under the management of Android and Windows OS.
6. To make a purchase, use the `InAppPurchases.BeginPurchasing()` method. As a parameter of the method, you should pass the ID of the purchase (as it is specified in the corresponding service) or the `InAppPurchase` object, which can be acquired using the `InAppPurchases.GetList()` method.
   After calling the method, control will be passed to the purchasing execution interface, which is provided by the operating system and cannot be redefined at the mobile application level.
   Upon completion of the purchase, control is returned to the mobile application (to the method of processing the notification of the completion of the purchase). One of the parameters of the method will be the purchase status: if the value `True` is passed to the handler, then the purchase is completed successfully. Otherwise, the purchase is not made for any of the reasons: there is no connection with the application store, the user canceled the purchase in the application store interface, the digital signature of the purchase receipt failed validation, etc.
7. Spending of purchases should be done using the `InAppPurchases.ConsumePurchase()` method. After the method is executed, a request is made to the purchasing service, which indicates that the purchase has been spent and is available for re-purchase.

29.3.5.12.3. Verifying an in-app purchase receipt

When making a purchase, a check is made in the mobile application that the purchase has actually been completed. However, the information system (including the mobile application) may perform the check more thoroughly.

`GetIn-AppPurcahsesReceipts()` method (in the mobile device) returns the list of receipts of purchases by the transferred set of purchases IDs. The obtained receipts of purchases can be transferred to the application for a PC to validate the receipts of the in-app purchases with the help of the method `InAppPurchasesValidation.ValidateIn-AppPurchaseReceipt()`. Check is performed completely analogous to a mobile platform. The difference lies in the fact that when checking from an application server for a personal computer, other communication channels are used with the purchasing service (and other environment of the testing system). When checking the purchase receipt, the following actions are performed:

- OS iOS: a request is sent to the Apple server to verify the receipt received. The request is sent to https://buy.itunes.apple.com/verifyRecepient (https://sandbox.itunes.apple.com/verifyRecepient during testing).
- OS Android: the electronic signature of the receipt of the purchase is checked.
- Windows: the electronic signature of the receipt of the purchase is checked. The digital signature validation certificate is obtained from Microsoft web-site (https://go.microsoft.com/fwlink/?LinkId=246509&cid=CertificateID).

At the same time, it is sometimes necessary to validate an in-app purchase receipt on a mobile device. Then, you should use the method `InAppPurchasesValidation.ValiadateIn-AppPurchaseReceiptOnMobileDevice()`.

If the application developer believes that this is not enough, then there is an option to perform additional checks that are based on the contents of the purchase receipt fields. These checks include verification of the uniqueness of the ID of the purchase receipt, the adequacy of the date of purchase, etc. To do this, you can use the object `In-AppPurchaseReceiptData`. This object allows access to the in-app purchase receipt data present in all supported services of in-app purchases and also to the original data of the receipt with the help of `In-AppPurchaseReceiptData.SorceData` attribute. `GetIn-AppPurchaseReceiptData()` method can be used to get the in-app purchase receipt data. This method requires the in-ap purchases receipts array and the relative set of the purchases IDs.

29.3.5.12.4. Setting the in-app purchases services

General information

Information about the features of specific types of purchases and system behavior when operating with purchases should be found in the documentation for the corresponding operating system.

Apple In-App Purchase

IMPORTANT! The description of operating with the web interface is given as of the time of writing the documentation.

1. Sign in to the Member Center (https://developer.apple.com/membercenter/index.action) on behalf of the developer.
2. Go to section Certificates, Identifiers & Profiles.
3. Go to section Identifiers.
4. Register the iOS App ID for the application (if it is not already registered in Member Center). In this case, the registered ID should match the full ID of the mobile application in which you plan to integrate operation with in-app purchases. When creating an iOS App ID, make sure that the In-App Purchase service is enabled.
5. If you plan to use the mobile developer platform to manage with in-app purchases, you should follow these steps:
   1. Go to section Provisioning Profiles.
   2. Create a new provisioning profile.
   3. Specify that this profile is intended for iOS App Development.
   4. Specify as the App ID exactly the iOS App ID that was created in the previous step. It is not allowed to specify iOS Wildcard AppID in this step .
   5. The provisioning profile created in the future will need to be used in the mobile application builder.
6. Sign in to the iTunes Connect (https://itunesconnect.apple.com).
7. Go to section Agreements, taxes and banking.
8. Accept agreement Paid Application. At the same time, it is necessary to indicate information on the bank account to which the earned funds will be transferred.
9. Go to Functions tab.
10. Go to section In-app purchases (open by default).

11. A new in-app purchase is added using the «+» button located to the right of the In-app purchases list header.

Google Play In-App Billing

IMPORTANT! The description of operating with the web interface is given as of the time of writing the documentation.

To create in-app purchases, follow these steps:

1. Enter the Google Play Developer Console (https://play.google.com/apps/publish/).
2. Create or find a mobile application where you plan to operate with in-app purchases. At this stage, you should build a mobile application for which the permission for In-app purchases is set, but there may be no purchasing functionality. Assembly should be performed using the mobile application builder. The assembled application should be downloaded to the application store (in the status of the alpha version).
3. Go to the application page.
4. On the application page, in the Services and API section, remember the value of the LICENSE KEY FOR THIS APPLICATION field. This value will be necessary to perform the check of the in-app purchase receipt on the side of the application for the personal computer.
5. Go to section Content for sale.
6. A new in-app purchase is added using the + Add product button. Be careful when adding products! They can not be removed!

Windows In-App Purchase.

IMPORTANT! The description of operating with the web interface is given as of the time of writing the documentation.

To create in-app purchases, follow these steps:

1. You should log into the Windows Development Center. (https://developer.microsoft.com/windows).
2. It is necessary to go to the information panel (Information panel button).
3. Go to the necessary application page. Find Customization title on the page and press Create customization button. Create new customization new page will be opened.
4. Several types of purchases are presented on the page. The following types suit AC:Enterprise mobile version:
   - Long use -permanent purchase (in 1C:Enterprise terms).

- Managed by the developer spendable -spendable purchase (in 1C:Enterprise terms).
- Subscription (including in terms of 1C:Enterprise).

Insert the unique ID in Product code field after the type selection. This ID will be used for work with the in-app purchases in the configuration code. Then, press Create customization button. The page of just created purchase will be opened.

5. Select Start sending item on the page. The page allowing to fill in the data on the purchase will be opened: Property, Price and availability, Age category, and Description in Store (in different languages).
   The purchase existence time (if it is a permanent purchase) and the type of content supplied with this purchase can be specified Properties section.
   The purchase price and the possibility to search for the purchase in the store or the purchase availability for the purchaser are specified in Price and availability section.
   It is recommended to set the price "free" at first so that nothing is charged for the purchase testing. The desired price needs to be set after testing.
   The purchase description in all supported languages is specified in Description in Store section; a purchase icon can be added if you wish.

29.3.5.12.5. Testing of in-app purchases

General information

This section contains recommendations for testing operation with in-app purchases. Recommendations relate to the features of the interaction of a mobile application (mobile platform) and in-app purchase service.

For OS iOS

General information

Testing of operation with purchases for a mobile application running on OS iOS can be divided into two parts:

1. Testing of the algorithms of operation with purchases, their accuracy and correctness of the response of the mobile application to the purchases made. This testing is performed using the mobile developer platform and the Apple In-App Purchase, which operates in the test mode.
2. Final testing, when it is checked the operation of the mobile application that will be placed in the application store. Purchases will be made

through the real Apple In-App Purchase service, but the actual withdrawals will not be made. For testing, the TestFlight program is used.

Further will be considered in more detail both tests.

On the developer platform

IMPORTANT! The description of operating with the web interface is given as of the time of writing the documentation.

The first step is to perform all operations on pre-setting the Apple In-App Purchase service. When setting up, specify the mobile application ID that will be used for the application being developed.

After making the settings, it is necessary to build a mobile developer platform, specifying the ID of the mobile application that was used to set up the service.

Then specify user accounts that will be able to test in-app purchases in the sandbox. To do this, follow these steps:

1. Sign in to the iTunes Connect (https://itunesconnect.apple.com).
2. Go to section Users and roles.
3. Go to the bookmark Sandbox Testers.
4. On this page, you should add the e-mail of those users who will perform the role of testers.
5. Accounts listed as testers can participate in product testing using the developer platform.

Using TestFlight program

IMPORTANT! The description of operating with the web interface is given as of the time of writing the documentation.

The first step is to perform all operations on pre-setting the Apple In-App Purchase service. When setting up, specify the mobile application ID that will be used for the application being developed.

Then perform the following operations:

1. Sign in to the iTunes Connect (https://itunesconnect.apple.com).
2. Go to section Users and roles.
3. Go to section My applications.
4. Find the application you plan to test.
5. Download the mobile application compiled using the mobile application builder.
6. Go to the properties of the mobile application.
7. Go to TestFlight section.
8. Go to the Internal testing section (open by default).

9. Specify who from internal testers will test the selected mobile application.
10. Accounts listed as testers may participate in product testing.

For OS Android

General information

Purchasing testing for a mobile application running under Android can be divided into two parts:

1. Testing of the algorithms of operation with purchases, their accuracy and correctness of the response of the mobile application to the purchases made. This testing is performed using a mobile developer platform and a mobile application builder (which provides the Google Play In-App Billing emulator).
2. Final testing, when it is checked the operation of the mobile application that will be placed in the application store. Purchases will be made through the real Google Play In-App Billing service, but the actual withdrawals will not be made.

Further will be considered in more detail both tests.

On the developer platform

The developer platform provides the option to perform testing of operation with purchases without the need to build a mobile application. To do this, in the properties of the infobase on which testing will be carried out, fill in the properties Address of server of purchases and User ID.

Tools provided by Google for testing purchases do not allow for this purpose to use the mobile version of "1C:Enterprise" for the developer. In this regard, a service that emulates Google Play In-App Billing is implemented in the mobile application builder.

To use this service, publish on a web server HTTP-service PurchasesTest. The address of the published HTTP service should be specified in the infobase property Purchases server address.

In the builder, you should create a user for which the role of Checking purchases is selected. Then, in the mobile application builder, you should open a tool for testing in-app purchases (Desktop - Service - Testing in-app purchases).

In the Mobile application field, select the mobile application for which the in-app purchases will be tested. The contents of the remaining tables are associated with the selected application.

In the Test users table, you can specify several users on whose behalf the execution of in-built purchases will be tested. Specifying different identifiers of test users in the properties of the infobase on a mobile device (User ID property) allows to quickly check the operation with purchases on behalf of different users operating with the mobile application.
In the Available purchases table, you need to create exactly the purchasing list (especially in terms of identifiers and types of purchases), which will later be created in the Google Play Developer Console. When specifying the purchase currency, use the alphabetic currency code from the standard ISO 4217 (http://www.iso.org/iso/ru/home/standards/currency_codes.htm).
In the Complete purchases table, there is a feature to create the necessary configuration of the test environment: what user, what purchases made, there is an option to disable the purchase acquiring, etc.

Assembled application

IMPORTANT! The description of operating with the web interface is given as of the time of writing the documentation.

The first step is to perform all operations for pre-setting the Google Play In-App Billing service. When setting up, specify the mobile application ID that will be used for the application being developed.

1. Enter the Google Play Developer Console (https://play.google.com/apps/publish/).
2. Go to section Settings (on the left of the screen).
3. In the Testers' Gmail Accounts section, specify the email addresses of those application developers who will test operation with in-app purchases. The developers (or testers) listed here will be able to make purchases without making a real payment, however, when making a test purchase, specify the data of a real, solvent bank card. A test debit (with subsequent return) of a small amount (approximately 1 USD) will be made from this card to verify the correctness of the specified data.
4. Accounts listed as testers may participate in product testing.

On Windows

On the developer platform

The developer platform provides the option to perform testing of operation with purchases without the need to build a mobile application. To do this, in the properties of the infobase on which testing will be carried out, fill in the properties Address of server of purchases and User ID.

The mobile purchase testing is analogous to such testing for the application operating under Android OS management.

29.3.5.13. Internet connection characteristics

When using a mobile application, tasks of exchanging information with an external system via the Internet often arise. In this case, the task of exchanging only through certain types of Internet connection may arise, or only if the Internet connection has sufficient bandwidth (speed) to perform the exchange.

Mobile version of "1C:Enterprise" allows to determine the parameters of the Internet connection, which is currently used on a mobile device. You can get these parameters using the `InternetConnectionInformation` global context property.

To determine the type of Internet connection, the method `InternetConnectionInformation.GetConnectionType()` is used. The system recognizes the following types of Internet connection: Mobile Internet connection (GRPS, EDGE, etc.), WiFi connection, local area network connection and no Internet connection.

Using the `InternetConnectionInformation.GetExpectedSpeedOfInternetConnection()` method, you can estimate the expected speed of the current connection. If the method returns a value of 0, this means that Internet access is absent on the mobile device at the moment.

There is an option, using the `InternetConnectionInformation.GetRoamingUsage()` method, to find out that the device is roaming when the method is called. Being roaming may mean that the billing of the downloaded data using a mobile data network significantly (dozens of times!) exceeds that in a home network. If the function returns `Unknown` value, this means that the type of connection used is not equal to the value `InternetConnectionType.CellularData`.

An important feature of the tools for operating with Internet connection in the mobile version is the capability to attach a handler that will operate in the following cases:

1. When changing type of Internet connection. This case also includes the situation when any access to the Internet on the mobile device is turned off, for example, the user has put the device into “flight mode”.
2. When changing the expected speed of the Internet connection when connecting via mobile Internet. In this case, changes in the used mobile cellular standard are recorded, for example, switching from a 2G standard network to a 3G standard network, etc.

3. When the roaming state changes for the current mobile device.

To connect (or disconnect) the Internet connection change handler, you should use methods `AttachInternetConnectionChangeHandler()` (`DetachInternetConnectionChangeHandler()`) of the `InternetConnectionInformation object.`

29.3.5.14. Additional user check

29.3.5.14.1. General information

So far as an application installed on a mobile device is concerned, a need may arise to verify whether a device is currently used by its owner, or not. For instance, a function is used which is unaccessible to all users granted access to unlocked mobile device. Moreover, you may need to store confidential data in the mobile device memory, including authorization details (username and password) required to access a remote service.

As such, an application needs to perform user authentication in any way - (password, graphic key, fingerprint etc). This authentication can be performed using `AdditionalUserValidation` and `SecureStorage` global context objects. For more information concerning these properties, see below.

So far as additional user validation is concerned, it should be noted that such verification is not deemed a method to authenticate a user. Thus, upon completion of additional user validation it cannot be definitely said whether a device is used by a specific user, or not. It can be asserted only that a user currently using this device knows how a telephone can be unlocked. In other words, a telephone owner can be identified (with due regard to the said restrictions). For instance, if you grant your business partner access to details which may be used to unlock your device (PIN code or unlock pattern), telephone software will not be able to identify exactly the user of your device: you or your business partner.

29.3.5.14.2. Additional user validation

`AdditionalUserValidation` global context property is intended to procure that an application installed on a mobile device runs an additional user validation process to verify that a mobile device (and mobile application) is currently used by its legal owner. `AdditionalUserValidation` grants access to `AdditionalUserValidationManager`. To be granted access to any manager property or method, use

`AdditionalUserValidation.Method()`. So far as method description is concerned, to make it easier `AdditionalUserValidation` prefix is ignored. It is evident that such omissions are unacceptable in examples and real applications.

Before validation process is started, make sure that your device supports an additional validation feature. To do that, select `ValidationSupported()`. An additional validation method required to be checked is used as a parameter. If your device supports this validation process, -you can run user validation (identification).

To identify a user, select `StartValidation()`. This method accepts an additional validation method and details of a handler to be used as soon as validation is over and moreover a text message displayed by the platform to a user. The said text has to describe why and for what purpose respective actions are to be performed by a user. Obviously, validation method specified in `StartValidation()` method has to be similar to that used to determine, whether an additional validation is supported by the device, or not.

If a mobile device supports biometric user identification, to know a method currently used, select `CurrentBiometricIdentificationMethod()`. The resulting value can be used for different purposes, for instance:

- to generate a message which is used in `StartValidation()` method.
- To make sure that reliability of a validation method so used is in compliance with applicable requirements defined by the application developer.

Generally, for an example of additional mobile application user validation, see below:

```
ValidateUser() procedure
  ValidationMethod = AdditionalUserValidationMethod.BiometricOnly;
  If not
AdditionalUserValidation.ValidationSupported(ValidationMethod),
    Return;
  EndIf;

  CurrentBiometricMethod =
AdditionalUserValidation.CurrentBiometricValidationMethod();
  If CurrentBiometricMethod <>
BiometricValidationMethod.FPIdentification,
    Return;
  EndIf;

  Message = "Scan your fingerprint";
  Handler = New NotificationDetails();
  AdditionalUserValidation.StartValidation(ValidationMethod,Message
, Handler);
EndProcedure

ValidationResultsProcessing(AbortedByUser, AdditionalParameters)
Export procedure
  If not AbortedByUser,
    // user validated
  EndIf;
EndProcedure
```

29.3.5.14.3. Secure storage

`SecureStorage` global context property is intended to enable an application installed on a mobile device to access a secure storage where confidential information is saved. `SecureStorage` grants access to `SecureStorageManager`. To be granted access to any manager property or method, use `SecureStorage.Method()`. So far as method description is concerned, to make it easier `SecureStorage` prefix is ignored. It is evident that such omissions are unacceptable in examples and real applications.

Secure storage stores data which can be accessed if you have a proper text key. For application purposes, the said key is deemed a unique value. Storage can use data of any kind which supports XDTO-serialization. `Undefined` value cannot be put in the secure storage. Whenever any data is put in the storage, secure storage access restriction method is set up (one of `MethodToProtectAccessToSecureStorage` system enumeration values):

- `ScreenUnlockRequired`. In this scenario, it is assumed that an application can access a secure storage, if a user unlocks any relevant device (access to the device desktop is granted).

- `AdditionalUserAuthenticationRequired`. As such, to be granted access to secure storage, a user has to undergo additional validation. Additional validation method is set up using the second parameter of `AccessProtectionSupported()` method (one of `AdditionalUserValidationMethod` enumeration values).
- `No`. As such, to be granted access to secure storage, no additional validation is required. Mobile operating system guarantees that access to secure storage is granted, only when a mobile device is successfully started.

The said setting defines the following behavior:

- To be granted access to a value, you need to use a validation method specified when the said value is set.
- If a proper access protection method supported by a mobile device is disabled, -data put in the secure storage with disabled protection method assigned thereto will be deleted from the storage, and thus further it remains unaccessible.

Secure storage has a number of distinctive features with their content depending on the mobile operating system used:

- OS Android:
  - Storage size: unrestricted.
  - Single element size: unrestricted.
  - Number of storage elements: unrestricted.
- macOS:
  - Storage size: 80Mb
  - Single element size: 16Mb
  - Number of storage elements: unrestricted.
- Windows OS:
  - Storage size: 320Kb
  - Single element size: 16Kb
  - Number of storage elements: 20 pcs

Before using secure storage, make sure that your device supports this functionality. To do that, select `AccessProtectionSupported()`. Storage access protection and additional user validation methods are used as parameters.

If your mobile device supports secure storage functionality, an application can put in and retrieve data from that storage. A standard set of operations is supported:

- Put value in storage. Use `StartPutValue()`.
- Get value from storage. Use `StartGetValue()`.
- Delete value from storage. Use `StartDeleteValue()`.

- Make sure that data with the key specified is available in the storage. Use `VerifyKey()`.

No key search in a secure storage is supported.

Let's take a closer look at storage functionality. When you put data in storage, in addition to a key and data so saved you need to define the way for the said data to be protected in. To do that, use `AccessProtectionMethod` and `VerificationMethod` parameters of `StartPutData()` method. `AccessProtectionMethod` parameter is assigned `SecureStorageAccessProtectionMethod` kind. For its distinctive features, see above. If `AccessProtectionMethod` is set to `SecureStorageAccessProtectionMethod.AdditionalUserValidationRequired`, specify `ValidationMethod` parameter value. This parameter is assigned `AdditionalUserValidationMethod` system enumeration kind and enables you to define the way additional user validation is performed in.

To get data (`StartGetData()`), along with a data access key specify a text message displayed to a user when additional validation is required. A user can refuse to get data from secure storage. So far as an application is concerned, this is similar to unavailability of access to specific data. An application needs to correctly respond to a situation when data cannot be retrieved from secure storage.

As a rule, methods used to manage storage are not intended to lock application. The result of method operation is delivered to a proper callback handler by one of method parameters when the said method is called. Whenever an error is identified, management is ensured by an error handler which needs to be defined when you create `NotificationDetails` object.

A method which is intended to verify availability of a certain value for the specified key in the storage (`VerifyKey()` method) is the only synchronous method.

Moreover, note that methods used for validation purposes when secure storage is accessed do not guarantee that any specific user is available. The said methods guarantee that a user is able to unlock a telephone using any relevant method. For instance, enter a PIN code or unlock pattern to be able to use a telephone.

29.3.5.15. Backup by mobile operating system

Mobile operating systems support embedded data backup tools for applications installed on mobile devices. Depending on the operating system, the following services are used for backup purposes:

- OS Android: Google Drive or dedicated backup applications supported by a personal computer.
- OS iOS: iCloud or iTunes-supported backup.
- Windows OS: Microsoft OneDrive.

For a mobile device to be able to backup data using operating system tools, select Backup by OS in a list of properties (Required Mobile Application Permissions). If backup by mobile OS tools needs to be disabled for an application, - clear Backup by OS tools check box.
No further actions are required. This permission is only applicable to a mobile application which is built using Mobile Application Builder.

29.3.5.16. Other mobile device features

Flashlight

When using a mobile application, it may be necessary to highlight an object. In that case, if the mobile device used at this moment has a flash of the main camera, then you can use this flash as a flashlight.
To verify that this feature is supported by the device, use the `DeviceTools.FlashlightSupported()` method. If the method returned `True`, then you can turn on the flashlight by calling the `DeviceTools.TurnFlashlightOn(True)` method. Turning off the flashlight is performed by calling the `DeviceTools.TurnFlashlightOn(False)` method.

Installing applications

NOTE. Applicable to mobile applications running under Android OS only.

A mobile application can install various third party applications on a mobile device which are required to operate a mobile application so built. This feature can violate safety regulations applicable to mobile applications.
Installation of third party applications is managed by Application Installation permission. This permission is only applicable to a mobile application which is built using Mobile Application Builder.

29.3.6. Add-ins

In a mobile application only add-ins developed with Native API technologies that can be separate files as well as appropriate ZIP archives are used.
In add-in designed for use with a mobile application, it is prohibited to use the user interface.

The add-in component must be compiled taking into account all processors, architecture and operating systems used on mobile devices. It is recommended that the functional modules of the add-in will be implemented in a platform-independent manner.

If the add-in uses additional modules, it is recommended to indicate this in the documentation for the component. Used non-system libraries should be statically included in the resulting file of the add-in.

Location of the add-ins depends on the client application used and the mobile operating system:

- OS iOS:
  - Mobile application:
    - Mobile application for developer - add-ins are downloaded through a web server on which the debugged configuration is published.
    - Mobile application for application store - add-ins are included in the assembled mobile application.
  - Mobile client:
    - Mobile client for developer - add-ins are downloaded through a web server on which the infobase is published.
    - Mobile client for application store - add-ins are included in the assembled mobile application.
- OS Android:
    - Mobile application for developer - add-ins are downloaded through a web server on which the debugged configuration is published.
    - Mobile application for application store - add-ins are included in the assembled mobile application.
  - Mobile client:
    - Mobile client for developer - add-ins are downloaded through a web server on which the infobase is published.
    - Mobile client for application store - add-ins are included in the assembled mobile application, however, they can be downloaded via a web server on which the infobase is published if a new version of the add-ins is loaded into the infobase.
- Windows OS:
  - Add-ins are included in the assembled mobile application in any case.

When managing mobile client, the following feature should be considered: If the mobile client (in the version for the application store) allows to operate simultaneously with information databases that use different versions of the same add-in, this may cause a problem. The problem is that the assembled

mobile application may contain add-ins of only one version (for one add-in). In this regard, it is recommended to develop add-ins with unconditional compliance with bottom-up compatibility, so that any add-in of N+1 version supports operability with an application that only expects version of add-in of version N. At the same time, running under Android OS will not lead to such problems, since the mobile client will automatically download the version of the add-in that is used in a specific infobase.
Debugging of the application, using add-ins, is different for different mobile OS:

- For OS iOS, - debugging is possible using the mobile version of "1C:Enterprise" for the developer by publishing an application or an infobase on a web server.
- For Android OS, - debugging is possible using the mobile version of "1C:Enterprise" for the developer by publishing an application or an infobase on a web server or when operating through the Android Debug Bridge.
- For Windows OS, - it is permissible to use only the assembled mobile application (using the mobile application builder).

In order to use add-in with a mobile application, you need to load it into the configuration in the template with the Add-in type. In addition, if the application will only run under Android OS, add-ins can be loaded from the template with the Binary data type, directory attribute, temporary storage and file system object.
In order to build a mobile application with the inclusion of add-ins, it is necessary to export the configuration in the form of a zip-archive (1cema.zip). If the configuration contains add-ins, then when executing the command Main menu- Configuration - Mobile application - Save in file or Main menu - Configuration - Mobile client - Save in file, this saving option will automatically be offered. The mobile application builder will automatically consider the presence of add-ins when loading the mobile configuration and then assembling the mobile application.

29.4. Application on the mobile platform

29.4.1. General information

This section contains information that should be considered when developing applications on a mobile platform.

29.4.2. Types of applications on the mobile platform

Application development on a mobile platform can be divided into the following stages:

- The decision is made on how the application will look like on the mobile platform:
  - Independent of any applications on the remote system. In this case, the data structure of the application on the mobile platform is determined only by the scope of the application. Most likely, no data exchange is required.
  - Autonomous workplace of some application located on a remote system. In this case, the data structure is mainly determined by the application, but there will be changes that are determined by both the scope and the features of the application on the mobile platform. The data exchange protocol is likely to be developed specifically for the exchange of a specific application with a specific application on a mobile platform. Most likely, this protocol will not support the exchange with other applications.
  - Universal autonomous workplace. In this case, the data structure should consider the specifics of all applications with which the application should interact on the mobile platform. You must also consider the features of the application on the mobile platform. The exchange protocol should be sufficiently universal and standardized.
- It clarifies whether the developed application should run only on a mobile device or it can also run on a personal computer (for example, on a laptop).
- The application is being implemented on a mobile platform and is being debugged on a personal computer (including data exchange mechanisms). At the same time, it is necessary to understand that the mobile platform has a number of differences and peculiarities, due to which it is impossible to completely test the performance of the developed application on a personal computer. However, the code in the 1C:Enterprise language can be debugged using a personal computer.
- On the mobile device (one or several) final check and debugging of the interface and application functionality is performed.

TIP. It is recommended to choose to develop an option of an autonomous workplace that can function on both a mobile device and a personal computer.

When developing a data exchange mechanism, it is necessary to consider that applications with different versions of the exchange interface should exchange data. This should not cause problems in exchange. This is due to the fact that updating the application at once for the entire infrastructure is almost impossible. It should also be noted that data synchronization with an outdated version of the exchange protocol can be used to perform backup operations before updating the mobile application configuration.

When developing an application on a mobile platform, it is necessary to consider the restrictions imposed by the mobile platform in comparison with the 1C:Enterprise platform for a personal computer:

- Limited list of available configuration objects and mechanisms;
- A limited set of attributes properties;
- A limited set of elements of managed forms;
- Simplified implementation of some mechanisms (for example, access rights or home page);
- Absence of the ordinary forms and the usual start mode;
- Lack of tools for debugging a mobile application;

It should also be borne in mind that the mobile platform and the platform for a personal computer, when the versions coincide, differ in the functionality provided. The following is a functional correspondence between the versions of the platform for the mobile device and personal computer:

Mobile platform	**Platform for personal computer**
8.3.12	8.3.12
8.3.11	8.3.11
8.3.10	8.3.10
8.3.9	8.3.9
8.3.8	8.3.7
8.3.7	8.3.6
8.3.6	8.3.5
8.3.5	8.3.4
8.3.4	8.3.2
8.3.3	8.3.2

In the documentation that is being developed for an application on a mobile platform, the application developer may not describe the behavior of the standard mechanisms of the system, but refer to the Mobile Application User Guide, which is located at http://v8.1c.ru/mobile_user_guide/.

29.4.3. Specifications

29.4.3.1. Reports

Reporting on a mobile platform is possible in two different ways:

- Processing of existing data and the formation of the actual report in the 1C:Enterprise language;
- An implementation on a mobile platform of some form, with which you can set report generation parameters (filters, etc.), and pass the formation itself to a remote system, which will return a spreadsheet document for display on a mobile device.

The choice of one of these options depends on a certain set of factors:

- Whether the infobase of the mobile application contains all the data that is needed to build the report;
- Availability and quality of the communication channel between the mobile device and the remote system;
- What is the complexity of the report;
- How the exchange is built with the remote system;
- How to deal with the report;
- etc.

If you decide to generate a report on a mobile device, you should remember about the features of the mobile platform. Samples are available for obtaining data, and software for generating a spreadsheet document (based on the template) and charts (if required) are available for building the report.

If permanent high-speed communication channels with a remote system are available for the mobile application, which has all the necessary data to generate the required report, then the implementation of only the interface part of the report (assignment of selections, processing of decryption, etc.) on the mobile application is possible. Actually the report will be generated in this case in the remote application, and the finished report will be returned for display on the device.

In this case, all intermediate options are possible, for example, if a periodic exchange is implemented, you can generate report settings on a mobile device and pass to a remote system during an exchange session, and get the generated report back for display.

To interact with a remote application, you can use a Web-service (which provides the remote application), exchange data using an exchange plan, and so on.

If it is necessary to implement decryption of a report, you need to understand that a mobile application alone can only perform a review of the values spec-

ified as a decryption of a report cell. The remaining actions should be implemented in the 1C:Enterprise language, including the receipt of a new option of the report, taking into account the imposed filters, etc.

29.4.3.2. Operation with roles and users

When operating with roles in a mobile application, remember that of all the access rights that the platform provides for a personal computer, only a limited set is available on the mobile platform. This set of rights is intended to ensure the functioning of various mechanisms that provide the interface of a mobile application. The rights that can be used on the mobile platform are marked with a «*» in the full list of rights.
There is also a feature to perform role-setting form, set up the role visibility of the command interface.

IMPORTANT! Rights to access data (`Read`, `Add`, `Change`, `Delete` etc.) are not supported by the mobile platform.

Privileged mode management methods in the mobile platform are used for compatibility with the personal computer platform. The calls of the methods for operating with the privileged mode of the mobile platform are ignored and the mobile platform does not perform any actions.
When defining user rights, the standard scheme is valid: if no users are specified in the infobase, then the `DefaultRoles` configuration property is used to determine the access rights. If a user is used to access mobile application data, access rights are determined by the components of the user's roles.
The mobile platform has a limited set of tools for operation with infobase users:

- Only one user can exist in the infobase. Attempting to create more than one user generates an exception.
- The user can only be created using the program code. The interactive creation of users in the configurator will not lead to the transfer of the user to the mobile device.
- Mobile platform does not offer authentication tools when starting a mobile application. If the user is listed in the user list, this user will automatically be set as the user of the current session.

As a result, we can recommend the following scheme of operation on behalf of users with different rights:

1. The user is created with the required set of roles, or the set of roles of the existing user is changed.
2. The mobile application is restarted using the construct `Exit(False, True);`.

29.4.3.3. Using full-text search

29.5. "1C:Enterprise" Mobile client

29.5.1. General information

This section contains information that should be considered when developing applications running in a mobile client.
It is commonly known that an application on a mobile platform contains a copy of the used configuration and operates only with it. At the same time, the mobile client should operate with the version of the configuration that is currently in the infobase to which the mobile client is attached. However, application stores may require that an application published in a store does not change its functionality after installation on a user's device. In a sense, these requirements are mutually exclusive.
In order for the mobile client application to be published in the application store, several requirements must be met:

1. The mobile client can operate only with those configurations that are specified before assembly. One mobile application can manage several information bases.
2. Each configuration has its own electronic signature (including the configuration metadata characteristic), due to which there is no option of a complete replacement of the configuration in the infobase.

Compliance with these requirements allows the assembled mobile client application to be published in the application store and meet the requirements of this store.
The mobile client interacts with the infobase via HTTP/HTTPS. This means that the mobile client does not support direct connection and can operate only with the information bases that are published on the web-server. When operating with the mobile client, it is not required to ensure exact correspondence between the mobile client version and the web-server or 1C:Enterprise server extension version. Compatibility is provided not within the version number of the 1C:Enterprise system, but within the version of the interaction protocol. In the event that the interaction protocol is substantially changed, it will be necessary to update the thin client application on mobile devices.
Unlike an application on a mobile platform, which, in fact, is individual and separate application, an application for a mobile client is an adaptation of an ordinary application that runs in a thin client or a web-client. But for the correct operation of this application in the mobile client, the platform needs to

“prompt” how to build the interface in the mobile client. The properties of the form elements are intended for this.

29.5.2. Preparing the configuration for operation in the mobile client

In order that configuration will be able to operate in a mobile client, you need to perform some set of actions:

- Set the Compatibility mode configuration property in the Version 8.3.7 value and later.
- Perform adaptation of configuration forms, taking into account the peculiarities of displaying these forms on mobile devices.
- In the modules with 1C:Enterprise language, use the compilation directive `# If NOT a MobileClient Then ... # EndIf` everywhere where you want to mark the code that can only operate on a mobile device.
- Generate a digital signature for the configuration.

29.5.3. Digital signature and configuration digest

As already mentioned, the mobile client application cannot arbitrarily change its functionality after downloading to the user's mobile device. Such requirement is imposed by the application stores on the mobile applications which download their code from the external resources. To comply with this requirement, each configuration which can operate in the mobile application contains some auxiliary information which allows tracking a considerable change in the configuration.
The scheme of operation is as follows:

- At the beginning of the development of the configuration (in the configurator) a private key is generated to form the signature of the configuration being developed.

IMPORTANT! Each configuration **must** have a unique private key. Using the same private key for multiple configurations - is not allowed.

- Configuration development in progress.

- If it is necessary to publish the configuration, a configuration signature is generated in the mobile client. When performing this action, a configuration digest is created, which is signed by the private key.
- When uploading the configuration for the mobile application builder, the public key (for the private key use) is placed in the file, with which you can verify the digital signature.
- The builder places the public key in the assembled mobile client application.
- When attempting to connect a mobile client to an infobase, the 1C:Enterprise server passes a digital signature of the configuration and the current configuration digest to the mobile client.
- A mobile client is connected, only if configuration name (Name configuration property) is not modified from the date when mobile application is built and a digital signature used for that purpose corresponds to the digest so transferred. This can be verified because the mobile application contains the public key with which the verification is performed.

IMPORTANT! The private key is confidential information.

When you use iOS mobile application home page to add an information base, the compatibility check is based on the digital signatures of all configurations included in the mobile client application build.

In the event that it is necessary for the platform to automatically check the configuration and the created digital signature, open the dialog box Main menu -Configuration - Mobile client - Setting the use of the mobile client. In the dialog box, select the Check mobile client signature check box when updating the database configuration and click the OK button. After that, when the Update database configuration command is executed, the compliance of the saved configuration and the existing digital signature will be checked.

The configuration digest is designed to test the feature of the mobile client to operate with the infobase to which it is attached. The configuration digest includes the following information:

- The name and internal ID of the following configuration objects:
  - Catalogues;
  - Documents;
  - Charts of characteristic types;
  - Exchange plans;
  - Enumerations;
  - Business-processes.

- Names and components of the write keys for the following configuration objects:
  - Information register;
  - Accumulation registers;
  - Accounting registers;
  - Calculation registers.
- The extensions used by the configuration are not included in the digest components.

A change in the configuration objects included in the configuration digest means the necessity to resign the developed configuration.

Thus, the actions necessary in the case of the configuration change are divided depending on the further developer of the mobile client configuration:

- Configuration author: in the case of the configuration digest change, it is necessary to resign the configuration without changing the published mobile client application. The mobile client application does not need to be republished in this case.
- Other experts without access to the configuration private key. One of the following actions can be performed in this case:
  - To ask the original configuration developer to resign your copy of further developed configuration.
  - To build (with the help of the mobile application builder) and publish your copy of the mobile application intended for the further developed configuration exploitation. The opportunity to work with the original configuration can be saved in this application. Do not forget that, in this case, the publication in the mobile applications store will be performed on behalf of the company which has further developed the configuration with all consequences (registration as the developer in the application store, certificates, etc.).

Thus, the use of a new private key to sign a configuration automatically means that such configuration can be used only with the mobile application which has the public key for the digest validation. So, the mobile application shall be updated before the undated configuration becomes accessible in the used information base.

29.5.4. Operation with data associated with the current row of tables

In the event that there is data on the form associated with the current row of the table (one or more), then it is recommended to do the following:

- All form data associated with the current table data should be divided into groups so that one group displays data associated with the current row of only one table.
- For each such group, you should set the properties Usage of current row (`CurrentRowUse`) and Used Table (`AssociatedTable`) by analogy with the command form. It follows from this that one group can display data of only one table. That is why the above is a recommendation to place related data in different groups by the number of tables used.
- For tables that are used as data sources for ancillary data, set Usage of current row property to `Selection` value.

In the mobile client on the form there will be no groups for which the Use of the current row property is set to the `Use` value.
In order to see the associated data, you need to open the context menu on the selected row and select a command, the choice of which will open the data window. The team name is generated as the name of the hidden group. If in this table the current data is displayed in several groups, - the name of the command will be generated from the headings of all groups, combined by the symbol «,».

29.6. Offline mobile client

29.6.1. General information

This section deals with specific aspects of development of a configuration intended to be installed in an offline mobile client.
So far as an offline mobile client is concerned, the following terms will be used:

- **Main server** - is an application sever cluster used by an offline mobile client. The said cluster is installed on a personal computer.
- **Standalone server** - is an offline mobile client server installed on a mobile device.
- **Offline mode** - means a mobile application mode, when there is no connection to the main infobase.
- **Conventional mode** - means that a mobile application can use both main and standalone server.
- **Standalone configuration** - is part of main configuration intended to be installed on a mobile device and used in offline mode.

- **External links** - means links to infobase data with their kinds not included in a standalone configuration, while the said data - is used in standalone configuration objects.

Mobile client is operable, only if a mobile device is connected to a web server where an application is published. Application installed on a mobile platform requires no external (vis-a-vis a mobile device) database. It interacts with a database installed on a mobile device. An offline mobile client combines these two features. If a mobile client is connected to an infobase: an offline mobile client operates in the conventional mode. If this connection is interrupted: a mobile application switches to offline mode and uses only data (and configuration) defined for the offline mode.

Availability of data and forms in the offline mode is defined when you set up an offline configuration. Moreover, in addition to a list of objects and forms available in the offline mode data is defined (attributes and tabular sections) which will be present in standalone objects.

So far as forms being part of a standalone configuration are concerned, you can respond to a situation when main server availability changes. As such, developers are able to adapt forms on a mobile device in view of changing conditions: i.e. availability or unavailability of the main server.

Use exchange plans for initial filling of offline mobile client database.

Generally, when an offline mobile client is developed, use the following basic scenarios as reference:

- Conventional mode
  - So far as server calls are concerned, you can use either a standalone or main server.
  - Standalone configuration objects are used in accordance with priorities defined during standalone configuration setup process.
  - Whenever connection to infobase is interrupted, it switches to the offline mode. Further:
    - An offline mobile client attempts to reconnect to the main infobase.
    - As such, an infobase is used with its content defined when setting up a standalone configuration content, while the volume of data- depends on exchange plan parameters with accuracy of recent synchronization.
    - Whenever deemed necessary, an application has to reopen an application form.
    - Features which are not made part of a standalone configuration, -are unavailable.
  - As soon as connection is reestablished, an offline mobile client has to synchronize data and (if necessary) reopen application forms.

- Offline mode.
  - If is enabled forcefully in the dialog box with mobile application infobase properties or dialog box with user properties.
  - In this mode, an offline mobile client makes no attempts to reconnect to the main infobase by default.
- Bad connection with the main infobase.
  - If is enabled forcefully in the dialog box with mobile application infobase properties or dialog box with user properties.
  - This mode is somewhat similar to the conventional mode. However, objects being part of a standalone configuration use a standalone server (irrespective of a priority assigned), while other objects use the main server.
  - An offline mobile client attempts to reconnect to the main infobase. As soon as connection is reestablished, objects not being part of a standalone configuration become available.

Depending on assigned priorities and availability of connection to the main database, a server to be used to perform operations on the mobile device can be defined as follows:

Priority/Connection	**Connection established**	**Connection lost**
Offline	Standalone server	Standalone server
Main	Main server	Standalone server
Not being part of a standalone configuration	Main server	Not available

For more information about distinctive features of development of an offline mobile client specified herein, see below.

29.6.2. Standalone configuration content

To define content of a standalone configuration, use Standalone Configuration Content property. This property is common for the entire configuration. Press Open to open content setup dialog box.

You can define whether the following objects are available in the standalone configuration, or not:

- Configuration objects. A standalone configuration can include all configuration objects supported by 1C:Enterprise mobile platform.

- Configuration object attributes, tabular sections and tabular section attributes.
- Configuration object forms, commands and layout.

Priority can be assigned to a standalone configuration object as follows:

- Generally, with respect to the following configuration objects:
  - Common modules.
  - Exchange plans.
  - Event subscriptions.
  - Common forms.
  - Common commands.
  - Constants.
  - Catalogs.
  - Documents.
  - Document journals.
  - Enumerations.
  - Reports.
  - Data processors.
  - Charts of characteristic types.
  - Information registers.
  - Accumulation registers.
- For configuration object forms.
- For configuration object commands.

Priority can be assigned in the similarly-named column and have the following values:

- Main server - in this scenario 1C:Enterprise uses its best efforts to access an object using the system main server.
- Standalone server - in this scenario 1C:Enterprise uses its best efforts to access a configuration object using the system standalone server.
- Auto - this value is interpreted as Main server for type-defining configuration objects and as a value of a type-defining object -, if an object which is subordinate to a type-defining object is described.

If so far as a configuration object (for instance, Goods) is assigned Standalone server priority, while for this catalog element form Auto priority is assigned, on a mobile device Standalone server priority is used for a form element. If priority value for Goods catalog is changed for Main server, -the form priority is modified by default.

If a metadata set defined for a standalone configuration is incomplete, in an offline mode this can cause conflicts. The said conflicts are settled as follows:

- If missing metadata is used in standalone configuration objects, instead of links to the said objects external links are used in the following standalone configuration areas:
  - Configuration object attributes.
  - Common attributes.
  - `Owner` standard attribute.
  - Constants.
- If references to missing configuration objects are used in other system mechanisms, they will be deleted from the following:
  - Configuration and subsystem command interface.
  - Subsystem components.
  - Access rights role components.
  - Exchange plan components.
  - Selection criterium kind and components.
  - Catalog and documents characteristics.
  - Register records.
  - Document journal components.

In addition to manual selection of objects included in the standalone configuration, the following commands are available in the dialog box intended to manage standalone configuration components:

- Include all - makes all main configuration objects part of a standalone configuration.
- Exclude all -makes all objects unavailable in a standalone configuration.
- Find connected -this command is available for a main configuration object made part of a standalone configuration. It selects main configuration objects used as attribute kinds in the configuration object so selected. As soon as it is executed, a dialog box is displayed listing all proper objects so selected. If the said list is correct, - press OK to add it to a standalone configuration.
  If the foregoing command is executed recursively, both configuration objects used directly in the current object and those available in the objects being used etc will be selected. If the foregoing command is not executed recursively, objects used in an object for which this command is executed will be selected only.
- Find connected for all - this command is available, only if there is at least one object included in a standalone configuration. In this scenario, per each object in a standalone configuration Find connected command is executed. As a result, a dialog box is displayed which lists all selected objects.

29.6.3. Defining a server during a call

One of distinctive features of an offline mobile client is that you can use two application servers at a time: standalone and main servers. When a call is made, system analyzes certain parameters and based upon results so obtained selects a server to be used. As such, priority of use of any specific object is analyzed and moreover selected modes of operation of a client application. The said analysis is performed both for an explicit (for instance, when common module server method is called) and implicit server call (for instance, when a form is opened).

If a server call is initiated on the client application side, server is selected when a call is made. If further server calls are made inside the said server call, sever is not changed. Let's consider some examples in support of the afore-listed assertions.

For instance, there is `Goods` catalog form with Standalone server assigned as its priority use. There is `CM1` common server module with Main server priority. There is `CM2` server module, which is not a standalone configuration component.

Let's see which server will be used when various actions are performed:

- A client application opens `Goods` catalog form. As this form is assigned Standalone server priority, to open a form, a standalone server is used. Data to be displayed by that form is received from the standalone server, too.
- If `Goods` catalog element form calls **any** (context or out-of-context) form module server method, - it will be executed on a standalone server side.
- If `Goods` catalog element form calls `CM1` common module method from the client method of form module, main server is called or an exception is generated. A call is executed, if an offline mobile application is connected to the main server, since `CM1` common module is assigned Main server priority.
- If `CM1` common module method is called from the server method of `Goods` catalog element form, a standalone server will be used. This is due to the fact that a server call made from a server call uses a server which is selected for the first server call (i.e. standalone server in the above example).
- If in `Goods` catalog element form a client method attempts to call `CM2` common module method, similarly to `CM1` common module a call will

be made, only if connection is established. This is due the fact that `CM2` common module is not included in the standalone configuration.

Notwithstanding the above, a situation may arise when you will need to change a server used to make a call. 1C:Enterprise allows you to do that upon any of the following:

- You need to call a server method of the main server from a method which is executed on a standalone server side. As such, use `MainServer` global context property. If this property is used, common server modules of the main sever are accessible on a standalone server. In this scenario, a call looks as follows: `MainServer.CommonServerModuleName.CommonModuleMethodName()`.
- On the client application side, you need to make a server call which uses either the main or standalone server irrespective of current parameters. For that purpose, use `SetUsedMainServer()` global context method. Moreover, it can be useful to benefit from `MainServerAccessible()` method, which allows it to know, whether the main server is accessible, or not.
- A client application form has an attribute with its kind unavailable in a standalone configuration. In this scenario, when you open a selection (or element) form, the main server is used irrespective of the server used to open a form where the said attribute is displayed.

29.6.4. Offline mobile application form

When an application form is developed which is intended to be operable on an offline mobile client, note common distinctive features of that form when used on a mobile device and on an offline mobile client.

These distinctive features include:

- Changes in the form element behavior when the main server is unavailable.
- A form reopening feature, if the main server availability changes.

If the main server availability changes when an open form is used, system behavior will depend on a server selected to open the said form:

- A form is opened from the main server:
  - The form becomes unavailable.
  - `IfMainServerAvailabilityChanges` form event is called.

    - Close remains accessible for the form. If a form has a similar form in a standalone configuration, - Open offline command is accessible.
- A form is opened from the standalone server:
    - Elements behavior is defined by `BehaviorWhenMainServerIsInaccessible` form element property value. The said property can take the following values:
        - `MakeInaccessible` - elements with this property value are disabled as soon as the main server becomes inaccessible.
        - `DoNotChangeBehavior` - elements with this property value remain accessible. So far as elements related to an external reference are concerned, when any of them is called behavior will be defined by a standalone configuration. Standard selection processing will return an error. However, standard data processor can be redefined.
        - `Auto` - is interpreted as follows:
            - So far as elements related to an external reference are concerned -, similarly to `MakeInaccessible`.
            - So far as other elements are concerned, -similarly to `DoNotChangeBehavior`.

If in a form interface Open offline button is pressed (or `ReopenFromAnotherServer()` method is called), the following operations are performed:

- For the current form (where the said button is pressed)`BeforeReopeningFromAnotherServer` event handler is called. The foregoing event handler accepts all parameters used to open the current form. A list of parameters can be modified by the developer. If when you leave an event, `Denial` parameter is set to `True` -, the form will not be reopened.
- Another (offline) form is created. Moreover, all server events for a form so created are triggered.
- For the said form (which has been so created) `OnReopenFromOtherServer` event handler is called. The event handler accepts a form being reopened and filling in parameters. If `StandardProcessing` parameter of the event handler is set to `True` -, the platform exports data from the previous form to the form so created based on `FillingInParameters` value. If `StandardProcessing` is set to `False`, it means that the developer has to export data from the previous form on their own. To generate a

list of copied attributes, use `FillingInParametersOnFormReopening` object. To fill in data in a newly created form based on that exported from the previous one, - use `FillInOnReopeningFromAnotherServer()` method.

- `OnOpening` event handler is called in a newly created form and all other events as per form opening protocol.
- The state of the previous form depends on its opening mode and location:
  - An independent form - will be closed, as soon as a newly created form is opened. Moreover, `OnClosing` event handler is called, while `BeforeClosing` handler remains uncalled. If `ClosingNotificationDetails` property is set for the previous form, its value will be exported to a newly created form. Notification is called, as soon as a newly created form is closed.
  - Locking form - will be closed, as soon as a newly created form is closed. Moreover, `OnClosing` event handler is called, while `BeforeClosing` handler remains uncalled.
  - If a form is placed on the home page -, it remains there. A newly created form will be opened in an independent mode.

Before opening a form, it is necessary to remember the following features:

- The form owner and form opening parameters remain unchanged.
- Attribute values of the following kinds are not exported:
  - Chart,
  - Spreadsheet document,
  - Planner.
- Data of the following kinds is exported, only if it is loaded in RAM in full:
  - Tabular section
  - Value list,
  - Value table,
  - Value tree,

29.6.5. Data exchange in operation

An offline mobile client needs to support continuous data exchange between the application main and standalone parts. Synchronization is required to make system operable:

- Data from the main infobase has to be exported to the standalone infobase. The said data needs to be exchanged to the extent sufficient to fill in standalone configuration object data.
- Data exported from the standalone infobase to the main infobase: This is data saved in the infobase when a mobile application operates in an offline mode.

So far as synchronization is concerned, it is recommended to consider both parts (standalone and main infobases) as independent nodes of data exchange mechanism supported by 1C:Enterprise. As such, requirements applicable to data exchange are based on a well-known and ready infrastructure:

- To register changes, an exchange plan is used.
- To export data from the standalone to the main part of a single infobase, standard message infrastructure or tools are used to access a list of changes duly registered.

Make sure that the said exchange plan includes objects made part of the standalone configuration. Per each infobase supported by a standalone mobile application, make sure that the main infobase has a proper exchange plan component.

Therefore, the general procedure is as follows:

- The first startup of an offline mobile client is always made subject to availability of the main server connection.
- As such, an offline mobile client accepts from the main server a standalone configuration and initializes its local infobase.
- As part of the initial startup, an offline mobile client calls the main server and executes the following operations:
  - It creats its own exchange node in the main infobase.
  - Moreover, it creats an exchange node in a standalone infobase which relates to the main infobase.
  - Further, initial population of a standalone infobase is ensured using data exported from the main infobase.
  - It regularly synchronizes data of the standalone and main infobases. The said synchronization is made when needed and to the extent practicable. "When needed" means that data is exchanged in a way ensuring that data is updated with the volume of data so transferred remaining unchanged per each synchronization portion in view of the communication channel used for that purpose. "To the extent practicable" means physical availability of a communication channel between a mobile device and main infobase.

So far as data exchange is concerned, the following distinctive features need to be noted:

1. When initial population of the standalone infobase is made, there is no need to use `CreateInitialImage()` method of exchange plan manager. On the main infobase side, generate a list of objects required to be made part of a proper exchange plan node (in fact, - offline mobile application instance) and send data back to the mobile application. The standalone configuration code saves this data in the infobase (if necessary, with a system operation protocol used to exchange data).
2. For data exchange purposes, the following can be used:
   1. Standard data exchange mechanism (using data exchange messages based on XML-files).
   2. A data exchange option when an offline mobile application initiates receipt of data from the main infobase and writes it in the standalone infobase without using a message file mechanism.

   A file-based mechanism is much more easier to implement. However, certain overhead costs have to be borne: generation and transfer of large message files between a mobile device and personal computer. Large volumes of data so transferred and restrictions typical of mobile device storage etc- all these problems have to be addresses by the configuration developer.

   A direct data transfer option has its own distinctive features: you need to provide for availability of a uninterrupted communication for entire synchronization duration. The said process can be time-consuming etc. Therefore, in each scenario developers have to select the method to run data synchronization between the standalone and main infobase.
3. The other important thing is that for synchronization purposes standard XDTO serialization mechanism is used. Moreover, if in the standalone configuration it is defined for a specific object that a set of attributes in the standalone configuration is different from that available in the main infobase, it means that you have to implement serialization/deserialization mechanism in full on your own. Standard serialization mechanisms supported by 1C:Enterprise are thus inoperable.

29.6.6. Standalone / main server switchover

When you use an offline mobile client, standalone and main servers can be operated simultaneously. An application is able to define whether the main server is available or not and call the main server from the code executed on the standalone server side. However, it cannot call a standalone server from the code executed on the main server side.

To determine availability of the main server, `MainServerAvailable()` global context method is used.
If a standalone server is used, you can call a method of the main server which is accessible in the main server common server module. For that purpose, use `MainServer` global context method. As such, it looks as follows: `MainServer.CommonModuleName.MethodName()`. If the main server is inaccessible, - an exception is returned.
If an offline mobile application client needs to call a server, the said call is made as per standalone configuration priorities. However, the system enables you to explicitly specify the server to be used (select `SetUsedServer()`). This method operates in the following way:

- If a server to be used is explicitly specified when a method is called, - the said server is used. For this purpose, "explicit" means `UsedServer` system enumeration values.
- If `Undefined` is returned when a method is called, - a server call mode is set which is defined before the recent used server is specified.
- When you exit a method where used server is changed, - further a server which is used before you enter the said method is set.

Consider an example:

```
SetUsedServer(UsedServer.Main); // string 1
SetUsedServer(UsedServer.Standalone); // string 2
SetUsedServer(Undefined); // string 3
```

After string 1 the main server is selected. After string 2 the standalone server is selected.
After string 2 the server used **before** the recently defined server is selected. Therefore, the main server will be used again, as the second-to-last server (string 1) is the main one.

29.6.7. Using an offline mobile application

29.6.7.1. Mobile device configuration update

An offline mobile client cannot be initially started in an offline mode. As such, you need to have an interrupted communication channel to connect to the main infobase, since upon initial startup a standalone configuration is loaded.
Further, upon each startup of a client application and establishment/reestablishment of connection between a mobile application and main server,

standalone configuration is checked for update availability. One of the following scenarios is as well feasible:

1. A need to update is identified when a client application is started before its actual use:
   - Standalone configuration update file is by default downloaded and installed on a device.
   - A standalone infobase configuration is updated using a configuration so downloaded.
2. Before connection with the main server is established, client operates offline and is able to connect to the main server:
   - A user is prompted to confirm that a configuration needs to be updated.
   - If a user gives their consent, a standalone configuration update file is downloaded to a mobile device. Client application is reloaded, following which a standalone infobase is updated using a configuration that has been downloaded.
   - If no consent is given by a user, client application continues to operate with the previous standalone configuration version, so that no connection to the main server can be established (actually, -it operates offline).
3. If a need to update configuration arises upon further connection to the main server (without operation in the offline mode), the current session is aborted and further the aforementioned scenario applies.

29.6.7.2. External references

External references - are references to objects not included in a standalone configuration, while they are used in standalone objects. As external references, those to configuration objects can be used, including objects which are unavailable on a mobile device. A mobile device fails to support any operation with standalone references: attribute value or object receipt etc. In fact, external references are intended to display real data in respective form fields. As a result, a user needs to take no care of data loss.

If data which is an external reference is displayed in the form field:

- If no connection with the main server is established:
  - This field becomes inaccessible.
  - A reference representation is cached, as soon as connection is reestablished.
  - If no external reference representation is saved on a mobile device, a message referring to unavailability of data is deemed its representation.
- If connection with the main server is established:

- A reference is available for further use.
- As such, the main server is called.

29.6.7.3. Command interface

If a command is not defined in a standalone configuration, it will be unavailable in an offline mode. If a command is defined in a standalone configuration, its availability in an offline mode will be defined by Behavior, if the main server is inaccessible command property. If the said property is set to Auto, it is interpreted as Do not modify behavior. If property editing is unavailable for a command, Do not modify behavior is used.

29.6.7.4. Users

No user operations are supported by an offline mobile client. Meanwhile, when an application is started, username and password are requested. Authentication is performed by the main server. If upon further startup of the application a user is specified which is different from that defined upon initial log in (when infobase is created), an error will be returned. Therefore, if a mobile device is used to operate a single main infobase by two or more users, -each of them needs to have a separate infobase (application) on a mobile device.

If a user with their details used to log in the main infobase changes their password, a standalone infobase is notified thereof, only when authentication is successfully completed on the main server.

29.6.7.5. History

History is stored both on the main and standalone infobase server. If a mobile device is granted access to the main server, -history is saved in both infobases. If a mobile device is not granted access to the main server, - history is saved in the standalone infobase only. As soon as access is granted to the main server, history is combined and saved in both infobases.

29.6.7.6. Favorites

Favorites are stored both on the main and standalone infobase server. If a mobile device is granted access to the main server, -favorites are saved in both infobases. If a mobile device is not granted access to the main server, - favorites are saved in the standalone infobase only. As soon as access is granted to the main server, favorites are combined and saved in both infobases.

29.7. Mobile device interaction during development

29.7.1. General information

Placing a mobile application on a mobile device is possible in several ways:

- Through the App Store (https://itunes.apple.com/ru/genre/ios), Google Play (https://play.google.com/) or Microsoft Store (https://www.microsoftstore.com). This method is used to distribute public versions of mobile applications.
  A mobile application prepared for publication is approved and published according to the rules of the respective application store. Details of this process in this book are not considered.
- Using the mobile version of "1C:Enterprise" for the developer. This method is used only for development of the mobile applications. Further, this method will be considered.

In order to start using the mobile version of "1C:Enterprise", you need to perform some actions:

- a mobile version of "1C:Enterprise" for the developer should be installed on the mobile device;
- mobile application published on a web-server;
- an infobase (application) has been created on the mobile device based on the published mobile application.

After performing these operations, we obtain a ready infrastructure for developing and executing a mobile application on a mobile device. At the same time, the system can be configured in such a way that the configuration update will cause the mobile version of "1C:Enterprise" for the developer to automatically update the configuration on the mobile device.
Further, each action will be discussed in more detail.

29.7.2. Installation of developer tools

29.7.2.1. General information

Depending on the configuration being developed, the developer may need different developer tools:

- Mobile developer platform in the case of developing a mobile platform application.

- Mobile developer client in the case of developing an application that operates in a mobile client.

The actual algorithms for the formation of developer tools are the same for both the mobile platform and the mobile client. Differences are present only in the file names that should be used in one way or another. So, if you need to get the prjios.zip file from the distribution package of the 1C:Enterprise mobile version (mobile.zip file) to prepare the mobile developer platform for OS iOS, then in the case of the mobile client, use prjios_client.zip file.

The table shows the corresponding file names for the mobile platform and the mobile client.

Mobile operating system **Client application kind**	**File name**
Android	
Mobile platform	1cem-arm.apk 1cem-arm64.apk 1cem-x86.apk 1cem-x86_64.apk
Mobile client	1cem-client-arm.apk 1cem-client-arm64.apk 1cem-client-x86.apk 1cem-client-x86_64.apk
Offline mobile client	1cem-standalone-arm.apk 1cem-standalone-arm64.apk 1cem-standalone-x86.apk 1cem-standalone-x86_64.apk
iOS	
Mobile platform	prjios.zip prjios_sim.zip
Mobile client	prjios_client.zip prjios_client_sim.zip
Offline mobile client	prjios_standalone.zip prjios_standalone_sim.zip
Windows	
Mobile platform	1cem-arm.appx 1cem-x64.appx 1cem-x86.appx
Mobile client	1cem-client-arm.appx 1cem-client-x64.appx

Mobile operating system Client application kind	File name
	1cem-client-x86.appx
Offline mobile client	1cem-standalone-arm.appx 1cem-standalone-x64.appx 1cem-standalone-x86.appx

In this table:

- suffix -arm, -arm64, -x86, -x64, and -x86_64 means the architecture of the processor used;
- -sim suffix means a version intended to be used in the device emulator (applicable to development for iOS OS only).

29.7.2.2. For OS iOS

To develop a mobile application for OS iOS, in is necessary to fulfill the following requirements:

- Managing a mobile device is only possible from an Apple computer (hereinafter referred to as - Mac) with installed OS X 10.8 and earlier.
- Mac computer should have one free USB port used to communicate with the mobile device.
- On this computer, it is necessary to install Xcode development tool of version 4.5 and higher (https://developer.apple.com/xcode/).
- It is necessary to register in the iOS Developer Program (https://developer.apple.com/programs/ios/).
- To use, you need a mobile device on iOS that meets the system requirements.

To install the mobile version of "1C:Enterprise" for the developer on a mobile device running iOS, perform the following steps:

- Attach mobile device to Mac computer;
- Unpack the file with necessary project of "1C:Enterprise" mobile version on Mac computer;
- Open the mobile version of 1C:Enterprise for the developer in the Xcode system (double click on file 1cem.xcodeproj or select file 1cem.xcodeproj using command File - Open in the Xcode system);
- Open tool Organizer using menu command Xcode Window - Organizer;
- In the opened window, in the left panel, select attached phone (marked with a green marker);
- In the bottom panel select the command Add to Portal;
- The Organizer tool will request the necessary phone certificates and provisioning profile from the Apple website and install them on the

phone automatically. You will need to enter the developer password received during registration in the iOS Developer Program.

- Using the Xcode menu command Product - Edit Scheme... it is necessary to open the settings dialog box of the current startup scheme. In the opened dialog box:
  - In the Scheme property, select 1cem value;
  - In the Destination property, select attached phone;
  - In the Build configuration property, select Release value;
  - Click OK.
- Start a project using Xcode command menu Product - Run.
- The mobile version of "1C:Enterprise" will be passed to the mobile device, started and its icon will appear in the list of installed applications of the mobile device.
- The mobile device can be disconnected from the Mac computer and then start the mobile version of "1C:Enterprise" by clicking on the "1C:Enterprise" icon in the list of mobile device applications. If at the moment of shutdown a mobile version of "1C:Enterprise" was started for a developer running on Xcode, it will be stopped.

29.7.2.3. For OS Android

To develop a mobile application for Android OS, fulfill the following requirements:

- Install on computer Android Studio (http://developer.android.com/studio/index.html). For developer platform functioning, it is necessary that Android SDK Platform-tools of version not earlier than 14 was installed on the computer.
- Check that the necessary components are installed, it can be done using Android Studio. To do this, in the start window, select Configure - SDK Manager. For convenience, it is possible to enable the check box Show Package Details in the opened window.
- On the Platforms SDK tab, you can check for the availability of the required Android SDK versions.
- Then on the SDK Tools tab, specify Android SDK Build-Tools, Android SDK Platform-Tools and Android SDK Tools. It is recommended to install all the Tools and SDK itself consistently. In this case, all versions will be compatible with each other.

- After the settings have been made, clicking the OK or Apply button will start the installation process of the necessary components. Before starting the installation process, Android Studio will ask to confirm the installation of the selected components (indicating the entire list).
- The computer must have one free USB port used to communicate with the mobile device.
- To use, you need a mobile device on Android that meets the system requirements.

To install the mobile version of "1C:Enterprise" for a developer on a mobile device running Android, perform the following actions:

- In the settings of the mobile device, select the following check boxes:
  - Unknown sources;
  - USB debugging;
- Download from the manufacturer's website and install a mobile device USB driver on your computer;
- Attach mobile device to computer;
- Using the OS Windows command line interpreter, execute the command:

Example for ARM platform:

```
%ANDROID_SDK%\platform tools\adb.exe install  r "<Platform
directory>\1cem arm.apk"
```

Example for x86 platform:

```
%ANDROID_SDK%\platform tools\adb.exe install  r "<Platform
directory>\1cem x86.apk"
```

Where, Platform directory - the directory in which the corresponding file with the mobile version of 1C:Enterprise is located. If the mobile version of 1C:Enterprise is already installed on the mobile device, this command will reinstall it.

- The mobile version of 1C:Enterprise will be passed to the mobile device and its icon will appear in the list of installed applications.
- The mobile device can be disconnected from the computer and then the system can be started by clicking on the "1C:Enterprise" icon « in the list of mobile device applications.

To install the mobile version of 1C:Enterprise for developer, it is possible to use the configurator command:

- For mobile platform: Main menu - Configuration - Mobile application - Using Android debug bridge - Install mobile platform.
- For mobile client: Main Menu - Configuration - Mobile client - Using Android debug bridge - Install mobile client.

In this case, you will need to perform additional setting of configurator. Installation of the mobile platform can be done either on a physical mobile device attached to computer or on device emulator created using the Android Virtual Device Manager. Virtual Device Manager can be started from Android Studio: Main menu - Tools - Android - AVD Manager.

29.7.2.4. On Windows

To develop a mobile application for Windows, fulfill the following requirements:

- Operation with mobile device is possible only from computer with the installed operating system Windows 8.1 and higher.
- You should have Microsoft account (https://login.live.com/).
- For operation, you need a device on Windows (tablet or computer with a touch screen) that meets the system requirements.

To install the mobile version of 1C:Enterprise for developer on device running Windows, perform the following steps:

- Copy the delivery file of the mobile version of 1C:Enterprise to the device (mobile.zip).
- Extract certificate file from it - 1cem.cer.
- Install the certificate in the Trusted people store on the local computer.
- Extract a file with necessary distribution package of desired architecture from the delivery file of the mobile version of 1C:Enterprise.
- Start PowerShell as administrator (system administrator rights are not enough);
- Execute command:

```
Show WindowsDeveloperLicenseRegistration
```

  In the opened window, specify the Microsoft account information;
- Execute command:

```
Add AppxPackage <Directory>\<file name>.appx
```

  Further:
  - <Directory> - the directory in which the mobile version of 1C:Enterprise is placed for the developer;
  - <file name> - name of the platform version to be installed.
- In the future, you can start the system by clicking on the 1C:Enterprise icon in the list of applications.

29.7.3. Publish a mobile application on a web-server

29.7.3.1. General information

IMPORTANT! When managing a mobile client, the actions listed in this section are not required.

In order for the mobile application to be available to the mobile platform, it is necessary to publish the mobile application on the web-server. In fact, the publication is divided into two parts:

- Preparation of infrastructure for operation and initial unloading of the mobile application configuration - is performed once.
- Updating the configuration of the mobile application in the prepared infrastructure.

Command Configuration - Mobile application - Publish... is intended for structure preparation. In the opened dialog box, select the check box Create a virtual directory on the web server. If the check box is not selected, the Name and Directory attributes will be unavailable, and pressing the Publish button will not create a virtual directory on the web-server.

IMPORTANT! Administrative rights are required to complete the publication.

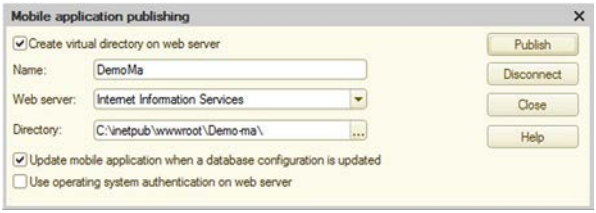


Fig. 299. Mobile application publishing

Then specify the properties of the virtual directory of the web-server:

- Name - specifies the name of the virtual directory. This name will participate in the URL configuration of the mobile application to indicate on the mobile device when creating the infobase.
- Web-server - determines which web-server is being used. If you are using the Microsoft Internet Information Services web-server, then it

becomes possible to select the Use operating system authentication on the web-server check box.

- Directory - specifies the physical directory on the hard drive in which the mobile application configuration file will be located as an xml file and where the virtual directory of the web-server will be displayed.

Check box Update mobile application when updating database configuration controls the capability to automatically update the mobile application configuration when updating the database configuration. If the check box is selected - the update will be performed.

You can also force a configuration update operation by using the command Configuration - Mobile applications - Update published application. If at the time of the call of this command the publication of the mobile application was not executed -, the publication dialog box described above will be opened.

IMPORTANT! The web server publishes the database configuration, not the editable configuration.

To perform the actual publication, click the Publish button. The following actions are performed at this:

- If the Create a virtual directory on a web-server check box is selected, a virtual directory is created on the web-server;
- A directory is created on the hard drive to which the virtual directory is displayed;
- It is proposed to export the configuration of the mobile application to the directory (specified in the attribute Catalog of the publishing dialog box);
- Checks if the database configuration of the configuration being edited matches or not;
- If the configuration of the infobase does not match the edited configuration, - it is proposed to update the configuration of the database. However, this operation can not be performed if it is required to publish the database configuration;
- The mobile application is being checked;
- If there are no errors, then the configuration of the infobase is downloaded to a file, otherwise, no download is performed.

The file with the mobile application configuration, unloaded using the Publish... command, has the fixed name 1cema.xml. Search (and update) of this particular file is performed when Update published application command is executed. When creating a virtual directory, the configuration file (1cema.xml) is set as its (directory) default page. This allows to specify the

URL in abbreviated form in the dialog box for creating an infobase on a mobile device.

The Disable button of the dialog box of the mobile application publishing cancels the publication: deletes the virtual directory on the web-server and the physical directory.

29.7.3.2. Publication details

A publication directory has the following structure:

- 1cema.xml - configuration description; specifies name, languages and required permissions.
- 1Cv8.1CM - file intended to accelerate infobase creation process. It contains configuration data.
- If a configuration has add-ins, a publication directory will have the following directory structure (all directories and files are optional):
  - Android
    - <Add-InName>.xml
    - ARM
      - <Add-InName>.so
      - <Add-InName>.apk
    - ARM64
      - <Add-InName>.so
      - <Add-InName>.apk
    - i386
      - <Add-InName>.so
      - <Add-InName>.apk
    - x86_64
      - <Add-InName>.so
      - <Add-InName>.apk
  - iOS
    - <Add-InName>.xml
    - Universal
      - <Add-InName>.a
      - <Add-InName>.dylib
  - WindowsRuntime
    - <Add-InName>.xml
    - ARM
      - <Add-InName>.dll
    - i386
      - <Add-InName>.dll
    - x86_64

    - <Add-InName>.dll

If a configuration has several add-ins, a publication directory (in any relevant directory structure) will have unique file sets with their number corresponding to that of add-ins available in a configuration.
All files created as a result of publication are deemed integral part of a published configuration. Deletion,relocation or renaming of files will result in abnormal performance of a published configuration.
To ensure normal operation of a published configuration, make sure that a web server has registered MIME kinds for the following extensions:

- Performed by Designer upon publication:
    - .xml - application/xml.
    - .1CM - application/octet-stream.
- Manual registration is required, if add-ins have to be debugged:
    - .so - application/octet-stream.
    - .apk - application/octet-stream.

29.7.4. Start and restart the application on the mobile platform

29.7.4.1. General information

Start of the mobile application is carried out directly on the mobile device or from the configurator. Mobile version of 1C:Enterprise for developer should be installed on the device.
Start from configurator is possible when two conditions are simultaneously met: development is performed using the Android OS and the corresponding configuration of the configurator is performed.

29.7.4.2. Restart mobile application published on web-server

If the mobile application is started using the mobile developer platform and the application has the Restart from configurator check box, then, there is an option to restart the mobile application from the configurator. This can be done using the main menu command Debugging - Start of debugging - Mobile application: start debugging or Debugging - Start of debugging - Mobile client: start debugging.
If the configuration has been modified (changes have been made), the Designer asks the following question: The configuration you are editing is

different from the database configuration. Update database configuration? Select Yes to save your changes.
If the No button is selected, the mobile application will be restarted without updating the configuration.
If when publishing the mobile application, the Update mobile application when the database configuration is updated check box was selected, then automatically updating of the publishing of the mobile application configuration will be performed.
Add-ins used in the configuration will also be placed in the directory where the virtual directory of the web-server is displayed. Only those add-ins files that are intended for use on Android and iOS OS (if they exist in the archive components with an add-in) will be copied.
The mobile application performs an update check on opening if the Restart from the configurator check box is selected in the properties of the mobile application on the mobile device. If this check box is cleared, the opening does not check for updates. However, this check can be done manually.

29.7.4.3. Starting and restarting a mobile application when using the Android debug bridge

You can start and restart the mobile application using the configurator. This can be done using the main menu command Debugging - Start of debugging - Mobile application: start debugging or Debugging - Start of debugging - Mobile client: start debugging. In order to use the Android Debug Bridge, the configurator should be properly configured. In addition, the configuration should be published in the file system (publishing on the web-server in this case is not mandatory).
If the configuration has been modified (changes have been made), the Designer asks the following question: The configuration you are editing is different from the database configuration. Update database configuration? Select Yes to save your changes.
If the No button is selected, the mobile application will be restarted without updating the configuration.
If when publishing the mobile application, the Update mobile application when the database configuration is updated check box was selected, then automatically updating of the publishing of the mobile application configuration will be performed.
When using Android Debug Bridge, the add-ins used in the configuration will also be copied to the mobile device being used. Only those files of the add-in

that are intended for use on the Android OS (if they exist in the archive components with add-in) will be copied.
After restarting the mobile developer platform, a new configuration will be detected and you will be prompted to perform the update.

29.7.5. Debugging an Application

Debugging an application on a mobile device is only possible using the HTTP debugging protocol. Mobile application running under Android OS can be started in debug mode directly from the configurator using Android Debug Bridge. Mobile application running on iOS or Windows cannot be started in debug mode directly from the configurator.
To debug the application, two parameters are used in the properties of the infobase of the mobile application:

- Debugging allowed - indicates whether debugging is allowed in this mobile application or not.
- Debugging address - indicates location of the debug server if debugging is enabled.
- The above parameters are ignored if:
  - The infobase is started directly from the configurator - in this case, the address of the configurator performing the start of the mobile application is the address of the debug server; it is applicable only for operation via the Android Debug Bridge;
  - For configuration, the Restart from configurator check box is selected- in this case, the debug server address is acquired from the publication of the mobile application.

29.7.6. Mobile application management for developer

29.7.6.1. Creating an application

In order to create an application on a mobile developer platform, you should:

- Start 1C:Enterprise on a mobile device;
- Select the Add Application command;
- In the opened window, enter the URL of the web-server on which the mobile application is published in the Address field;

- If necessary, specify the username and password for authorization on the web-server on which the mobile application is published and click the Download button;
- Then specify the name of the application and click Finish;
- After the window closes, the application will be created.

The list of applications will be skipped, and the application will be immediately started if it is the only one in the list. In this case, to jump to the list of applications, select the List of applications command in the main menu of the application.

29.7.6.2. Changing application properties

Application properties are changed in a special window. Depending on the operating system, access to this command is different:

- For OS iOS: on the right side of the row with the name of the application you need, click on the picture with a larger symbol.
- For OS Android: perform a long press (or touch on the right side of the screen) on the desired application. In the opened context menu, select the Change command.
- On Windows OS: perform a long press (or touch on the right side of the screen) on the desired application. In the opened context menu, select the Change command.

In the opened window, you can change the properties of the application that were specified when it was created: application name, configuration location URL, username and password for accessing the web-server where the mobile application configuration is located.

Also in this window you can perform the following actions:

- Start mobile application. To do this, click the Open button.
- Check for the availability of a new version of the mobile application configuration on the web-server. To do this, click the Check for updates button. If there is a new version of the configuration, you will be prompted to update the mobile application.
- Delete the application using the Delete button.

IMPORTANT! After the application is removed, the infobase cannot be recovered.

- Configure the ability to automatically update and restart the mobile application in case a new version of the configuration is found on the web-server. To do this, change the state of the check box Restart from configurator.

- Configure debugging capability (Debugging is enabled check box) and specify the debugger address (in the Debugger Address field).

The list of applications will be skipped, and the application will be immediately started if it is the only one in the list. In this case, to jump to the list of applications, select the List of applications command in the main menu of the application.

29.7.6.3. Uninstalling the application

To delete an application, select the Delete command and confirm your action. Depending on the operating system, access to this command is different:

- For OS iOS: click the Change button, then click the picture on the left side of the row with the name of the application to be deleted. After that, on the right side of the same line, click Delete.
- For OS Android: perform a long press (or touch on the right side of the screen) on the desired application. In the opened context menu, select the Delete command.
- On Windows OS: perform a long press (or touch on the right side of the screen) on the desired application. In the opened context menu, select the Delete command.

The list of applications will be skipped, and the application will be immediately started if it is the only one in the list. In this case, to jump to the list of applications, select the List of applications command in the main menu of the application.

IMPORTANT! After the application is removed, the infobase cannot be recovered.

29.7.7. Build a mobile application before publishing to the application store

In order to publish a mobile application in the application store, you need to build it using the mobile application builder. Mobile version of 1C:Enterprise system for developer cannot be published in the application store.

A manual description of the process of publishing a mobile application should be received in the developer documentation of the corresponding application store.

29.8. Mobile Application Builder

29.8.1. General information

The mobile application builder is designed to store and compile various mobile application artifacts into a single file, suitable for publication in the application store or for direct use on the device. The components of these artifacts include:

- Configuration of application.
- Graphic information.
- Multimedia information.
- Add-ins.
- Actually the mobile version of 1C:Enterprise.

The builder allows to build both mobile applications and mobile client applications. In both cases, a single assembled mobile application allows to use multiple configurations.

The mobile application builder functions only under Windows. Function under control of other operating systems is not supported.

Assembly is supported for all platforms on which the mobile version of the 1C:Enterprise system is running, while it is possible to disable the assembly for operating systems that are not required in each specific case. However, within one operating system, the variability of the assembly is not supported. This means that for Android and Windows OS, all architectures are always assembled, without capability to manage it.

The result of the assembly is:

- OS iOS: project file for self-assembly in Xcode, an executable file (.ipa-file) and a mobile application for operation in the iOS emulator. Capability to build .ipa-file is governed by the setting.
- OS Android: executable files (.apk files) for ARM, ARM64, x86 and x86-64 architectures.
- Windows OS: executable files (.appx files) for x86, x86-64 and ARM architectures.

29.8.2. Setting the builder

29.8.2.1. General information

Setting up the builder consists of several stages, the most important of which are the configuration of the parameters of the assembly infrastructure and the parameters of the supplier of applications. If all the setting steps are not completed, - the mobile application will not be assembled.

General scheme for setting a builder is as follows:

1. Setting of the builder parameters are performed using the setting dialog box.
2. Setting of the parameters of the mobile applications provider is performed.
3. One or more versions of the mobile platform are downloaded.
4. Files with provisioning profiles are loaded if you plan to use push-notifications for OS iOS.
5. One or more options of different audio information for mobile applications is loaded.
6. One or more options of various graphic information for mobile applications are loaded.
7. Mobile configurations are loaded from which mobile applications will be built.
8. Necessary entries for mobile applications are created.

After completing all the steps, you can assemble the mobile applications and publish them in the application stores.

29.8.2.2. Infrastructure setting

29.8.2.2.1. General information

The first step is to generate the necessary environment on the computer on which you will build applications for mobile devices. In the future, this computer will be called the assembly server.

The environment is generated depending on which operating systems will be built for:

- General requirements:
  - JDK 1.8 or later.
- OS iOS (if necessary to build .ipa-files):
  - Utilities pscp and plink from PuTTY package.
  - Mac computer with Xcode version 8.0 and higher installed.
- OS Android:
  - Relevant version of Android Studio with SDK installed.
  - The use of SDK versions and necessary components (Tools, Build tools) that have not reached the release stage (beta, release candidate, etc.) is not supported.
- Windows OS:
  - Windows SDK.
  - If it is necessary to build applications for operation on Windows OS, the assembly server must function under Windows 8.1 and later.

After installing all the necessary components, you need to configure the parameters of the builder using the dialog box Service - Builder parameter settings.
The Working directory and builder cache parameter specifies the path to the directory that is used for the actual assembly and storage of caches of the components used. A directory can occupy a substantial amount of hard drive space - of several gigabytes. Clearing this directory (between build processes) does not affect the build result. The directory path should contain **only Latin characters**. The user, on behalf of which the 1C:Enterprise server is running, used for the build, should have full access to the working directory of the builder.

29.8.2.2.2. For OS Android

The required version of the Android SDK is determined by the version of the mobile platform that is used for assembly. The version used for assembly is determined by the value of the `Android:targetSdkVersion` attribute of the `uses-sdk` element of the AndroidManifest.xml file of the used mobile version of 1C:Enterprise. The builder allows to build mobile applications for Android using Gradle, while building using Apache Ant is also possible. Mobile version of 1C:Enterprise of version 8.3.10 and later can only be built using Gradle.
Remember the following when setting the paths to Android SDK:

- Android SDK parameters (version 25 and earlier), as well as the Apache ANT parameter (for SDK versions 25 and earlier), should be specified only if the mobile version of 1C:Enterprise is used for assembly, which can only be assembled using Apache ANT (version 8.3.9 and earlier). Information about this is available in the card of the downloaded mobile version of 1C:Enterprise in the Assembling for Android is implemented using field. There is also an indication of the minimum version of the Android SDK, which is required when building an application that operates with specified mobile version of 1C:Enterprise. If the mobile version of 1C:Enterprise requires for its building:
  - Gradle - a mobile application using this version cannot be built using the Android SDK version 25 or earlier.
  - Apache Ant - a mobile application using this version cannot be built using the Android SDK version 26 or later.
- For a generic build (using both Gradle and Apache Ant), customize as follows:

- Android SDK (version 26 and later) parameter should point to the directory in which the Android SDK is installed from the components of Android Studio.
- Android SDK (version 25 and earlier) parameter should point to the directory where the “old” option of Android SDK is installed. Apache ANT (for SDK version 25 and earlier)parameter should be specified only if the Android SDK (version 25 and earlier) parameter is specified.
- It should be remembered that the "old" option of the Android SDK Tools (running using Apache Ant) can only be received using a direct link (http://dl-ssl.ggle.cm/Android/repsitry/tls_r25.2.5-Windows.zip).

29.8.2.2.3. For OS iOS

If you want to build an executable file to run under OS iOS, you should select the Build application package (.ipa) on Apple computer check box and set access parameters to this computer. Correctness of specifying parameters can be checked using the Check connection hyperlink. If all settings are correct, then after clicking on the hyperlink, a window will open in which the version of the operating system on the Apple computer is indicated. If the Build application package (.ipa) on Apple computer check box is cleared, the build result for OS iOS will be a project archive for self-building the application using Xcode and a mobile application for operation in the emulator. In order for the mobile application builder to perform any actions on the side of the Apple computer, this computer must be configured. To do this, follow these steps:

1. Install all available updates of OS X.
2. Install Xcode program. Get the package here: https://developer.apple.com/xcode/.
3. Download and install in a keychain System certificate Apple Worldwide Developer Relations (http://developer.apple.com/certificationauthority/AppleWWDRCA.cer).
4. Using the Apple developer portal tools, create and install a distribution certificate for the Apple AppStore in the keychain System or for corporate distribution:
   - This will require the creation of a private key by means of a keychain.
   - It should be noted that all private keys and certificates must be located in the System chain.

- After creating a private key, on the Access tab of the key properties dialog box, set Allow all programs to access this object.

5. Using the Accounts tab of Xcode settings, set up a provisioning profile corresponding to the distribution certificate.
6. In the Shared access settings, system settings of your Mac, you must enable the Remote login mode.
7. Here you should get the name of the computer, which is then specified in the settings of the mobile application builder.
8. You should also make sure that the IP port number 22 (SSH) of the Apple computer is available for the computer on which the server part of the mobile application builder is running.

29.8.2.2.4. On Windows

The mobile application builder allows building the mobile applications operating under the management of both Windows 8.1 (including Windows Phone) and Windows 10. The supported OS is determined by the version of 1C:Enterprise platform used for the building:

- 8.3.13 and lower versions -are supported by Windows 8.1, Windows Phone 8.1, and Windows 10.
- 8.3.14 and earlier versions -are supported only by Windows 10.

So, it is necessary to download and install only Windows SDK which will be really used for the building of the mobile applications. Both parameters (Path to Windows 8.1 SDK and Path to Windows 10 SDK) must be specified only when it is expected to build the mobile applications that use the whole rage of 1C:Enterprise mobile versions. Otherwise, it is recommended to install only Windows 10 SDK (and to fill in only one parameter of the builder setting- Path to Windows 10 SDK).

29.8.2.2.5. Artifacts automatic clearing

Due to the fact that 1C:Enterprise mobile version files and collected mobile applications occupies a significant amount, the capability to automatically clear the infobase from outdated versions of both mobile version and assembled applications is implemented. For this, the builder docks the download date (for the mobile version of 1C:Enterprise) and the latest build (for the mobile application). Then, the deletion dates (separately for the mobile version of 1C:Enterprise files and build results) are calculated and all objects whose modification date is less than the resulting deletion date are deleted. Object storage time is configured separately for the 1C:Enterprise mobile version files and build results files. This setting is performed using the following parameters:

- Mobile file storage duration, days. The default value is - 180 days.
- Duration of storage of the built versions, days. The default value is - 90 days.

If any of the parameters describing the duration of storage is set to 0, then automatic deletion will not be used for the corresponding objects.
If the deleted object is required again in the builder, then:

- Files of the mobile version of 1C:Enterprise can be re-downloaded from the mobile.zip file of the corresponding version.
- Mobile application files will need to be rebuilt.

29.8.2.3. Provider parameters setting

The next step is customization of the parameters of the provider of the built mobile applications. To do this, open the dialog box Service - Provider parameters setting.
First, you should specify for which mobile operating systems, applications will be built in the builder. For this purpose, a group of check boxes Build a new applications of this provider. Selecting each check box makes a tab available with parameters for the corresponding operating system.

Parameters for Android OS

Specify ID prefix of the mobile application. In the future, the full ID of the mobile application will be generated from the prefix, which is specified in the parameters of the provider and the application ID itself.
To build a mobile application, create (or import) a developer key (Developer key group). With this key, the built application is signed.
If built mobile application will be published to the application store directly from the builder, using the commands of the Access to the developer console group, you should get the required access.

Parameters for OS iOS

Specify ID prefix of the mobile application. In the future, the full ID of the mobile application will be generated from the prefix, which is specified in the parameters of the provider and the application ID itself.
On the Certificates tab, specify the certificates that will be used to sign the built application (.ipa-file). Clicking the Add button downloads a list of certificates from an Apple computer. Because of this, if you are not set connection to Apple computer, you will not be able to add a certificate. Certificates are used only if building is done using Xcode version 8 and earlier.
The iTunes Connect Access tab describes usernames and passwords for accessing iTunes Connect. These settings are used when you need to download the built applications to the application store (AppStore).

On the Developer groups tab, you can specify identifiers of development groups corresponding to accounts on the portal https://developer.apple.com/. For each group of developers, there is a capability to specify the default distribution method for applications. The specified value will be used when setting the parameters of the built mobile applications for OS iOS.

Parameters for Windows OS

Specify ID prefix of the mobile application. In the future, the full ID of the mobile application will be generated from the prefix, which is specified in the parameters of the provider and the application ID itself. It should be remembered that the prefix is assigned by the Windows Store, and not entered manually. To get this prefix, click the Go to Windows Store link and get ID using the web-interface of the applications store.

The Provider ID in the Windows Store field is automatically filled when the developer certificate is installed. The Provider name field in the Windows Store can be received from the Package/Properties/Displayed Name of the publisher field. This field is available in the description of any application using the command Application management - Application identity.

A developer certificate is installed using the Install developer keys command. Before executing the command, you must enter a password to access the key in the Password for private developer key field. Two files should be selected and uploaded: with ".cer" extension (the actual certificate) and with ".pvk" extension (the private key for this certificate). The information on the developer key obtainment is available here: https://msdn.microsoft.com/en-us/library/windows/desktop/jj835832(v=vs.85).aspx (in Ebglish).

29.8.2.4. Components used

The building of the mobile application is performed on the server side. The following software should be installed on the computer (or computers) on which the server part of 1C:Enterprise of the mobile application builder is installed (as needed):

1. Android Studio package. The package can be downloaded from here: http://developer.android.com/studio/index.html.
2. Java SDK package. The package can be downloaded from here: http://www.oracle.com/technetwork/java/javase/downloads/index.html.
3. Apache Ant package. The package can be downloaded from here: http://ant.apache.org/bindownload.cgi.

4. PuTTY package (utilities pscp and plink are used). The package can be downloaded from here: http://www.chi-ark.greenend.org.uk/~sgtatham/putty/download.html.
5. Windows SDK package. The package can be downloaded from here: https://msdn.microsoft.com/en-us/windows/desk-top/bg162891.aspx.

29.8.3. Download the required data

29.8.3.1. General data

29.8.3.1.1. Mobile versions

In order to build any mobile application, you must download the required mobile version of 1C:Enterprise to the builder. Mobile version is distributed in mobile_A_B_C_D.zip files. Where A_B_C_D - is the full mobile version number. So, for version 8.3.12.64, the file name will look like this - mo-bile_8_3_12_64.zip. To download, you must use this file. The builder will download only those files that are required for building. The actual number of the downloaded version is determined by the version.txt file, which is located in the root of the archive.
Mobile application builder supports both full and partial download of the mo-bile version. The loading process determines the following information:

- Version of downloaded files.
- Technology of building for Android OS.
- The required version number of the Android SDK that will be used when building for the Android OS.
- All mobile version variants loaded (Android, iOS and Windows):
  - For building a mobile application.
  - For building a mobile client.
  - For building an offline mobile client.

Thus, already when downloading the mobile version, it becomes clear what and how can be assembled using the downloaded version.

In this example, version 8.3.12.64 is loaded, for Android build, using this version, Android SDK version 27 will be required and building will be per-formed using Gradle. Also, this mobile version can be used to build both mo-bile applications and mobile clients for all supported operating systems.

At any time, you can reload the mobile version with the command Download platform. When reloading, it is not checked that the same version is loaded as it was downloaded before. Therefore, such a download should be done carefully.
If the mobile version of the platform is outdated and deleted, then at the bottom of the form there will be an inscription Mobile platform files are deleted (outdated). In the form of the list there will also be a check box indicating this.
When a mobile version is loaded into Mobile Application Builder, it is recommended to follow the below instructions:

- Load exactly a file downloaded from 1C resources. No preprocessing of mobile_<version number>.zip is required. The builder will download only those files that are required for building.
- Whenever the third digit in the mobile version number is changed, load a file of a further mobile version, only if you have already updated the builder to the version available in the version distribution package. Otherwise, -builder's behavior will generally remain undefined. For instance, mobile version 8.3.13 is supplied with builder 2.0.9. Mobile version 8.3.14 is supplied with builder 2.0.10. Thus, mobile version 8.3.14 can be loaded, only as soon as you update Mobile Application Builder to version 2.0.10.

29.8.3.1.2. Provisioning profiles

Provisioning profiles -are the special files granting the mobile application the access to some iOS features. Downloaded provisioning profiles are used to build .ipa files. A downloaded provisioning profile can also be reloaded via the item form and the Load provisioning profile command.

29.8.3.1.3. Audio information

This directory stores audio information that will be used when building mobile applications. The developer can download multiple sets of audio information. The builder does not impose any restrictions on the combination of a set of audio information and various mobile applications. To download, you should use the Download audio information command.
The downloaded archive should contain the following directories: iOS, Android and Windows (character register is important). Names of files in these directories should consist only of the characters ".", "_", numbers and Latin letters in the lover register. For iOS, only files with WAV, CAF and AIFF extensions are supported, other files are skipped when building a mobile application.

Audio information can be re-downloaded at any time. If required, - there is a capability to unload the entire set back in the form of an archive. To do this, use the command More -Unload audio information.

29.8.3.1.4. Graphic information.

This directory stores graphical information that will be used when building mobile applications. The developer can download multiple sets of graphic information. The builder does not impose any restrictions on the combination of a set of graphic information and various mobile applications. To download graphic information, use the Upload graphic information command of the form.

When downloading graphic information, the system expects that the graphic information will be represented by a zip file with the following structure (the register of the file and directory name - is important!):

- For OS Android: Android directory. Files in this directory:
  - Icons of the mobile application with the appropriate permission:
    - icon-36x36.png;
    - icon-48x48.png;
    - icon-72x72.png;
    - icon-96x96.png;
    - icon-144x144.png.
  - Splashes of the mobile application of the appropriate resolution:
    - splash-320x480.png;
    - splash-480x854.png;
    - splash-640x960.png;
    - splash-768x1024.png;
    - splash-800x1280.png;
    - splash-854x480.png;
    - splash-1024x768.png;
    - splash-1242x2208.png;
    - splash-1280x800.png;
    - splash-1536x2048.png;
    - splash-2048x1536.png;
    - splash-2208x1242.png.
  - Icons for displaying push-messages of the appropriate resolution and size:
    - pushsmallicon-18x18.png;
    - pushsmallicon-24x24.png;
    - pushsmallicon-36x36.png;

    - pushsmallicon-48x48.png;
    - pushsmallicon-72x72.png;
    - pushlargeicon-36x36.png;
    - pushlargeicon-48x48.png;
    - pushlargeicon-72x72.png;
    - pushlargeicon-96x96.png;
    - pushlargeicon-144x144.png.
- For OS iOS. iOS directory. Files in this directory:
  - Icons of the mobile application with the appropriate permission:
    - icon-29x29.png;
    - icon-40x40.png;
    - icon-50x50.png;
    - icon-57x57.png;
    - icon-58x58.png;
    - icon-72x72.png;
    - icon-76x76.png;
    - icon-80x80.png;
    - icon-87x87.png;
    - icon-100x100.png;
    - icon-114x114.png;
    - icon-120x120.png;
    - icon-144x144.png;
    - icon-152x152.png;
    - icon-167x167.png;
    - icon-180x180.png;
    - icon-1024x1024.png.
  - Splashes of the mobile application of the appropriate resolution:
    - splash-320x480.png;
    - splash-640x1136.png;
    - splash-640x960.png;
    - splash-750x1334.png;
    - splash-768x1024.png;
    - splash-1024x768.png;
    - splash-1242x2208.png;
    - splash-1536x2048.png;
    - splash-2048x1536.png;
    - splash-2208x1242.png.
- For Windows Store OS. Windows Store directory. Files in this directory:

- Icons for a mobile application of the appropriate resolution for the tablet:
  - icon-50x50.png;
  - icon-150x150.png;
  - icon-30x30.png.
- Splashes of the mobile application of the appropriate resolution for the tablet:
  - splash-620x300.png;
  - splash-1208x800.png;
  - splash-2048x1536.png.

The archive may not contain all the pictures listed above. In this case, the standard icons and splashes from the mobile version of 1C:Enterprise will be used. When building a mobile application in the building log, it will be indicated which image is taken from the installed graphic information, and which remains standard. Renaming a picture from the format used for downloading to the mobile application builder to the format used for a specific mobile operating system is performed by the builder automatically. The building log displays information on the compliance of the image name from the archive of graphic information and the real name of the image.

To check that the loaded set of images contains all the necessary files, use the Check image components command. As a result of the check, a dialog box is displayed that contains a list of pictures that are not in the downloaded archive.

Graphic information can be reloaded at any time. If required, - there is a capability to unload the entire set back in the form of an archive. To do this, use the command More - Unload graphic information.

29.8.3.2. Configurations

The directory Mobile configuration is intended for storing configurations. Configurations should be stored exclusively in groups. A group of configurations is intended to store different versions of the **same** application. It is strongly not recommended to mix different applications in the same group.

Each group can store one kind of configuration:

- designed to build mobile applications;
- designed to build mobile client applications.

To determine the build method, use the Configuration type group field. After at least one configuration has been loaded into the group, it becomes impossible to change the type of configuration in the group.

Once a group is created, you can enter the group and upload the actual configuration. The builder controls that the correct configuration is loaded into the group:

- Configuration for building a mobile application: files with extension 1cema.zip and 1cema.xml.
- Configuration for building a mobile client application: files with extension 1cemca.xml and 1cemca.zip.

The following information is received from the loaded configuration:

- Configuration name (configuration property Name).
- Configuration view in all supported languages (configuration property Synonym).
- Configuration version (Configuration property Version).
- Version format unload.
- Primary configuration language.
- List of configuration permissions (configuration property Required permissions of the mobile application).
- Add-ins (if the configuration is loaded in a .zip file).

The configuration name and version define the name of the loaded configuration.

In the mobile application builder, you can change only the comment to the loaded configuration. The remaining parameters should be edited only in the configurator.

At any time, you can reload the configuration. In addition, you can upload the configuration to a file on the hard drive. To do this, use the command More - Unload configuration. This action has no sense because the unload results do not allow restoring the configuration for subsequent change.

Make sure that a configuration being loaded into the builder is retrieved from Designer first. Depending on the kind of application being used, there are certain upload commands and related formats:

- For mobile platform. Use Main Menu- Configuration- Mobile Application - Write in File. Format: zip-file 1cema.zip. Uploaded data consists of a configuration file and related information, including add-ins which are available in Add-In Designer templates.
- For mobile client. Use Main Menu- Configuration -Mobile Client - Write in File. Format: xml-file 1cemca.xml or zip-file 1cemca.zip. If a configuration has add-ins, - data is retrieved as a zip-file only. If a configuration has no add-ins, - data is retrieved as a xml-file only.

Mobile Application Builder can upload configurations of any format used as of the date when any relevant version of the builder is issued. Please note that data has to be loaded into the builder exactly in a format in which configuration is retrieved from Designer. No further actions with an upload file are

required. Whenever uploaded data is made simpler, a configuration can be successfully loaded into the builder, while no building is feasible.
It is recommended to update the builder first when you update the mobile version and further repeatedly upload a configuration using Designer of a new version, even if no data is changed in the said configuration. As such, maximum allowable compatibility of a configuration so uploaded with mobile 1C:Enterprise of a new version is ensured.

29.8.4. Setting of the application to be built

29.8.4.1. General information

The storage structure of mobile applications is organized similarly to the storage configurations. First, a group is created that describes the general parameters of the mobile application to be built, then, in this group, an element with unique settings is created for each version of the mobile application.
The parameters of the application to be built can be set both in the group of applications and directly in the item. The parameters "inheritance" scheme is used in the general case: practically all setting made in the group (with small differences described separately) are transferred to the application item. Each separate setting can be changed or the values "inherited" from the group can be saved in the application item. In the result, the developer can set all the main parameters of the application to be built once and subsequently only create a new copy of the application when the developer releases another version or several configurations.

29.8.4.2. Applications group

29.8.4.2.1. General information

The group of mobile applications describes the general parameters that will be used to build applications and the initial installation of the parameters of a particular application. Before creating an application, you should familiarize yourself with the 1C Company licensing policy, which describes the rules for licensing mobile applications. To familiarize with licensing policy, follow the link from the mobile application group (or from the Service menu -Licensing policy) and then select the corresponding check box.
First, the group determines which type of applications will be built in this group: mobile application or mobile client. It is impossible to build both mobile applications and mobile clients in one group. You can change the build-

ing type only until no specific application is created in the group. After creation, the Build field becomes unavailable for editing. If Build is set to For mobile client, you can select a mode for the builder to build an application: as a mobile client or offline mobile client. Any relevant mode is set in standalone mode field. Unlike Build field, you can any time specify whether an application needs to be built as an offline mobile application, or not (except applications already built). Therefore, per each configuration version you can build both mobile client and offline mobile client applications. For that purpose, you need to create two applications with the only difference being their standalone mode property value (and build number).
Secondly, it is necessary to determine for which mobile operating systems applications will be built by default. To do this, use the Build Mobile Application group on the General settings tab.
Then, the mobile version 1C:Enterprise will be indicated, which will be used to build group applications. If a specific version - is specified, it will always be used, until a new version will be chosen in the group. If the Mobile platform field is left empty, then the builder, when creating a new item in this group, will select the mobile version of 1C:Enterprise with the maximum number (from the list loaded into the builder). Thus, it becomes possible to automatically use the “latest” mobile version when building a new application.
Then you should specify the audio information and a set of graphic files that will be used by applications in this group. The values of the Graphic resources and Audio resources properties will be "inherited" when creating a new mobile application.

29.8.4.2.2. Mobile Application IDs

In the Mobile Application IDs group, you should specify the ID of the mobile application that will be built in this group. This is the second part of the ID (the first one is specified when setting the properties of the mobile application provider). After specifying the IDs, the builder will automatically generate complete application IDs for mobile operating systems marked for building in the Build mobile application group. It should be remembered that the ID of the mobile application for Windows should be received in the application store of this OS, and not to invent.
The full ID of the mobile application for each mobile OS is generated according to the following algorithm: Application ID Prefix field from the corresponding provider property, symbol "." and the Solution ID field (for the relevant OS). This field is mandatory. The solution ID may consist of Latin

characters, numbers and symbols "." and "_". No part of the solution ID (including the number following any "." character) can begin with a digit.
To check that there are no applications with the resulting IDs in the database of the builder, use the Check for IDs uniqueness command. This command does not perform global check.

29.8.4.2.3. Configurations used

The Configurations and version tab shows the application version in the A.B.C format (Application version field) and the build number (Building number). Both of these parameters, together, will provide the full version number of the application in the format A.B.C.D, where D - is a number of building. If one configuration is listed in the configuration list, then the version number can be received automatically from this configuration. If you use multiple configurations, the application version number is better to set yourself. Below the property Application Version, what will be used as the application version is specified.
The list below the version number indicates the configurations that will be used when building the mobile application. This table lists the **groups** from the mobile configurations directory. Specific items here are impossible to specify. The formation of a list of specific configurations that will be placed in the collected application will be performed when creating a specific application.
Depending on the OS for which the mobile application will be built, bookmarks will be available in the group, allowing you to specify parameters for each OS used:

- For OS Android: If the mobile application will operate with Google maps, then you need to receive the corresponding key using the personal account of the developer on Google website. To receive a key, you should use the data presented in the Parameter of receiving a key for operation with Google maps field. The received key must be entered in the Key for operation with Google maps field.
- For OS iOS. If the mobile application will be used on OS iOS devices and it is planned to build an application package (.ipa) for a mobile application, then, depending on the version of Xcode used, the following parameters should be set:
  - Xcode 8 and older: in the Provider group field, specify the group of developers with the required ID (the setting is made in the provider parameters). In the Distribution method field, you can change the distribution method for the current mobile application.

In Xcode, all necessary certificates must be configured and installed.

- Xcode 7 and earlier: In the Certificate for building field, specify the certificate that will be used to digitally sign the application package.

In fact, the contents of the Development team field tells the builder what version of Xcode will be used for building. You need to understand that if the Development team field is cleared, but the Apple computer is running Xcode version 8 and older, - the build will not be possible.

If the mobile application uses push-notifications, then in the Provisioning profile field specify the appropriate profile that is required for the Xcode system to build the application package.

In the ID in the AppStore field, enter the numeric ID corresponding to the built application in the application store of Apple.

- For Windows OS. You can specify the background color (in hexadecimal form) of the start screen.

29.8.4.2.4. Mobile client

If an application for a mobile client is being built, then the Mobile Client tab becomes available for setting. This tab is used to set the mobile client application.

On the List of applications tab, a list of information bases with which the created mobile application can operate is generated. To add an infobase, click the Add button, then in the Type field specify the Infobase, select the configuration and its representation in the mobile client list and indicate the URL where the published infobase is located with the specified configuration. You can specify the last action (field C.) this infobase will be immediately displayed in the list of applications of the mobile client or to display it you will need to manually indicate this already in the built mobile client. This may be required if the assembled mobile client application includes access to the infobase that is not needed by a very large number of users (relative to the total number of users of the application). In this case, users who need such an infobase will be able to add it to their list, while other users will not see this database at all.

It should be noted that you can omit the URL of infobase. In this case, the text << Any link>> will be displayed in the builder and to manage the infobase, its address will need to be explicitly entered on the mobile device. One configuration can be added to the list several times. In this case, you can specify for each entry your URL of the infobase publication.

In addition to specifying the addresses of information databases, there is a capability to specify the addresses of various web-pages. So, if the mobile client application starts with a blank list of information bases, a web page will be automatically opened, the address of which is specified in the Address of the initial web-page field of the mobile application builder. You can also open this web-page at any time using the Start of operation command of the form menu with a list of information bases. Using this page you can implement several operations:

- User registration in the infobase (or service). This feature may be required if the infobase requires a username and password, and the user is not yet registered. If the mobile client application that is used to access corporate information bases can be registered by the administrator of this infobase, then for the infobase that is publicly available, such registration can be done using a web-page. The address of this page should be specified in the Address of initial web-page property.
- Add a web service link to the mobile client application that provides a list of common information bases. Operation of system in this case is similar to the behavior of the platform for a personal computer. To do this, a hyperlink generated in a special way should be placed on the web-page (using scheme `e1c://` and command `AddToStartCfg`). After adding the link to the web-service, the mobile client will automatically retrieve the list of infobases and display the retreived list.
- Start the necessary infobase, without adding it to the list of infobases of the mobile client application. To do this, a hyperlink generated in a special way should be placed on the web-page (using scheme `e1c://` and command `WS`).

It is assumed that the following script will be implemented on the home page:

1. The user is offered the registration interface in the infobase (or service).
2. If necessary, after successful registration, a link to the web-service for receiving a list of common infobases is added to the mobile client application.
3. Start of the required infobase is performed. In doing so, the user uses the name and password received in the first step of this scenario for authorization in the infobase.

The builder also provides the capability to specify the address of a web-page in the list of infobases. To do this, click the Add button, then in the Type field, specify Web-page, in the Presentation field indicate how this web-page will be presented in the list of infobases, and in the Web-server address field, specify the URL of the added web-page. The C. field (as for the

infobase) defines the initial visibility of the entry in the list of applications of the mobile client.
The web page specified in the list assumes setting the page on which actions are performed, access to which requires authorization, but these actions are not implemented (for any reason) in the infobase (or databases) that the user has access to. An example of such an action is the payment for access to an infobase or service (if access is paid).
Web pages that are specified in the Address of the initial web-page field and in the list of infobases will open in the mobile client application, and not the default web-browser on the mobile device used. On these pages, it is permissible to specify hyperlinks using a scheme e1c://. Clicking on such links is processed directly by the mobile client and allows to add infobases to the list of the mobile client applications.
On the List of common infobases tab, the builder allows to specify on the basis of which resources the mobile client application will generate a list of common infobases. The Use an arbitrary address of the list of the common infobases check box will allow the user to add the address of a web-service that returns a list of common infobases. In the event that the addresses of the web-services of the lists of common infobases are known at the time of the creation of the mobile application, they can be indicated in the list on this tab. It should be remembered that the mobile client uses the same web-service for receiving a list of common infobases as the interactive program of platform start for a personal computer.
The check box Allow adding information bases using QR-code enables and disables the feature of the same name. Remember that this box checking does not mean the obtainment of such feature in the built application. The use of QR-codes in the built application (provided the box is checked) will be accessible if one of the following conditions is met:

- It is allowed using the arbitrary address of the list of the common infobases (Use an arbitrary address of the list of the common infobases check box is checked).
- There is at least one line with empty web-server address in the list of infobases and web-pages (Web-server address column).

If a built application gets the support of QR-codes scanning for the infobases adding the permissions necessary to use the device camera are added to the list of permissions of the application and Add with the help of QR-code command appears in the mobile application interface.
Allow adding infobases using QR-codes check box status can be changed both in the group and in the setting of a certain mobile application but the final "decision" is made during the application building.

If the field Web-server address in the list Addresses of infobases and web-pages or Paths to lists of common infobases contains URL beginning with the line «https://» then the builder allows setting the means to validate the server and client certificates:

- The source of a certain certificate for the server certificate validation can be specified in Certificate validation means field:
  - Do not validate -means that the server certificate will not be validated but the mobile client connection to the server will be encrypted.
  - OS certificates storage - means that the root certificates stored in the certificate storages of the device on which the application to be built will operate will be used to validate the server certificate. The mobile client connection to the server will be encrypted.
  - Specified certificate -means that the root certificate specified in Server certificate field will be used to validate the server certificate. The mobile client connection to the server will be encrypted.
- The root certificate used to validate the server certificate is specified in Server certificate field. The field is enabled only if Specified certificate value is selected in Server certificate validation means field.
- The certificate used for the client certificate validation is specified in Client certificate field.

The certificates preliminary uploaded to the catalog Certificates for HTTPS work are specified in Server certificate and Client certificate fields. The uploaded certificates (for convenience of use) are divided into the root certificates (with For server certificate validation checked box) and client certificates.

29.8.4.2.5. e1c scheme:

Command description

On web-pages that are specified in the Address of the initial web-page property and in the entry of Web-page type (in the builder itself and in the list of common infobases), you can specify URL (for example, in the `<a>` element) that belongs to the `e1c` scheme .

Such an address will be processed directly by the mobile client. This URL allows to specify commands (and parameters) for the command-line to start the mobile client.

In addition to the general, several special commands are implemented:

- AddToStartCfg command. This command is used to add links to the web-service (or file) of the list of common infobases to the mobile client application. Service is described by command parameters. The command is as follows:

```
AddToStartCfg[&N=<user>][&P=<password>][&Title=<heading>][[&Interne
tService=<URL>]|[&CommonInfoBases=<URL>]]
```

  this command uses the following parameters:
  - N=<user> - allows to specify the user on whose behalf the service will be accessed.
  - P=<password> - password of the user specified in the parameterN.
  - Title=<heading> - presentation of a web-service in the list of Internet-services for receiving lists of common infobases of the mobile client.
  - InternetService=<URL> - URL of the service for receiving a list of common infobases. Once added, the mobile client will retrieve the list that the service returns. In one command can be used or parameter InternetService or CommonInfoBases.
  - CommonInfoBases=<URL> - contains the full name of the file with the list of common infobases (.v8i file). After adding, the mobile client will retrieve the list contained in the specified file. In one command can be used or parameter InternetService or CommonInfoBases.
- AddByQRCode command. The command starts QR code scanning. The command is as follows:

```
AddByQRCode[&SkipDetails]
```

  this command uses the following parameters:
  - SkipDetails - not to show the detailed information on the added infobases to the user.
- AddInfoBase command. The command adds the infobases to the list of infobases of the current mobile application. The command is as follows:

```
AddInfoBase[&WS=<connection
string>][&Run][&DisplayAuthDialog][&DisableUseBiometrics][&UseBiome
trics[-|+]][&DisableRememberMe][&RememberMe[-|+]]
```

  this command uses the following parameters:

- `WS=<connection string>` - describes the string of connection to the infobase which is needed to be added to the list of infobases.
- `Run` - run the infobase after it is added.
- `DisplayAuthDialog` display an authentication dialog box upon initial startup.
- `DisableRememberMe` makes Remember me check box invisible in the authentication dialog box.
- `RememberMe[-|+]` initial state of Remember me check box. It is cleared by default.
- `DisableUseBiometrics` makes Use Biometrics check box invisible in the authentication dialog box.
- `UseBiometrics[-|+]`initial state of Use Biometrics check box. It is cleared by default.

QR-code content

Some certain information shall be encrypted in the QR-code for it to be successfully decoded by 1C:Enterprise mobile version. Different information can be encrypted in QR-code depending on what client application QR-code is intended for.

Mobile platform of the developer

1. URL of the configuration location. This URL results from the mobile application publication on a web-server.

Example of a text for QR-code generation:

```
http://host/cfgFolder
```

2. String of connection to the infobase of `ws=URL` type. `WSN` and `WSP` commands can be specified in the next strings. These commands describe the parameters of authentication on the web-server where the configuration is located.

Example of a text for QR-code generation:

```
ws=http://host/cfgFolder
wsn=<user name>
wsp=<user password>
```

Mobile client

1. Infobase URL. Such URL allows the address with the help of the thin client of the platform for the PC or with the help of web-browser.

Example of a text for QR-code generation:

```
http://host/folder/dbName
```

2. Infobase list file item (*.v8i) describing the infobase. However, you can use N, P, WSN, WSP, DisplayAuthDialog, DisableRememberMe, RememberMe, DisableUseBiometrics, UseBiometrics commands. N and P commands describe the parameters of authentication in 1C:Enterprise application. WSN and WSP commands describe the parameters of authentication on the web-server where the application is published.

Example of a text for QR-code generation:

```
[Text for infobases list]
Connect=ws="https://host/dbName/";
MobilePublicKey=AC05BDFD484C8F21C078A2E564765057
N=<IB user>
P=<IB user password>
WSN=<web-server user>
WSP=<web-server user password>
```

If the formed QR-code contains the commands not supported by the mobile clients these commands will be omitted.

3. Infobases list file item (*.v8i) describing the list of common infobases. N, P, WSN, and WSP commands can be used. N and P commands describe the parameters of authentication in 1C:Enterprise application. WSN and WSP commands describe the parameters of authentication on the web-server where the application is published.

Example of a text for QR-code generation:

```
[List of common bases]
InternetService=https://host/service/
WSN=<web-server user>
WSP=<web-server user password>
```

Examples

The parameters of various links will be considered in this section. It should be borne in mind that the links will be given in a type that is as easy to read as possible, and not for placement in a real attribute href. For placement in the attribute, bring the string to the correct type.

```
<a
href="e1c://?AddToStartCfg&Title="Application"&N=User1C&InternetSer
vice=https://my.site.com/is/hs/">
```

Going to this address will lead to the adding of mobile client's application a service for receiving a list of common information bases, for access to which, the user User1C will be used.

```
<a href="e1c://?WS=https://test.app-site.com/app-name">
```

Going to this address will lead the mobile client to connect to the application, which is published at address https://test.app-site.com/app-name. At the same time, this application will not be added to the list of infobases of the mobile client.

```
<a href="e1c://?AddToStartCfg&InternetService=
https://my.site.com/is/hs/&WS=https://test.app-site.com/app-name">
```

Going to the specified address will result in two actions:

1. An internet-service for receiving a list of common infobases, published at https://my.site.com/is/hs/ address will be added to the mobile client application and receiving of this list will begin.
2. After performing this action, the mobile client will be connected to the infobase published at https://test.app-site.com/app-name.

29.8.4.3. Specific application

In order to get a specific application, you need to create a specific application in the group created earlier. When creating an application, all the fields that can be filled automatically or can be "inherited" from the parent group will be filled. Overwhelmingly, "inheritance" is accomplished by setting the value specified in the group in the desired field.
However, there are a few exceptions:

- Mobile version used for building. If a certain version is specified in the parent group -this version will be used in the application. If the Mobile platform field is not specified in the group, then the builder will try to select the mobile version with the maximum number that meets the following conditions: not marked for deletion, not deleted from the infobase and suitable for the current building mode (поле Build). If in the base of the builder there are no suitable versions, then the user can try to specify the necessary version himself. If the necessary versions were not picked up, - the building will be impossible.
- Configurations used for building. For each group of configurations that are specified in the parent group (Configurations and version tab), the most recent loaded configuration is searched for, which will meet the

following conditions: not marked for deletion, belongs to the selected configuration group and is suitable for the current build mode (Build field). It is believed that the larger the item code, the more "fresh" version of the configuration. The builder selects the configuration with the largest code. As a result, when creating a new application, the latest configurations (of the necessary ones) loaded into the builder will be included in it.

- On tabs corresponding to the mobile operating systems used, the full application identifiers are set. In a specific application, you cannot change the full application ID. The change is possible only through the change of application identifiers in the group and will be valid only for newly built applications.
- For the mobile application, the version and build number of the mobile application is determined and indicated.

On the Configurations and representation tab, you can specify which configurations will be included in the application being built, as well as the presentation of the application being built in all supported languages. In addition to automatically added configurations, on the Configuration of the application tab, you can add an arbitrary list of configurations.

The presentation of a mobile application is determined as follows:

- All localization languages that are present in all used configurations of this mobile application are determined.
- For each localization language, a presentation is generated. The presentation of the configuration in a specific language will be the presentation of the configuration in which the language was encountered for the first time (in the traversal order).
- Then Application representation list of the mobile application group is analyzed. The representation for the matching languages is filled in from the group.

Then on the Presentations of the mobile application tab, you can edit presentations for all found languages. The presentation language of the entire application by default (the name of the application icon on the mobile device) is selected in the Language field. If it is necessary to change an application representation permanently -it is recommended to be done in the list Application representation of the mobile application group.

The remaining parameters of the built application, if necessary, can be changed relative to the settings specified in the group.

29.8.4.4. Build result

The Build results tab contains the results of building a mobile application.

The Date and time of the build of the mobile application field contains the date and time of the last successful build of the current mobile application. Last unload date in … fields contain the date and time when the built application was unloaded into the corresponding application store (supported by Google Play and Apple AppStore).
Below, in the table, there is a list of artifacts that were generated during the last build. For each artifact, its type and the presence of the application itself and the build log are indicated. If the record is displayed in a pale color, - it means it is outdated and all files are deleted from the database. The following build artifacts are available:

- For OS Android:
  - Mobile application for ARM architecture.
  - Mobile application for ARM64 architecture.
  - Mobile application for x86 architecture.
  - Mobile application for x86_64 architecture.
  - Mobile client for ARM architecture.
  - Mobile client for ARM64 architecture.
  - Mobile client for x86 architecture.
  - Mobile client for x86_64 architecture.
  - Build log for each of the above artifacts.
- For OS iOS:
  - Mobile application for OS iOS.
  - Project to build a mobile application using Xcode.
  - Mobile application for OS iOS simulator.
  - Mobile client for OS iOS.
  - Project to build a mobile client using Xcode.
  - Mobile client for OS iOS simulator.
  - The build log for the mobile application/mobile client and the project for Xcode is common. The build log for the emulator is separate.
- On Windows OS:
  - Mobile application for ARM architecture, telephone.
  - Mobile application for x86 architecture, telephone.
  - Mobile application for ARM architecture, tablet.
  - Mobile application for x86-64 architecture, tablet.
  - Mobile application for x86 architecture, tablet.
  - Mobile client for ARM architecture, telephone.
  - Mobile client for x86 architecture, telephone.
  - Mobile client for ARM architecture, tablet.
  - Mobile client for x86-64 architecture, tablet.
  - Mobile client for x86 architecture, tablet.
  - Build log for each of the above artifacts.

This tab provides the ability to retrieve all build results (and applications and build logs) from a database of the builder in one archive. To do this, you should run the command form More - Get the applications. As a result, a .zip archive will be generated that contains all the files received as a result of the build. This archive is proposed to be saved to the client computer. The archive has the following structure:

- The Android folder contains Android application files. The file name suffix determines the processor architecture for each file.
- The iOS folder contains application files for OS iOS. A file with the .zip extension contains a project archive for self-build using Xcode. The file with the .ipa extension contains an executable file suitable for sending to the device or to the application store. The file, whose name ends in "-simul.zip", is a mobile application for operation on the emulator.
- The Windows folder contains application files for Windows OS. The file name suffix determines the processor architecture for each file and the implementation option (phone or hybrid device). These files will have the extension .appx.
- The Logs folder contains the build logs of all files.
- The name of each file (including the build log files) is the full application ID for corresponding operating system.

If necessary, each artifact of the build can be downloaded from the infobase. To do this, you need to get to the desired row of the results of the build and click the Get application button. If you need to see the build log for this artifact, - in the same row, click the Show log button.

29.8.5. Build the application

Build a mobile application begins with the command Build application. This command is available both in the form of a specific application and in the list of mobile applications.

Before building a specific application (directory element) must be recorded.

The building is performed as follows:

1. Initial checks are performed:
   - It is checked that at least one mobile operating system is indicated in the built application.
   - It is checked that at least one configuration is specified.
   - The availability of the required set of files for the selected mobile version of 1C:Enterprise (mobile application or mobile client) is checked.
   - If at least one check is not completed, - the building is impossible.
2. The previous building results are being removed from the database.

3. A cache is generated for the used mobile version of 1C:Enterprise. The cache is generated in the platform directory of the directory specified as the working directory and builder cache in the parameters settings of the builder.
4. The cache of graphic information used is generated. The cache is generated in the pictures directory of the directory specified as the working directory and the builder cache in the builder parameter settings.
5. The cache of used audio files is generated. The cache is generated in the sounds directory of the directory specified as the working directory and the builder cache in the builder parameter settings.
6. The cache of add-ins used in the mobile application is generated. The cache is generated in the add-ins directory of the directory specified as the working directory and the builder cache in the builder parameter settings.
7. The actual unload to the cache is performed only if the required files are not on the hard drive or they do not correspond to the files in the database.

If you are building for Android OS, then the components that will be used for building are selected. The selection algorithm is as follows:

1. The list of available SDK versions (API Level) is determined.
2. If the installed versions are earlier than the version specified in the requirements for the mobile version being used, - the building becomes impossible. If the required version is not specified in the version used, the minimum SDK version for building using Apache Ant is 24, and for building using Gradle - 26.
3. From the found SDK versions, the oldest version is selected.
4. The dependencies of the selected version determine the associated tools and the capability of their use:
   - The SDK Tools version is determined by the requirements of the selected API Level.
   - The version of the SDK Platform Tools is determined by the requirements of the SDK Tools.
   - The Build Tools SDK version should match the selected API Level.
   - The mismatch of the required versions is an error blocking the building.
5. It is determined which building technology the selected SDK (API Level) uses: Apache Ant or Garadle.

The above algorithm is performed for each of the two settings of access paths to the Android SDK, which are specified in the builder parameter settings. Then it is determined which building technology is required by the used mobile version of 1C:Enterprise (the version that is specified in the settings of

the application being built). If specified mobile version of 1C:Enterprise can be built within the border specified in the infrastructure settings, the building is continued. If the infrastructure is configured - inadequately, the building is blocked.

Also, as part of the preparatory actions, the availability of the building tools needed in the case of building mobile applications for specific mobile operating systems is checked. In general, the following checks are performed:

- The presence and correctness of JDK version is performed in any case.
- The capability to build the desired type of mobile application using the specified mobile version of 1C:Enterprise. In other words, it is impossible to build a mobile client application using the mobile version of 1C:Enterprise earlier 8.3.12.
- When specifying building for Android OS, in addition to the general checks, it is checked the following:
  - SDK version (according to the algorithm described above).
  - Availability of developer certificate.
- When specifying building for OS iOS, in addition to general checks, it is checked the following:
  - Correctness of specified version of mobile application.
  - Availability of PuTTY utilities.
  - Parameters setting for connecting to Apple computer.
- When specifying building for Windows OS, in addition to general checks, it is checked the following:
  - Availability of the Windows SDK on the computer of the builder server.
  - Availability of developer certificate.

The texts of the permissions requests necessary for the mobile application are set for the application on the whole. But each configuration used in the mobile application may have its own texts of the permissions requests. The builder uses the following algorithm determining the list of texts of permissions requests (in different languages) to be used in the mobile application to be built:

- The list of languages of the mobile application to be built is formed on the basis of all configurations of the application to be built.
- The text of the permission request is formed for each language. To do it, the builder iterates the used configuration in the order of their location in the list and tries to get the text of the permission request. The first not empty text of request for each language will be used for the application.
- Then, all languages where no text of the permission request is specified are deleted for each permission. The text from the platform on this language (if the deleted language is included in the languages of the mobile

version localization) or English (if the deleted language is not included in the languages of the mobile version localization) will be used for such languages.

- If a mobile application contains the configurations with the specification of the texts of the permissions requests and the configurations without the specification of such text then only the configurations containing the texts of the permissions requests will be used for the resulting requests text formation.
- If no configuration of the mobile application uses the permissions request texts - then all texts of the permissions requests will be taken from the platform.

The resulting list of permissions request texts will be placed in the application to be built.

Each external component used in the mobile application to be built shall be built for all operating systems for which the mobile application is built. If the mobile application is to be built for iOS and Android then the external component shall be built for iOS and Android. But the situations when an external component is written only for one mobile operating system are possible. In this case, the mobile application builder will write the warning that a certain mobile component is not built for a certain mobile operating system in the mobile application building log. But the building will be continued anyways. An exclusion will be formed in the case of an attempt to use an absent external component on a mobile device (in the built application).

After completing all the preparatory actions, a building of all the required artifacts is started. Each artifact is built by its background job. The number of simultaneously launched background jobs is determined by the number of processor cores of the computer acting as the server for 1C:Enterprise mobile application builder. If the number of background jobs necessary to be launched exceeds the number of cores the number of launched background jobs will correspond to the number of cores and the remaining background jobs will wait in queue for the completion of already launched building tasks. The waiting background jobs will be launched as and when the earlier launched background jobs are completed. If building was not ordered for any mobile operating system (the corresponding check box in the settings of the mobile application is not checked), then the corresponding background jobs are not started.

When building an iOS application, building an .ipa file on Apple computer is only done if the parameters for connecting to Apple computer are specified in the builder settings. On the Apple computer specified in the builder settings, the Xcode package should be installed and set correctly.

When building applications for Android OS using Gradle, the gradle directory is generated in the working directory of the builder. It occupies a significant amount (several gigabytes), however, it is not recommended to delete it after each building, since it will be generated again at the next building, and besides, the presence of this cache significantly speeds up the application build process.
At the time of building, a window is displayed showing the time that the building has already taken and the state of each of the started building processes.
Each background job creates its own working directory in the builder working directory. The name of this directory is a textual representation of the unique ID received by the platform. This means that the name of the working directory of each background job is unique each time it is executed. After the corresponding background job is finished, - the working directory of this background job is deleted.
After the end of the building, the results are written to the database.
If building for all required artifacts is completed, - the successfully built mobile application becomes unavailable for changes. It can only be rebuilt and see the parameters that were used for building. This is done to minimize the possibility of getting an already built mobile application, which differs from the parameters that were set during its building.
The working directory of builder can be completely cleared at any time (unless at this point the mobile application is being built). After that, the next several building processes can run longer (until the build cache is again filled with the necessary information).
The build log can provide help in solving problems with building or operation of the application. This log contains both the information of the mobile application builder itself and the information of the utilities that are started by the buldir in the process of operation. Standard output (stdout) and standard error output (stderr) of the utilities started are logged.

IMPORTANT! When contacting 1C Company with problems related to the operation of the mobile application builder, the **build log** should **be attached immediately and completely**. It will significantly simplify the problem diagnostics.

29.8.6. Application publishing

For built applications, the builder provides capability to download to the application store. Supports operation with Google Play and Apple AppStore.

Downloading can be performed by the command Download to stores from a specific application or a list of mobile applications. Directly unloading is performed from the dialog box, the appearance of which is shown above.
For unloading to be possible, the following conditions should be met (for each store):

- All required applications should be built.
- The builder specifies the parameters for access to the application store:
  - For Apple AppStore:
    - in the provider's card (Desktop - Service - Provider parameters setting - Parameters for OS iOS - Access to iTunes Connect).
    - parameters for access to Apple computer are set (Desktop - Service - Setting of builder parameters, group of parameters for accessing Apple computer)
  - For Google Play: in the provider's card (Desktop - Service - Setting of provider's parameters - Parameters for Android OS - Access to provider's console).

If the download is impossible for some reason, - these reasons will be generated in the Diagnostics field for each application store.
For the Google Play store, you can specify which application configuration is loaded into the store (Configuration field).
Clicking the Download application button starts the download process. If the download failed, then pressing the Log for … button will allow to view the download log. It should be remembered that the download log can only be viewed immediately after the download attempt. After closing the form Download to application stores, logs will be lost.

29.8.7. Additional features of the mobile application builder

29.8.7.1. Interface of interaction with the service of the 1C Company for sending push-notifications

29.8.7.1.1. General information

1C Company has implemented an auxiliary service for sending notifications. This service has its own user interface. At the same time, the mobile application builder allows to perform a number of operations without visiting the website. These operations include:

- Registration on the website https://pushnotifications.1c.com.
- Registration and cancellation of registration of the recipient of notifications in the service.
- Attach the feature to receive notifications through supported services to the registered application: APNs, GCM, FCM, WNS.

To access the form of operating with the service, you must open the form for editing the group of mobile applications and in this form click the Operation with "1C" service button. As a result, the form of operation with the service will be opened.

Further, the actions performed will be discussed in more detail.

29.8.7.1.2. Registration on the website

To connect to the 1C Company service, you must enter a username and password. To do this, click the Specify parameters of access to the service button. For each user of the builder, you should register your service access parameters.

If the user does not have access to the 1C Company service, then, there is a capability to register with the service directly from the builder. To do this, in the registration form of access parameters, click the Register in the "1C" service button.

In the registration process, specify a valid mobile radio telephone number to receive a confirmation code, then enter the received code and other parameters into the form.

After the registration is completed, it will be possible to register the application in the service.

29.8.7.1.3. Managing application

The dialog box provides the capability to register a mobile application that will receive notifications. To register, click the Register application button. The name of the application will be the text from the Application description field of the dialog box. By default, this field is populated with the name of the group of mobile applications.
If the use of the application through the 1C Company service is no longer planned, then you should click the Delete application registration button. It should be remembered that the deletion of registration can not be restored. Even if you re-register, from the point of view of the service, this will be a new application and the previously specified data for the notification services will have to be reinstalled.

29.8.7.1.4. Operation with services

After completing the registration of the application in the service, you can connect to the services application of sending notifications. The current connection status is displayed at the bottom of the dialog box.
Clicking the Attach... button causes the builder to attempt to connect the corresponding service. Depending on the attached service, you may need to enter some additional information.

29.8.7.2. Debugging in-app purchases

29.9. Application store publishing

29.9.1. General information

Preparing an application for an application store is done in the same way as checking a mobile application before publishing it in the application store.

29.9.2. For OS iOS

Before sending the application to the Apple AppStore, you should create an application in iTunes Connect and fill in all the required information about the application. Then click the Ready to Upload Binary button, and the status of your application in iTunes Connect should change to Waiting For Upload.

29.9.2.1. From the archive of the project

After generating the file with the application archive for OS iOS, copy the archive file to your Mac computer and do the following:

- Unzip the file with the project of the mobile application <Application ID>.zip on Mac computer.
- Open the project of the mobile developer platform in the Xcode system (double click on the file 1cem.xcodeproj or select file 1cem.xcodeproj using the command File - Open in the Xcode system).
- Using the Xcode menu command Product - Edit Scheme... it is necessary to open the settings dialog box of the current startup scheme. In the opened dialog box:
  - In the Scheme property, select 1cem value.
  - In the Build configuration property, select Release value.
  - Click OK.
- Create an application archive using the Xcode command Product - Archive.
- Open tool Organizer using menu command Xcode Window - Organizer.
- In the top panel, switch to the archives Archives.
- Select the created archive and click the Distribute... button.
- Follow the instructions of the assistant.

29.9.2.2. Binary file (.ipa)

Using mobile application builder

Open the list of mobile applications for which you need to download the application, and select the Download to stores... command. The builder will post the selected mobile application and inform the user about the capability or impossibility to complete the download. If a download is available (Download to Apple AppStore check box is selected), then by clicking the Download button, the bulder will attempt to perform the download. With this in mind, the username and password for accessing iTunes Connect is used. It can be specified in the provider settings dialog box on the Parameters for OS iOS tab (if set by the builder administrator) or in the Service dialog box - Parameters of access to iTunes Connect (if the publication can be done by a regular user).

Publishing is done using the Mac computer on which the mobile application was built.

At the end of the download, the builder will report the status of the download. If there is a suspicion that the download has ended unsuccessfully, - it is recommended to press the Log for AppStore button. The document containing the download log will be opened. The log is not saved in the infobase and will be lost after the loading window is closed.

Manual

Sending a prepared application package (.ipa) to Apple App Store is done using the Application Loader utility, which is included in the Xcode development kit. To start the application loader utility, first run Xcode, then select the menu Xcode - Open Developer Tool - Application Loader.

In the loader, enter your account information in the Apple Developer Program. Then select Deliver Your App. In the opened dialog box, select the application you want to send. If it is not on the list, then iTunes Connect for this application is not filled with the necessary information. Then click Next, then Choose, and select required .ipa file. To send a file, click Send and wait for completion of the process.

29.9.3. For OS Android

Using mobile application builder

The first download of the application to the Google Play store should be done manually. Subsequent downloads can be performed using the mobile application builder.

Open the list of mobile applications for which you need to download the application, and select the Download to stores... command. The builder will post the selected mobile application and inform the user about the capability or impossibility to complete the download. If the download is available (Download in Google Play check box is selected), then specify the configuration in which the application will be placed in the Configuration field, and then click the Download button. The builder attempts to perform loading. The executable files for both architectures are being loaded: ARM and x86. To access Google Play, the parameters that can be set in the dialog box of the provider parameters setting on the Parameters for Android OS tab are used. This setting can only be performed by the builder administrator. The access algorithm is described in the mobile application builder help.

At the end of the download, the builder will report the status of the download. If the download failed, - it is recommended to click the Log for Google Play button. The document containing the download log will be opened. The log is not saved in the infobase and will be lost after the loading window is closed.

Manual

To display an application on Google Play, create an application on Google Play portal, fill in the required information about the application and download the .apk file(s).

When creating an application, select Prepare Store Listing. After filling in the application information, jump to the APK tab and switch to advanced mode by clicking the Switch to advanced mode button. Next, for each processor architecture (ARM and x86), click Upload new APK to Production, and select the appropriate application file. Only advanced mode allows to download application files for different processor architectures. As a result, both applications will appear in the list with appropriate ARM and x86 architectures. Now press Publish now button.

29.9.4. On Windows

Using mobile application builder

Mobile Application Builder does not support publishing applications to the Windows Store.

Manual

Before you build an application in the mobile application builder, reserve the ID of your application in the Windows Store. To do this, go to the store information panel (https://appdev.microsoft.com/StorePortals/).

To create an application, select + Create new application. Now you need to enter the planned application ID and click the Check availability hyperlink. If the entered name is busy, it will be impossible to reserve this name, which will be reported to the right of the field with the application ID. If the name is available, then click the Reserve application name button.

After building the mobile application, upload the resulting files into the created application from the information panel. To do this, in the information panel, click the created application with the mouse and select the Send item in the left menu, and then click the Start sending button.

In the opened form, set the application properties, prices, product description, age limit, etc. Then click the Packages hyperlink and download the .appx files created by the mobile application builder. If the download was successful, - click Save button. After filling in all the sending parameters, click the Send to store button and wait for the results of the check.

Chapter 30. Collaboration system

30.1. General information

Collaboration system - is a mechanism that allows client applications, the server and users of one or several infobases to interact with each other. One infobase (or data area) that is uniquely identified by the interaction system is called an **application**. Collaboration system is designed to simplify the automation of business processes of the enterprise and provides users with an easy way to communicate within the border of their official duties. Only application users can use collaboration system. It does not need to register separately in the collaboration system. User authorization in the collaboration system is carried out by the 1C:Enterprise.
Collaboration can be:

- Non interactive - Only subsystems of an application that do not involve interactive collaboration are involved in the exchange of information. Such interaction can occur if it is necessary to tpass any information between the subsystems of the application during the execution of any activity.
- Interactive only - when all participants of collaboration are real people. For example, two users discuss capability of providing a discount to the counterparty and its (discounts) size.
- Mixed - when collaboration is carried out between a real person on the one hand and an application subsystem on the other. The simplest example of such collaboration is a robot that responds to certain requests.

With interactive collaboration, users of the application can communicate with each other in real time using text messages and video calls. The term "video call" refers to any possible option of communication between two users, which includes both a voice communication channel and a full-fledged video call (both voice information and image). Video calls can only be interactive (only real people participate in them). During a video call, you can share your screen. Screen can be shared, only if a video call window for this user covers the entire display area. Otherwise, instead of screen sharing Screen Display icon is shown.
Collaboration of users is carried out within the border of conversations. **Conversation** - this is a set of messages related to one subject. Anything can be

the subject of conversation: document, weather, execution of an action, etc. The conversation can be of two types:

1. **Noncontext** conversation - conversation that is not associated to the infobase object. In this conversation, any users of application who are selected by the participants in the conversation can take part. Noncontext conversation can be any number.
2. **Context** conversation- this is a conversation that is associated to the application object. Only those users of application who have access to the object under conversation can participate in such conversation. Each object can have only one conversation.

Various files can be attached to the collaboration system messages. To do it, set the collaboration server work with the external storage. The external storage must support Amazon S3 protocol. -It is impossible to attach files if the collaboration system server is not connected to the storage. If for file transfer purposes 1C:Dialog service is used, file transfer and storage mechanism has certain restrictions which are listed at https://1cdialog.com/ru/pricing/.

Access to messages is possible using the Notification center, the Conversation element of the open items panel, a message panel in the forms of system objects (for context messages) and using the program interface provided by the global context property `CollaborationSystem`.

The collaboration system is implemented in a client/server architecture and consists of the following parts:

- **Client** collaboration systems is an integral part of the 1C:Enterprise.
- **Server** is a specialized server represented by the service 1cDialog.com or product instance 1C:Collaboration server. Collaboration content is stored on the server of the collaboration system. Information about video calls (including the facts of the commission) is not stored in the collaboration system.

Messages are exchanged between the client and the collaboration server using the WebSocket protocol. This protocol provides secure data transmission that allows to securely pass messages through which users communicate, both on the local network and over the Internet. If access to the server of the collaboration system is carried out through a proxy-server, then the proxy-server settings are received from the configuration of the operation system (automatic setting scenario is not supported) or from the configuration file inetcfg.xml. NTLM authentication on the proxy-server is not supported. For video calls, a peer-to-peer connection is used, and the connection is established through the server of the collaboration system.

1C:Enterprise system provides a user interface for exchanging messages (of any supported types), and also provides a software interface that allows to perform certain operations, including the following:

- register or cancel the registration of the application in the server of the collaboration system;
- combine or cancel the association of applications, as well as receive information about the combined applications;
- get and set data allowing the application to collaborate with the server of the collaboration system (registration data);
- operate with users of the collaboration system;
- create and receive conversations and messages;
- implement specialized messaging tools.

Operation with multiple copies of the same infobase is not supported. In that case, if the infobase is copied, then such a database will be able to attach to the service, however, during operation, various features will be observed, for example:

- Conversations and notifications will be available in all copies of infobases.
- There will be problems when managing context conversations for those objects created after cloning the original infobase.
- There will be problems when managing recipients of messages if users in the infobases were created after cloning.

At the same time, it is possible to combine several applications into a single information space (from the point of view of the collaboration system). To do this, register each application in the collaboration system, and then combine the necessary applications. Combined applications have a single list of users (from all combined applications). When combining applications, you can specify how the users of the applications being combined will be matched.

Video calls are supported:

- In the thin client when working under the management of:
  - Microsoft Windows 7 and earlier;
  - macOS of any supported version.
- In web client:
  - At operation in Google Chrome web-browsers, Mozilla Firefox of version 52 and older.
  - At operation in Google Chrome web-browser, video calls are supported only if the infobase is accessed using the HTTPS protocol.
  - Screen sharing:
    - This features is supported when you use Google Chrome and Mozilla Firefox.
    - It is available, only if HTTPS protocol is used.
    - If you use Google Chrome web browser, both screen and specific tab can be shared.

- Google Chrome requires installation of a web browser extension and enable it to make screenshots.

30.2. Collaboration system arrangement

Collaboration system operation is built on a client-server data exchange between the 1C:Enterprise system and the collaboration system server. At the same time, both the client and server application of the 1C:Enterprise system act as clients of the collaboration system server. Collaboration is organized around the application of the collaboration system (infobase or data area).
To start using the collaboration system, an application whose users will participate in this collaboration should be registered with the system service. During registration of the application, the following information is created (and saved) in the collaboration system service:

- Application UUID in the service. This ID is stored both in the infobase itself and on the collaboration system server. This ID cannot be received and set singly.
- Cryptographic keys that are used to ensure the safe transfer of messages between the client and the collaboration system server. These keys are stored both in the application and on the collaboration system server. Each side of collaboration stores its own set of keys.
- The service also stores the name of the registered application and the email address that is used when registering the infobase. The email address that is used when registering the application with the service will mean the subscriber of the service. One service subscriber can register several applications. This subscriber will be considered the **owner of the subscriber**.

In the future, the address of the collaboration service, the application name, e-mail and activation code (collectively) will be called "application registration parameters in the service". At the same time, the application registration parameters in the service, the application ID in the service, and the cryptographic keys used for the exchange will be collectively called "application registration data".
In order to terminate the operation of collaboration system, it is necessary to unregister the application in the service. In this case, the application removes the cryptographic keys used to provide the exchange and, as a result, the exchange of messages becomes impossible. When you re-register the same application, a new key pair will be created. As a result, the messaging will be

restored and the entire message history will be available. For correct restoration of messaging, it is important that the application remains unchanged the UUID of this application in the service. If this rule is violated, - it will be impossible to restore access to the message history.
In order for the application to be able to register with the collaboration system service, users should be specified in it. Otherwise, the application will not be able to register with the service and, as a result, it will be impossible to use the capabilities of the collaboration system in it. Only those users of the application who are registered in the collaboration service can participate in the process of information exchange. For each user of the application a pair user is created in the collaboration system. Creation of such a user is performed at the first user authentication of the application after registration of this application in the collaboration system server or using the methods of the 1C:Enterprise language. With subsequent user authentication, the name and full name of the user of the infobase are automatically updated in the collaboration system. Users association (application and collaboration service) is performed by the UUID of the 1C:Enterprise infobase. If a different user list synchronization algorithm is required, - the application developer must implement it himself.

30.3. Interactive collaboration

30.3.1. Users parameters

After connecting the application to the collaboration service, appears a capability to specify additional parameters for the user who is registered in the collaboration system.
The storage of the full username, picture (avatar), phone number and email address is supported.

30.3.2. Kind of conversations

30.3.2.1. Noncontext conversations

Noncontext conversations are not associated to any application objects. Any user can create an arbitrary number of such conversations. Noncontext conversations can be of two types: with any number of participants (option of "group conversation") and only with two participants (option "one to one"). When creating a conversation, its participants are the user who created the conversation, and those users that he specified when creating the conversation. Any conversation participant can:

- View the conversation
- add messages to conversation;
- the following features are only available for group conversation (and are unavailable for one-to-one conversations):
  - change the title (subject) of the conversation;
  - add new conversation participants (any users of the infobase);
  - Leave the conversation

If the last member leaves the conversation, the conversation becomes unavailable. In this case, on the collaboration server, the contents of conversations are saved. Noncontext conversation is visible only for conversation participants.

Creation of noncontext conversation is performed from the main form of the collaboration system, which becomes available after the application is registered in the collaboration service.

The main form of the collaboration system is always the second (after the home page) form in the open items panel. This form cannot be closed and dragged to another place on the panel. If there is no home page in the application, the basic form of the collaboration system becomes the first form in the open items panel.

When creating a new conversation, you are invited to specify the message subject and participants. The initiator of the conversation is automatically added to the list of participants.

To create one-to-one conversation, specify the user with whom you want to create a conversation in the search form line Conversations. For such conversations, the subject and the components of participants are not indicated. The "one to one" conversation subject cannot be set, and the components of participants is automatically formed by the system.

The conversation list shows all noncontext conversations for which the user has notifications. If there are less than 50 of such conversations, then the list is supplemented with noncontext conversations in which the user is a participant and for which there are no notifications for the user. These conversations are added to the list in descending order of the date of the last message in the conversation, until the size of the list reaches 50 items.

Search operates in the conversation list by subjects of conversations and usernames. Search in the message list only operates by message of the selected conversation. The capability to search for conversation on the message text is not supported.

When new messages appear in the conversation, the conversation participants receive notifications.

30.3.2.2. Context conversations

Context conversations (as already mentioned earlier) are associated with application objects. If an application is attached to a collaboration service, then for forms of elements of reference types and forms of information register records, the command of the form Conversation becomes available. For each object, there is only one context conversation. All users of the application (or applications) who have the Read permission for the object under conversation can take part in the conversation.

Context conversation is performed in the form of the object under conversation (on the right side). This conversation has no title. Opening and closing the list of messages is performed by cyclic clicking the Conversation button. Despite the fact that all users of the application (with appropriate rights) can take part in the conversation by default, there is a capability to select recipients of the message when creating a message in the context conversation. In this case, the recipients are selected in order to display notifications of the new message. Other users will be notified only if they sign up to receive notifications on new messages in this conversation.

The Conversation command is not displayed in forms that block the owner window or the entire interface. These forms are usually used to display and edit data that does not have independent application value; therefore, there is no conversation mechanism for them.

For forms that do not block the owner window or the application interface, there is a capability to control availability of the Conversation command in the form. Control is performed using the managed form property ConversationsRepresentation . Setting this property to Not display value will result in a situation where Conversation command will not be available on the form.

However, it should be borne in mind that the context conversation is available only in the form, the navigation link of which indicates the data stored in the infobase or command.

30.3.3. Kind of messages

30.3.3.1. Text messages

The main mechanism in the collaboration system is text messaging. Text message size is limited to 4,000 characters. A text message can contain either text or formatted string (FormattedString object). A formatted message can be used, only if it is generated programmatically. If a formatted string is defined for CollaborationSystemMessage.Text property, format

remains unchanged, while all images (if any) are deleted from the string. The message format will remain unchanged in any of the following:

- push notifications;
- notification center;
- conversation forms.

In any other scenario, message text is displayed unformatted. If a formatted message is displayed, no hyper links are recognized by default, while emoticons are recognized for the following combinations of symbols and emoticons only (Unicode special characters):

Code	Text alternative	in 1C:Enterprise language
U+1F610	:|	`Symbol(55357) + Symbol(56848)`
U+1F600	:D	`Symbol(55357) + Symbol(56832)`
U+1F642	:)	`Symbol(55357) + Symbol(56898)`
U+1F641	:(	`Symbol(55357) + Symbol(56897)`
U+1F62E	:o	`Symbol(55357) + Symbol(56878)`
U+1F61B	:P	`Symbol(55357) + Symbol(56859)`
U+1F609	;)	`Symbol(55357) + Symbol(56841)`

30.3.3.2. Video calls

To use video calls, the following requirements must be met:

1. Collaboration server should allow video calls for the subscriber who registered the application.
2. Client application, which is used to access the application, allows to use video calls.
3. Computers of users - of video call participants should be equipped with microphones and speakers. If a video call participant has a camera, their video feed is displayed to the other participant.

The video conferences with more than two participants are supported in the collaboration system. A video conference participants can be added and deleted at any time.

The maximum number of video conference participants is limited by the collaboration server settings. The maximum default number of subscribers participating in video conference is 10.

30.3.4. Notifications

To inform users about new messages, the collaboration system uses notifications. Notification is generated for each new message within the border of each conversation. Notifications are sent to:

- when adding a message to a context conversation - to all users that the message's author has indicated as addressees, as well as users who have turned on the monitoring mode for this context conversation (taking into account the notification rules);
- when adding a message to a noncontext conversation - to all participants in the conversation (taking into account the notification rules);
- with a missed video call - to the user who missed the video call.

Conversation contains the messages sent by the users. A notification appears when a new message appears. The system setting allows managing both the messages accessibility and notifications display. Both the received message and received notification are managed with the help of some settings: user rights, notification receipt settings, and new messages notifications display setting. Let's consider the system behavior broken down by these settings.

As we know, there are two types of conversations: context and non-context. The user rights for these conversations are formed differently:

- For a context conversation: the user must have `Read` right to the discussed object.
- For a non-context conversation: the user must be the conversation participant.

The user will get the notifications of new messages in the case of the following conversation settings:

- For a context conversation: the user has specified Notify of all messages in the conversation settings or is the message recipient.
- For a non-context conversation: the user has specified Notify of all new messages in the conversation settings.

Thus, if a user has the rights to the conversation and he has resolved to receive the notifications - the notifications of new messages will be displayed in the Notification center.

The notification receipt mode is managed with the help of the methods `GetWatchMode()` and `SetWatchMode()`. The parameter describing the watch mode actually means the following (depending on the conversation type):

- Context conversation:
  - `True` -corresponds to the mode Notify of all new messages.
  - `False` -corresponds to the mode Notify only of new messages addressed to me.
- Noncontext conversation:
  - `True` -corresponds to the mode Notify of new messages.
  - `False` -corresponds to the mode Do not notify of new messages.

As well as the notifications receipt mode setting, the watch mode can be set for both the current and arbitrary user. In the second case, the privileged mode setting is necessary for the operation execution.

The last setting controls the display of pop-up notifications of new messages:

- If Do not disturb mode is selected in the user settings - the notifications will not be displayed for all conversations.
- If Do not disturb mode is selected in a conversation settings -the notifications will not be displayed for the conversation in which Do not disturb mode is specified.

Do not disturb mode can be managed in the conversations with the help of embedded language. `GetConversationNotificationRepresentation()` and `SetConversationNotificationRepresentation()` methods of the collaboration system manager are used for it. The notification representation mode can be set for both the current user and any user of the collaboration system. -The privileged mode use is necessary to work with the notification representation mode for any user.

30.4. Non-interactive collaboration

30.4.1. General information

Most of the operations performed using the collaboration system can be performed interactively or using the methods of the 1C:Enterprise language. In this section, examples of implementation of various capabilities of operation with collaboration system will be considered.

IMPORTANT! Examples given in this section are not complete, but serve only to illustrate the operation of the described mechanisms.

To access the program interface of the collaboration system, the collaboration system manager is intended, which is represented by the global context property `CollaborationSystem`. To call any method, you should specify `CollaborationSystem. MethodName()`. However, to simplify the text, during the description of a particular method, the `CollaborationSystem` prefix will go down. Examples, of course, will contain the correct call syntax.

30.4.2. Register/register application cancel

To perform the registration of the application on the server of the collaboration system, a couple of methods is intended: `BeginInfobaseRegistration()` and `ExecuteInfobaseRegistration()`. The first method operates in the client application and is asynchronous. The second method is executed on the server and is synchronous.

Both methods acquire the registration parameters for the input (`CollaborationSystemInfoBaseRegistrationParameters` object) and return the registration result (`CollaborationSystemInfoBaseRegistrationResult` object). The client variant of the method also requires a link to the notification handler (`NotificationDescription` object) to which the registration result will be transferred. The method work result will be the registration result in the sever variant of the method.

In the object `CollaborationSystemInfoBaseRegistrationResult`, the registration status is displayed by the property `RegistrationExecuted` and if the registration is not completed that the property `MessageText` contains the information for the use.

Actually registration is performed in several steps:

1. In the first step, the parameters of the registered application are passed to the server of the collaboration system. To do this, fill in the properties of the `Email` and `ServerAddress` object `CollaborationSystemInfoBaseRegistrationParameters` and call the `BeginInfoBaseRegistration() method` /`ExecuteInfoBaseRegistration()`.
2. In the next step, the collaboration system server sends an activation code to the email address that was specified in the previous step.
3. At the last step, repeat call of the `BeginInfoBaseRegistration()`

method/ExecuteInfoBaseRegistration(), but in addition to the Email Address and ServerAddress properties, you should also fill in the ActivationCode property. In this property, you should put the value that came in the email from the collaboration system server.
The server address of the collaboration system can be the server address 1cDialog.com (wss://1cdialog.com:443) or the address of the self-deployed server of the collaboration system. Documentation for deploying an interoperability server is located at address https://its.1c.ru/db/cs20.

4. If no errors occurred, then by this time the registration was completed successfully. In case of any errors during the registration method call, then the information for the user will be transferred in MessageText property of CollaborationSystemInfoBaseRegistrationResult object. The means to obtain the object with the registration result depend on the registration method used. After the end of registration, all cryptographic keys that are necessary for managing the collaboration system are generated and saved.

The property CollaborationSystemInfoBaseRegistrationParameters.InfoBaseName is used to indicate the representation of the registered application.
To cancel the registration of the application in the service, you should use the BeginInfoBaseUnregistration() or BeginInfoBaseUnregistration() methods.
After successful completion of the method, users of the current application will not be able to use the collaboration system. In particular, all necessary cryptographic keys will be deleted. However, if you re-register the application, then the collaboration system will be restored and access to all the correspondence that was generated before the disconnection will be restored.
To completely disable an application from the collaboration service (including the loss of access to all correspondence generated when operating in the application), you should use the SetInfoBaseRegistrationData() method, where the value of the registration data is passed to Undefined value. After performing such an action, the registration of the application is canceled, and access to messages is irretrievably lost.
The method SetInfoBaseRegistrationData() has a paired method (GetInfoBaseRegistrationData()), which allows to get all the information about the collaboration system connected to the server as a single

object. This object contains all the information that was specified during registration, as well as all cryptographic keys that are used to operate with the collaboration system server. If you transfer this data as a parameter of the `SetInfoBaseRegistrationData()` method, then the collaboration system consider the application in which this action is performed to be completely equivalent to the application in which the data used for the installation were obtained. Therefore, the main purpose of the pair of the methods`Get/SetInfoBaseRegistrationData()`- is the migration of a data area between different databases.
The last thing worth mentioning in the context of registering an application in the collaboration service is testing the interaction between the application and the collaboration server. There are two methods for this purpose:

- The `InfoBaseRegistered()` method verifies that the application is registered in the service and this registration is not disabled.
- The `CanUse()` method checks if the application is registered in the service, the registration is not disabled and there are users in the application.

30.4.3. User management

30.4.3.1. Collaboration system users creation and update

After the application is registered in the collaboration service, you can synchronize the application users list and the collaboration system users list. To perform this operation, the platform provides a variety of methods.
The current user is already created in the collaboration system - this happens automatically when you first log in to the 1C:Enterprise system after registering the application in the service. But it may be necessary to transfer all existing users of the application to the service immediately after registration. To do this, use the `CreateUser()` method. The object `InfoBaseUser` is the method parameter. To get a list of the info base users, you can use the `InfoBaseUsers.GetUsers()` method.

Example:

```
SetPrivilegedMode(True);
IBUsers = InfoBaseUsers.GetUsers();
For Each IBUser In IBUsers Do
  Try
    CSUserID = CollaborationSystem.GetUserID(IBUser.UUID);
  Exception
    CSUser = CollaborationSystem.CreateUser(IBUser);
    // it is possible to fill user collaboration system properties
    CSUser.Write();
  EndTry;
EndDo;
SetPrivilegedMode(False);
```

In this example, an attempt to obtain the user ID of the collaboration system for each application user is made. If the user of the application has not yet been created in the collaboration system-, an exception is generated. In this case, based on the application user, the user of the collaboration system is created.

When managing users of the collaboration system, you should pay attention to the `MatchingKey` property. This property can be used to matching users when combining applications. When managing users of the collaboration system, you should pay attention to one more useful property: `InfoBaseUserID`. This property contains the unique infobase user ID, which can be used to search for the user `InfoBaseUser.FindByUUID()`. If `InfoBaseYderID` property is equal to `Undefined` value it means that the infobase user corresponding to the collaboration system user was deleted from the infobase. It is impossible to repeatedly (after the collaboration system user is created) associate the infobase user and the collaboration system user.

`InfobaseUser.Blocked` property can be used when it is necessary to disallow an infobase user using the collaboration system. Such need may arise, for example, when a user has left the company in which the collaboration system was used. If the collaboration system user is blocked then such user:

- Can't connect to the collaboration system server.
- Ceases to get the notifications.
- Is not displayed in the collaboration system user selection.
- A video call to such user is impossible.
- Is displayed with a certain mark.
- An exception arises when a message is sent to such user as well as when he is added to a conversation or video conference participants list.

30.4.3.2. Collaboration system users selection

30.4.3.2.1. General information

When a user is specified in the collaboration system the platform allows managing this process. `CollaborationSystemUserAutoComplete` event handler is intended for it. This event handler may be located in the client application form module or in the application module. An own collaboration system user selection form can be implemented. `CollaborationSystemUserChoiceFormGetProcessing` event handler is intended for it. This handler may also be located in the client application form module or the application module. `DefaultCollaborationSystemUserChoiceForm` configuration property can also be used to specify the collaboration system users selection form.

The system allows processing the user selection in the following cases:

- selection of the recipient of messages in a context conversation;
- selection of non-context conversation participant;
- selection of video conference participant.

The purpose of users selection is described by the system enumeration `CollaborationSystemUserChoicePurpose`. This enumeration value is shown in the user autoselect and form choice event handler in `ChoicePurpose` parameter. The developer can understand where an action is performed with its help.

The general user selection procedure is as follows:

- When a new message is created the user starts typing his correspondent name. The user autoselect is activated in the collaboration system. The users chosen are displayed in form of a select from list next to the user name entry field.
- After the select from list is formed, the platform determines whether a separate form for the collaboration system user selection exists in the given application solution. If such form exists it is displayed in form of hyperlink Show all in the bottom part of the drop-down user list. The selection form is determined with the help of a separate event as well as a configuration property.
- One user can be selected from the select from list but -at once several users can be selected from the select form. The selected users get to the relative entry field and the selection is completed. Now, it is possible to begin selecting a new user.

30.4.3.2.2. User list determination

Let's consider the sequence of events arising during the users input in more details:

1. `CollaborationSystemUserAutoComplete` event handler of the client application form is called. This event handler can be called only if an addressee is chosen in a context conversation.
   The handler gets the list of users formed by the system for the input. This list can be modified or completely replaced.
   If `Standard processing` parameter is set in `False` value when the handler is completed then the platform will use the list of users obtained from this handler. If `Standard processing` parameter is set in `True` value -the autoselect handler will be called from the application module (the next step).
2. `CollaborationSystemUserAutoComplete` event handler of the client application module is called.
   The handler obtains the list of users formed by the system or `CollaborationSystemUserAutoComplete` event handler of the client application for the input. This list can be modified or completely replaced.
   The system will use the exact list of users obtained from this handler for the drop-down list of users formation.

The following parameters are transferred to the autoselect handler:

- The text input by the user (`Text` parameter).
- The list of the collaboration system users selected by the platform based on the input text (`ChoiceData` parameter). The developer can redefine this list.
- What the users are chosen for (`ChoicePurpose` parameter).
- The description of the form of the client application in which the context conversation for which the recipients of the message are chosen is located (`Form` parameter).
- ID of the conversation to which the users are added (`ConversationID` parameter).

The data contained in `ChoiceData` event handler parameter, after the handler closure, will be used in the further chain of handlers or directly in the collaboration system.

The list of values acts as `ChoiceData` parameter value. The following may act as this item values:

- `CollaborationSystemUserID` type value. If the value of this type is "empty" then such value will be interpreted as No notification

in the formation of the list of recipients of the context conversation message. Such value will be ignored in all other cases. An "empty" value of the type `CollaborationSystemUserID` can be formed with the help of `CollaborationSystemUserID` builder object without any parameters.

- `UUID` type value. The platform will treat it as infobase user ID. The collaboration system users will be searched by this ID.
- Values array of `CollaborationSystemUserID` or `UUID` type. A "group" of users for which own representation can be set will be formed in the select from list in this case. The user grouping by function can be used as an example: all users performing the seller function will be enumerated in the group Sellers. In this case, a group selection will result in the addition of a large number of users to the list without the necessity to choose the users one by one.

If a picture is set for the list item -it will be displayed in the select from list. If no picture is set then a picture of the group of users will be displayed in the case of an array and a picture of the set user will be displayed for certain users.

List of values item representation is either set in the relative item property or formed by the system automatically:

- User full name will be displayed for a certain value.
- -The full names of the users included in the group divided by comma will be displayed for a group.

30.4.3.2.3. User select form determination

After the drop-down user list is formed, the system determines whether a form for the collaboration system user selection exists in the application solution. It is determined after each list of users formation. The following sequence of actions is used for it:

1. `CollaborationSystemUserChoiceFormGetProcessing` event handler of the client application form is called. This event handler can be called only if an addressee is chosen in a context conversation. `SelectedForm` handler parameter serves for the select form name specification. The subsequent behavior, as usual, depends on `StandardProcessing` handler parameter value:
   - the parameter is set to `True`: application module form selection handler is called.
   - the parameter is set to `False`: the platform uses the form specified in `SelectedForm` parameter as the collaboration system user select from form.

2. CollaborationSystemUserChoiceFormGetProcessing `handler of the` application module is called.
   The handler behavior is similar to the behavior of the analogous handler of the form module. But there is the following distinctive feature: the specification in this handler of `True` value for `StandardProcesing` parameter will lead to the system attempt to use the form specified in `DefaultCollaborationSystemUserChoiceForm` configuration property as the user selection form. If this configuration property is not set -the collaboration system users selection form will not be used. It means that the drop-down list of users will not contain the hyperlink Show all.

The following parameters are transferred to the user selection form selection event handler:

- What the users are chosen for (`ChoicePurpose` parameter).
- Each form contains a context conversation if the message recipients selection is performed (`Form` parameter).
- ID of the conversation to which the users are added (`ConversationID` parameter).
- The parameters to be used to open collaboration system users selection form (`Parameters` parameter). The platform forms the following parameters in this structure:
  - `ChoiceMode` parameter is set to `True` value.
  - `MultiSelect` parameter is set to `True` value.
  - `CloseOnChoice` parameter is set to `True` value.
- The form to be used to select the collaboration system users (`SelectedForm` parameter). If this parameter is not equal to an empty string when the handler is closed -the hyperlink Show all is added in the collaboration system user selection list and the selected form is opened by clicking on this link. The form opening parameters are set by `Parameters` parameter values and can be modernized in the select form determination handler. Owner of the opened form is the form in which the conversation is displayed.

The select form used to select the collaboration system users shall return the chosen users as `Close()` method parameter. The following may act as `Close()` method parameter:

- CollaborationSystemUserID `type value`.
- `UUID` type value. The platform will treat it as infobase user ID. And the collaboration system users will be searched by this ID.

- An array of values of the types enumerated above.

30.4.4. Using conversations

30.4.4.1. Basic features

To use the conversation you need to create this conversation first. To do this, use the `CreateConversation()` method. This method creates an object which allows to customize the conversation and use it. Created conversation is not yet written. To write it you must explicitly call the `Write()` method of the created object. Let us consider some properties of the conversation being created.

The `ID` property is required to uniquely identify a conversation. In particular, to create a message (using the method `CreateMessage()`) you need to know to which conversation this message will be sent. Conversation identification is performed using the `ID` property.

The `Key` property allows you to identify a conversation by any unique ID, for example, provided in advance. So, if the conversation is used for communication between the server and the client, then to identify the required conversation, you can create a conversation key at the stage of the application solution development. Later this key can be used everywhere to get the conversation. The conversation key cannot be changed for a conversation already written.

The essential difference between the `ID` and `Key` properties is that the `ID` becomes known at the time of the conversation is written, but the `Key` can be determined in advance, before the conversation is written.

The `Displayed` property allows to manage the availability of a conversation for interactive work with it. If this property is set to `True`, then this conversation is available in the main conversation form and users can work with this conversation. Accordingly, if the property is set to `False`, then such a conversation is not available for interactive work and such conversation can be accessed only in the program interface. The context conversation should always be created with the `Displayed` property set to `True`.

The `ConversationContext` property controls the conversation type to be created: context or non-context. This property is writable only for new conversation. After the conversation is written it is impossible to change its type. You also need to remember that an object can have only one context conversation.

When a non-context conversation is created, the `Group` property allows you to specify how many participants the conversation can have. If the `Group` property is set to `True`, then the conversation can have an arbitrary number

of participants. If this property is set to `False`, then there are always two participants only: conversation author and one his interlocutor.
Using the `Participants` property you can define the list of users who can access the created non-context (!) conversation. If the list of participants is set to `CollaborationSystem.StandardUsers.AllApplicationUsers`, then all users have access to this conversation. If the list of participants is not set for non-context conversation, the conversation is unavailable to all users of the system. Access to context conversation is managed by the `Reading` right for the context object (the `ConversationContext` property). When creating a context conversation, the value of the `Participants` property should not be set. When creating a one-on-one conversation:

- you can't use the value `CollaborationSystem.StandardUsers.AllApplicationUsers`in the list of participants;
- the size of the list of participants must be equal to 2.

As mentioned above, to create a message - you need to know the ID of the conversation in which this message falls. The `CreateMessage()` method forms an object that must be filled with data and written. For the created message, the `Date` and `Author` properties are automatically filled. To change these properties you need to switch to privileged mode. After the new message is written it is “sent” to the participants. Let us consider the properties of the message in more detail.
The `ID` property is a unique ID for the created message. Allows you to uniquely identify each message in the system.
The `Conversation` property attaches the message to any existing conversation. You can not create a message that is not related to any conversation.
The `Data` property allows you to attach arbitrary data to the message. This data has two limitations: the attached data must support XDTO serialization and the amount of data in the serialized form cannot exceed 100 KB. This property is not available from the user interface. It is reasonable to use this property when the conversation is used as technological conversation, for the collaboration of various subsystems of the application.
Using the `Receivers` property, you can specify which users should receive the message alert for the displayed context message. This property does not determine the availability of the message!
If a message is sent to the displayed conversation, then using the `Actions` property you can set a list of “commands” that should be displayed in the

message form. The selection of such commands leads to navigation to a hyperlink or to the corresponding client handler call. Using this feature, you can, for example, implement a fixed list of robot answers in a special conversation.
You should remember that in all cases where reference to the conversation is required (when creating a message, when a handler is attached, etc.), this conversation must exist at the time of reference. It does not matter how the conversation will be identified. It is important that at the time of the execution of the required action, the conversation, which should be referenced, already existed in the database of the collaboration system.

30.4.4.2. Search conversations and messages

During operations, it may be necessary to search (or obtain) any specific conversation. Let, for example, the data about a long process running on a server are transmitted through a conversation with a known key. To get the specific conversation you can use several methods:

1. By conversation ID (the `GetConversation()` method). In this case, a concrete conversation is obtained.
2. By conversation key (another variant of the method `GetConversation()`). In this case, a concrete conversation is also retrieved.
3. With the help of filter (method `GetConversations()`). In this case, a list of conversations that match the filter criteria is retrieved. The result of the method execution may include several conversations or empty list.
4. Obtainment of a context conversation of any object. `BeginGettingConversation()` method of the client application form is used for it.

The system also provides the ability to get a list of messages according to some criteria. To do this, use the `GetMessages()` method. The result of the method execution is an array of messages that satisfy the specified (in the filter parameter) criteria.

30.4.4.3. Extended capabilities

30.4.4.3.1. General information

The application may include automatic response capabilities in some conversations or may require a response to the data received from the 1C:Enterprise server. To implement this, it is necessary to respond to the changes that occur

in the conversations. Then the application can quickly process new messages in conversations and adequately respond to them.
This behavior can be divided into two large classes of tasks:

- implementation of automatic response to non-interactive messages;
- implementation of automatic response to interactive messages;

The reaction to messages is implemented by subscribing the client application to new messages in a conversation. The following tasks can fall into this class of tasks: notifying the client application about some service actions performed (for example, the client application restart), displaying the execution status of long-running tasks performed on the server, etc. New messages handler is attached using the `BeginAttachNewMessagesHandler()` method. To perform this operation, you need to know how to identify the conversation to which the client application subscribes. The method of the unambiguous identification of the conversation (and the message to react) strongly depends on the specifics of the implemented algorithm. To choose proper implementation scheme it is recommended to answer some simple questions that allow to understand more clearly how the interaction will be performed. Now we consider these questions in more detail:

1. What do we want to do? The answer to this question allow to determine how the indication of the necessary action is passed. The action can be passed using the following properties of the `CollaborationSystemMessage` object: `Data`, `Text` or using property `CollaborationSystemConversation.Key`.
2. Who perform the required action? The answer to this question determines the addressing of the performed action. Addressing can be implemented in the following ways:
   - properties `Recipients` or `Data` of object `CollaborationSystemMessage`;
   - properties `Participants` or `Key` of the object `CollaborationSystemConversation`.
3. How is the action parametrized? Performed actions can be simple and unambiguous. The simplest example - is the need to restart a client application at the moment is receives a command to do so. At the same time, actions may require to specify some parameters. For example, it may be necessary to restart the client application at some time or on some condition. In this case, the action parameters can be passed through the `Data` and `Text` properties of the `CollaborationSystemMessage` object.

Let us consider in more detail the answers to these questions.
The method of issuing the commands can significantly depend on the task performed: if the message processing is implemented in an interactive or non-

interactive conversation. So, when the reaction to interactive commands is considered, then it is logical to assume that the commands will be entered by the user using text messages (the `CollaborationSystemMessage.Text` property), but for non-interactive interaction the properties `CollaborationSystemMessage.Data` or `CollaborationSystemConversation.Key` are more convenient. At the same time, using the message property `Data` is a reasonably optimal way to transfer data, since it ensures the proper structure of the transmitted information and allows to describe the contract for interaction between the messages sender and the recipient more formally.

It is obvious that programmatic processing of messages implies a subscription to the new messages. Since the system does not allow to subscribe to new messages of all conversations, it is necessary to determine which conversation should be processed. In this sense, the addressing is even more important then the choice of the method to transmit the command. Thus, defining the necessary conversation determines the way to determine the composition of the processed messages: automatically (if addressing through conversation properties is used) or programmatically (if addressing via the properties of each specific message is used).

From the point of view of the data transmitted in the message, the most general way to answer all these questions is to use the `Data` property of each message. From the point of view of purely non-interactive actions, it is quite reasonable to construct the interaction scheme when a service conversation is formed in the system (with already known conversation key), and the message property `Data` specifies all the data necessary to perform the action: the command, the command participants, the command parameters, etc. Obviously, this approach is not suitable for interactive commands. In this case, it is necessary to solve the problem of the transmitted data by parsing the text of the message.

30.4.4.3.2. Implementation examples

General information

Now we consider a few examples of the implementation of processing messages in the 1C:Enterprise language. Please note that these examples just demonstrate certain features of the mechanism, and they are not complete tools for performing a real task.

Example 1

The general scheme of subscription to a conversation looks quite simple. To do this, you should know only the value that uniquely identifies the conversation. The identifier or key of the conversation can play this role. Thus, a subscription to the processing of new messages of an existing conversation is as follows:

```
Procedure SubscribeToServiceConversation(ServiceConversationKey)
Export
  AttachmentProcessing = New NotifyDescription("AttachmentEnd",
MessageClient, , "AttachmentError", MessageClient);
  ServerMessagesProcessing = New
NotifyDescription("ServerMessagesProcessing", MessageClient);
  CollaborationSystem.BeginAttachNewMessagesHandler(AttachmentEnd,
ServiceConversationKey, ServerMessagesProcessing);
EndProcedure

Procedure ServerMessagesProcessing(Message, AdditionalParameters)
Export
  // this handler will process new messages
EndProcedure

Procedure AttachmentEnd(AdditionalParameters) Export
  // this handler will process the handler attachment completing
EndProcedure

Procedure AttachmentError(ErrorDetails, StandardProcessing,
AdditionalParameters) Export
  // this handler will process the handler attachment errors
EndProcedure
```

In this example, there is no difference (in terms of the method syntax) between the discussion identification methods. Although the formal parameter is named `ServiceConversationKey` its actual value may be the conversation key or the conversation ID.

Example 2

This example demonstrates the creation of a conversation intended to perform a single action, as well as the implementation of this action and sending the command to execute the action.
We create a conversation intended only to send a command to restart the client application for specific user(s).
The command is executed if the specific user is specified in the message. The conversation is identified by the key.
This conversation can be created on the server as follows:

```
Conversation = CollaborationSystem.CreateConversation();
Conversation.Display = False;
Conversation.Key = "ClientsRestart";
Conversation.Participants.Add(CollaborationSystem.StandardUsers.All
ApplicationUsers);
Conversation.Write();
```

A message can be sent to such conversation as follows:

```
ConversationID = CollaborationSystem.GetConversation("Clients
Restart");
CommandRecepients = New Array;
CommandRecepients.Add = (<UserID_1>);
CommandRecepients.Add = (<UserID_N>);
Message = CollaborationSystem.CreateMessage(ConversationID);
Message.Data = CommandRecepients;
Message.Write();
```

The message should always contain either text or data, even if these components are not analyzed by the system.

The new message handler can contain the following code in 1C:Enterprise language:

```
CurrentUser = CollaborationSystem.CurrentUserID();
If Message.Data.Contain(CurrentUser) Then
  Exit(False, True);
EndIf;
```

NOTE. This example is not good enough because it restarts the application "without look" at the user action. This variant should not be used in real system.

Example 3

Another implementation of message processing can be the implementation of one conversation for all service traffic within the system. The conversation is identified by its key, and the "details" of the needed action are in the property `Data` of the message:

- the `Command` property contains the command to be executed;
- the `Recipient` - property contains the user who executes the command.

The command is sent as follows:

```
ConversationID =
CollaborationSystem.GetConversation("CommandChannel");
Parameters = New Structure;
Parameters.Insert("Command", "Restart");
Parameters.Insert("Recipient", RecipientID);
Message = CollaborationSystem.CreateMessage(ConversationID);
Message.Data = Parameters;
Message.Write();
```

The message is processed similar to the previous example:

```
CurrentUser = CollaborationSystem.CurrentUserID();
If Message.Data.Recipient = CurrentUser Then
  If Message.Data.Command = "Restart" Then
    Exit(False, True);
  EndIf;
EndIf;
```

30.4.4.3.3. Bot implementation

If you want to implement a robot that will respond to messages interactively generated by users (so called “bot”), then the logic of handling is a bit different. In this case, you should subscribe not to receiving a new message, but to sending the message. `AttachAfterMessageSendHandler()` is a method for this. The attached handler is triggered every time the user sends an interactive message. The handler is triggered on the client computer where the user entered the message. The attached event handler receives the entered message and the conversation in which the message is sent.
Further response again depends on the logic of the application. In the handler you can analyze the text of the message and respond in some way if keywords are found in this text. In the handler you can analyze the participants of the conversation and respond only if the user who personifies the robot is participant, etc.

```
// === Code on the side of the client application
Procedure CreateBot() Export
  Handler = New NotifyDescription("MessageSendingHandling",
MessageClient);
  CollaborationSystem.AttachAfterMessageSendHandler(Handler);
EndProcedure

Procedure MessageSendingHandling(Message, Conversation,
AdditionalParameters) Export
  If StrStartsWith(Message.Text, "?") Then
    // here is the implementation of the bot
    CreateBotMessage(Conversation.ID, "I, Robot!");
  EndIf;
EndProcedure

// === Code on the side of the server application
Procedure CreateBotMessage(ConversationID, MessageText) Export
  Conversation =
CollaborationSystem.GetConversation(ConversationID);
  Conversation = CollaborationSystem.CreateMessage(Conversation);
  Message.Text = MessageText;
  Message.Write();
EndProcedure
```

The above example attaches the message sending handler, which analyzes the message entered by the user. If a message begins with character “?”, it is considered that such a message is addressed to the bot. In this case, the bot responds to the conversation with the message "I, Robot!".

30.4.4.4. Setting actions into message

Creating bots may require to place some "commands" in the message so that the user can quickly perform the necessary action. To implement this, the message object has the `Actions` property. This property is a list of values that contains information about the "commands". As a value of value list element, use a value which is transferred to the handler and allows it to identify an action being performed, while representation of that action is used as its representation.- If a message action handler is not attached to the message (using `AttachMessageActionHandler()` method), no action is performed. If the handler is attached, then any value analyzed in the handler can act as an action. Whenever a handler is attached, system will transfer values defined for actions to the action handler. And each specific action has to be implemented in a handler.

To display a hyper link (which is active), there is no need to use actions. It is sufficient to place a hyper link in the message.

30.4.4.5. Attachments

The collaboration system allows the attachments to the collaboration system messages. This option is accessible only if the collaboration system server is set accordingly. Use `AttachmentsSupported()` method to find out the opportunity to work with attachments.

The work with the attachments is performed with the help of `Attachments` property of `CollaborationSystemMessages` object. The collaboration system server controls three parameters working with the attachments:

- maximum attachment file size;
- size of all attachment files per twenty four hours;
- size of all received attachment files per twenty fours.

These parameters are set by the collaboration system server. An exclusion is formed in case of any noncompliance with these parameters.

Use `Attachment.Add()` method to add a new attachment. An attached object is not expected to be a file on disk only. The following, for example, can be done to send a print form of PDF-document:

- Create a memory stream (`MemoryStream` object).
- Save a spreadsheet to the created stream in PDF format (`Write()` method of `Spreadsheet` object).
- Add attachment from the created stream.

The following recommendations should be observed in the process of an attachment creation:

- If an attachment is expected to be used in the interactive messages it is recommended to specify the name of the file with extension corresponding to the actual type of content as `CollaborationSystemAttachment.Name` property value. At the same time, `CollaborationSystemAttachment.ContentType` property must contain the real MIME type of the attachment.
- If the attachment is not expected to be used for interactive operations then the values of `Name` and `ContentType` properties of `CollaborationSystemAttachment` object can formally be any but it is recommended to specify the real MIME type maximally correctly describing the type of transferred data as `ContentType` property value.

If `CollaborationSystemAttachment` property is not set then the platform tries to determine the attachment type on its own:

- By the interactive attached files in the message:
  - The type is determined by the file extension in the thin and thick clients.

    - The type is determined -on the basis of the information provided by web-browser in the web-client.
- The type `image/png` is set when an image is inserted.
- If `CollaborationSystemAttachment` object is formed from the embedded language and its property `ContentType` is not set then it is considered that `Name` object property contains the attached file name and an attempt to determine the type on the basis of this file extension is made.

The platform displays the messages with attachment in the client application interface with the help of certain pictures:

- The default platform picture of the corresponding file type is displayed for the standard files of platform type (text document, spreadsheet, graphic schema, geographical schema, HTML-document, external report, external processing).
- The pictures of the corresponding type are displayed for Microsoft Office, Open Office, Libre Office, PDF-files, and archives.
- "Audio" picture is displayed for an attachment with `audio/*` type of content.
- "Video" picture is displayed for an attachment with `video/*` type of content.
- "Text" picture is displayed for an attachment with `text/*` type of content.
- "Image" picture is displayed for an attachment with `image/*` type of content.
- "Unknown file type" picture is displayed for all other types of content.

30.4.4.6. Actions for attachments

If the work with attachment is set in the collaboration system it may be necessary to take some special actions for the attachments in the application solution. `AttachCommandsGenerationHandler()` method is intended for it.

This handler is activated each time when it is necessary to get a default command (by left-click) or context menu (by right-click) in the client application interface. The commands generation handler will be called by right-clicking if the cursor is above the link, attachment, action or message itself. The list of commands will be requested by left-clicking only for a link, attachment or action. As the commands generation handler is called interactively and rather often it is not recommended to place any long and interactive operations (for example, ask questions or open forms) in it.

The commands generation handler gets the following parameters for the input:

- `Parameters` - contains the information on where in the message the click is made. `Parameters.Source` property allows understanding where the click is made. `Conversation` and `Message` properties set the action "context".
- `Command` parameter contains the objects array of `CollaborationSystemCommandDescription` type. When the handler is called this parameter contains the commands which the platform places for this data source in the context menu by default. The developer can form this menu completely on his own by deleting the default commands. It shall be pointed out that the embedded language structure `Commands.Add(New CollaborationSystemCommandDescription(Undefined));` will add a separator to the context menu. The standard collaboration system commands can be added. In this case, the necessary adding of command looks as follows `Commands.Add(New CollaborationSystemCommandDescription(StandardCollaborationSystemCommand.CopyMessage));`.
- `DefaultCommand` parameter contains the description of the command called by left-clicking in the message. If this parameter is equal to `Undefined` value then the context menu will be opened in any case: by right- and left-clicking.
- `AdditionalParameters` parameter contains the data which may be necessary to be transferred to the commands generation handler from the context in which this handler is connected.

If necessary, the commands generation handler can be detached with the help of `DetachCommandsGenerationHandler()` method.

30.4.5. Shared use

The collaboration system allows to combine several applications registered in the collaboration system into a single space. In the case of application combination the users of all combined applications act as the users of the collaboration system. When combining applications, the exchange of the messages between applications becomes possible. Only the owner of the subscriber of all combined applications can combine applications. The shared use is supported:

- by service 1cDialog.com,
- by product 1C:Collaboration server.

To combine applications, you can use the standard function of the 1C:Enterprise system (when using the 1cDialog.com service) or implement the necessary code in 1C:Enterprise language (when using the 1C:Collaboration server instance).

The geographically distributed organization, i.e. a chain of stores and the central office of the chain, is an example of situation when such application combining is necessary. Each store and central office has its own infobase. At the central office you can deploy an instance of the 1C:Collaboration server or use the 1cDialog.com service. Databases are registered in the selected service (each store and the central office). Then the resulting applications are combined. The result is a single information space where users of all databases can communicate in real time.

To combine applications, you should use the `CollaborationSystemApplicationLinks` object. The list of applications to be combined can be set using the `Applications` property of this object. This property can be filled using the `GetSubscriberApplications()` function call. This method should be called when operating on behalf of the user who owns the subscriber.

In addition to the list of combined applications, the `CollaborationSystemApplicationLinks` object allows to specify how to match users of the collaboration system where the applications are combined:

- By name (value `Name`) - in this case the users of the collaboration system with the same `Name` property are considered as the same users.
- By full name (value `FullName`) - in this case the users of the collaboration system with the same `FullName` property are considered as the same users.
- By matching key (value `MatchingKey`) - in this case the users of the collaboration system with the same `MatchingKey` property are considered as the same users. In this way you can perform a selective matching of the application users. The users matched between different applications should be specified using the `MatchingKey` property. In this case, users with the same keys are transparent for the combined applications, and other users are available only within the scope of "their" collaboration system applications.

The user matching method is selected using the `UserMatching` property.

Finally, the `ConversationContextMatching` property of the `CollaborationSystemApplicationLinks` object allows to specify objects having the same identifier - in different applications to be considered the same within the information space of the combined applications. So, if

within the distributed information system the conversation contexts are matched, then one document can be accessed from different applications (infobases).
After combining several applications they begin to function according to the “all with all” scheme. It means that each of the combined applications can communicate with any other combined application. At the same time, the collaboration system does not support the following scheme: `Application1` is combined with `Application2`, and `Application2` is combined with `Application3`, but `Application3` can't communicate with `Application1`.
To cancel an application combination, use the `CancelSubscriberApplicationLinks()` method. Cancellation of sharing is performed in the same way as setting a matching: a list of applications that you don't want to use together (but not less than two) is formed, and the `CancelSubscriberApplicationLinks()` method is called.
Using the `CollaborationApplicationSharing()` method, you can get a list of shared applications and sharing options. You should take into account that shared applications are returned in pairs.

30.5. Administration

The administration guide for the collaboration system is given in te apropriate book.

Chapter 31. Development tools

31.1. Form editor

31.1.1. The editor decription

The form editor allows to perform all the actions to create and modify a form. A form editor is a group of several related editors:

- attributes - tab Attributes;
- command - tab Commands;
- elements - tab Elements;
- parameters - tab Parameters;
- module - tab Module;
- command interface - tab Command interface.

The lower part of the window displays the form in preview mode.

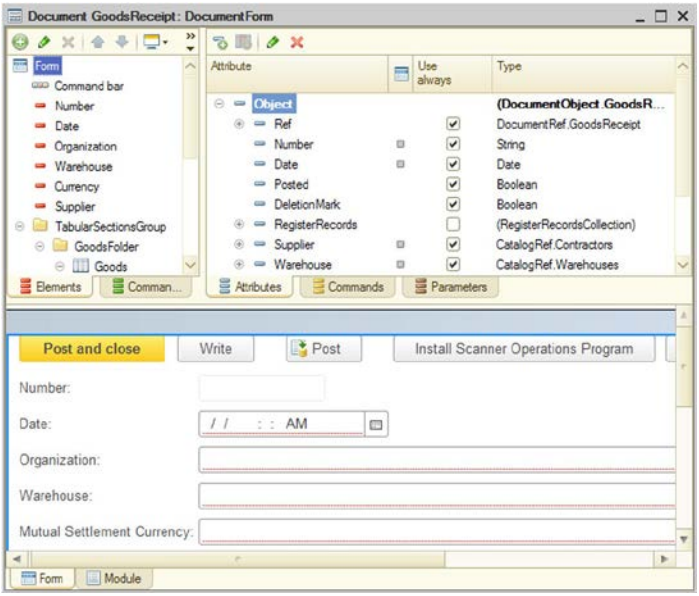


Fig. 300. Form editor

Multiple selection is available in the editors. In this case, only those properties that are present in all selected objects are shown in the property palette. Changes made in the property palette are applied to all selected objects.
The form elements are usually added by dragging the form attribute onto the elements panel. The name of the added form element is equal to the attribute name, and the path to the requisite is set as the data of the form element. The name of the element can be changed later. The system automatically determines both the form element to display the attribute, and the type of this element, if possible. The commands (both form commands and global commands) are added to the form in the same way. If the form attribute or the command is placed on the form, it has special grey marker to the right of its name.
If a form element is added by dragging from another form, then its Data property is not transferred. For all elements transferred in this way, it is necessary to restore the correct paths to the data.
When a form element is selected on the Elements tab, the selected element is automatically highlighted in the preview window. If an element is selected in the preview window, this element is automatically selected in the form element tree on the Elements tab.

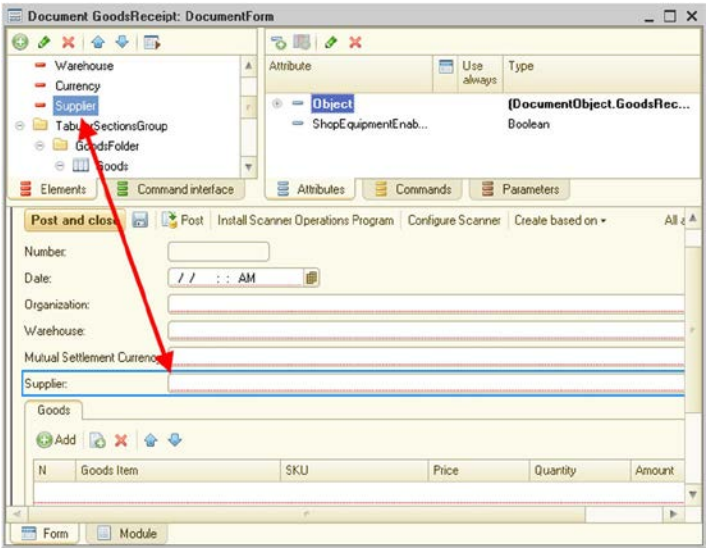


Fig. 301. The editors connection

You can select form elements in the element editor or in the preview window. The features available in the preview window are almost the same as in the list

of elements. In particular, you can use context menus of elements, open the properties window, drag elements, etc. If you select an element in the preview window, a border appears around the element. The border color depends on the element.

Border color	Description
Blue	Usual element selected
Orange	Automatically created buttons and submenus of a command bar selected. Indicates that no settings can be made for this element (properties, position)
Gray	Highlights the parent element if the selected element cannot be highlighted in the preview window. For example, if you select the context menu of the input field in the list of elements, then the input field is highlighted in gray in the preview window
Green	When dragging elements displays the element to which you drag (for example, a table or group of the form), and shows where the dragged element will be placed

Clicking the window title in the preview window is equivalent to selecting the Form element (on the Elements tab). The context menu that opens in this case is also equivalent to the context menu of the Form element.

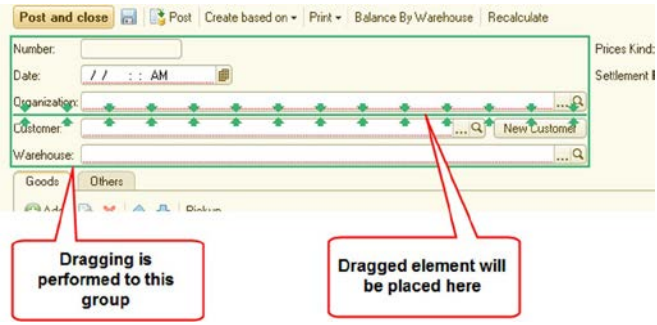


Fig. 302. Dragging indication

To select a parent element (for example, a group that includes this field) for an element, click two times the subordinate element by left mouse button. This selection works if the interval between clicks is from 1 to 2 seconds. With less time, the system can count the clicks as a double click, with more time - they are treated as separate actions. If at the moment of the first click several elements are selected, the parent element is not selected.

To quickly navigate to related objects, use the Go To command. This command is located in the context menu of objects; in element, command interface fragments, attributes, commands editors; and in preview window. Using this command, you can, for example, quickly navigate from a form element (highlighted in the list of elements or in the preview window) to the form attribute displayed by this element. If it is possible to navigate to several objects, a window with a suggestion to select the desired object is opened.
Also the form editor allows to quickly create (or navigate to) event handlers of a particular form element using the Events submenu of the element context menu.

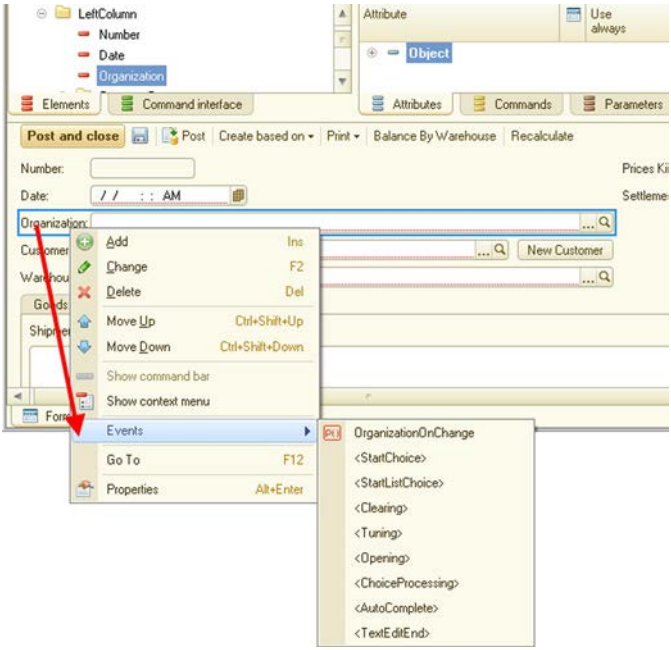


Fig. 303. Menu Events

This menu includes all the events available in the properties window of given form element. If the menu contains the event name in angle brackets, it means that the event handler is not defined (for example, for the event `Clearing` of the `Organization` element on fig. 303). To create a handler, select the desired event. - The editor of the form module opens and creates the necessary handler.
If an event handler is already created for the event, then selection the corresponding item in the Events submenu navigates to the existing event handler in the

form editor (for example, the `OrganizationWhenChange()` handler on fig. 303).
For a button the Event submenu is not formed, and the command to navigate to the handler of the corresponding command is located directly in the element context menu.
If you create a client handler, the system offers several options to do it:

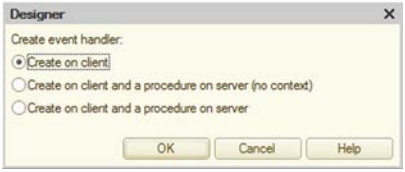


Fig. 304. Creating a client handler

The first option (Create on client) involves creating only a client event handler. The option Create on client and procedure on the server without context implies the creation of a client event handler, a non-contextual server procedure and a procedure call from the client handler. The last option differs from the previous one in that the server procedure is created as contextual. If in some client event handler it is not recommended to use contextual server calls (information about it is available in the Syntax Assistant), then the use of the latter option (with the contextual server procedure) is not possible. Deleting an event handler does not delete the server procedure (if it was created when creating the event handler).
When deleting an attribute (from the list of the form attributes) the system displays the question Delete the related elements? If the answer is negative, then the `Data` property is cleared for the related form elements. If the answer is positive, the form elements associated with the deleted attribute are deleted as well. If the link of the element with this attribute is not the main link (footer data, data displayed in the group header, etc.), then this element is not deleted, but the link with the deleted attribute is cleared. Regardless the answer to the question about deletion, the commands of the command interface of the form associated with the deleted attribute are deleted.
If you delete the form commands, you are also prompted to delete related items. If the answer is negative, the form elements are not deleted, but the element property Command is cleared. If the answer is positive, the buttons associated with the deleted command are deleted as well.

When designing forms, you need to see how the form is displayed in the desired interfaces (Taxi, version 8.2, mobile device). This can be done directly in the form editor.

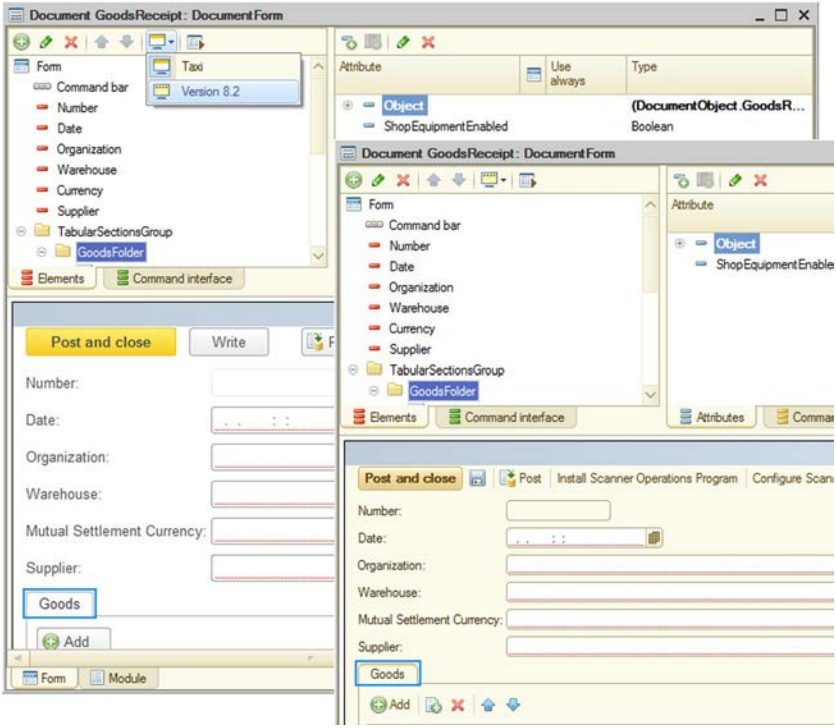


Fig. 305. Interface mode switching

Figure fig. 305 shows an example of the display change when the interface mode changes from the Taxi interface (upper left part of the figure) to the version 8.2 interface (middle part of the figure) and to the mobile device interface (right lower part of the figure). However, you should remembered that when the form is tested in the Taxi interface, some elements are displayed in the v. 8.2 interface. The composition of menu items that is displayed when pressing the Interface Option button depends on the value of the Interface compatibility mode property. If this property does not allow to use the Taxi interface, then the Personal Computer (Taxi) item is absent in the menu. Otherwise, the menu displays all three interface options, as shown in fig. 305.

In special cases, it is required to develop forms for which the display scale can be chosen. If the form itself has the Scaling Mode property set to Auto, then the form editor allows to choose the scale to display the form preview.

If the form should be used only in the Taxi interface and the compact display mode is not used, you can check in the form editor the form look if the custom scale is set.

Changing the scale results in a corresponding change in the form in the preview window. In order to change the actual scale of the form, - you should use the form property Scale.
Since the forms of the 1C:Enterprise can be displayed on devices with different resolutions, you should during the development process check the developed form look at needed resolution. To do this, the editor has the ability to quickly select the resolution for which the system places items in the form preview panel.

After a resolution is selected, the form is rebuilt as if the selected resolution is set in the preview panel. The physical dimensions of the form are not change. The current monitor (where the form editor is displayed) is displayed on the first line of the menu and starts with the line Current monitor.
Applications can run on devices which screens can change their orientation (portrait or landscape orientation) during operation. To check the form look when changing the screen orientation the form editor allows to quickly emulate this operation.

When developing an application for a mobile device, it is necessary to preview the developed form look on a mobile device. To do this, use the Interface Option button to select the display mode Mobile Device (Taxi).

For the mobile device mode it is also possible to choose the orientation: portrait or landscape.

In this sense the behavior of the editor is the same as for any other interface option.
The form editor also allows to select the display mode of the mobile device in the preview window. To do this, use the button Fit to window.

You can choose one of the several previewing options:

1. Fit window - in this case, system scales the display to fit the mobile device screen into the preview window without the need to scroll it.

2. Actual size - in this case, the size of the device in the preview window corresponds to the geometric dimensions of the target mobile device. To achieve the closest match, the display scale in the operating system should match the real DPI value of the monitor used.
3. Pixel precision - each pixel of a preview window on the developer's PC is mapped to a pixel on the mobile device display.
4. Select any of the available form scale options.

If during the form development you find various errors in setting the properties of elements that can lead to problems when placing form elements, it is possible to display these errors using a special button.

To open the form for verification press the keyboard shortcut Ctrl + R. The form opens in a separate window. If Mobile Device mode is selected in the form editor, the form validation is unavailable.
When renaming or deleting an attribute, parameter or form element, the system looks for usage of the renamed (deleted) object in the form module and in the entire configuration (if it is allowed in the Designer settings).

31.1.2. Role-based form customization

In the form editor, you can perform role-based customization of form behavior. This feature is provided for the following properties:

- View - a property of the form attribute. The absence of the view feature excludes the attribute from the form data (the attribute is not passed from the server). The user cannot change this setting.
- Edit - a property of the form attribute. With the absence of edit feature the form element associated with the attribute is available only for viewing. Regardless of the state of this property, edit is unavailable if view of the form attribute is not available. The user cannot change this setting.
- User visibility - a property of the form element. Specifies the default visibility of a form element. The user can individually change this property in the form settings editor.
- Usage - a property of the form command. If a command usage is disabled, then all associated buttons are not present in the command interface. The user cannot change this setting.

Editors of all the above properties look and operate the same. We consider the operation of the editor on the example of the property View.

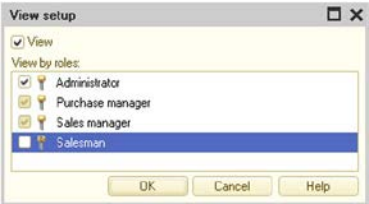


Fig. 306. Editing the "View" property of the form attribute

The View property (in the upper part of the editor window) defines the view state for each role with its check box in the "third" state (the checkbox background is grey). Then, view states for all available roles are added "using OR operation", and the result of this addition determines the final View property for the edited object.

31.2. Text editor

The 1C:Enterprise text editor provides all the basic text editing functions. The text editor provides operations with text blocks, search and replace functions, and highlight of syntax elements of program modules in color.
In the 1C:Enterprise the text editor is used in two modes: to edit text documents and as part of a form editor to edit text modules.
This chapter describes the features of the text editor when editing the texts of the modules.
Since all text editors in the Microsoft Windows operate similarly, this section describes the specific features of the 1C:Enterprise text editor.

NOTE. The text editor doesn't allow to enter characters invalid in the XML specification version 1.0 (http://www.w3.org/TR/xml/). When you try to enter such a character, it is simply ignored; when such characters are inserted from the clipboard, they are skipped and do not appear in the inserted text.

For the list of keyboard shortcuts available in the text editor, see the application help.

31.2.1. Module edit

Modules are edited during creation the configuration object form, as well as in the development of modules (application module, external connection module, common modules, application object modules).
When a form is created, the text editor operates as an integral part of the form editor. To call it click on the Module tab in the form editor window; otherwise, the text editor is called in a separate window. To open a managed application module, an external connection module, a session module, and a ordinary application module, right-click the configuration name (the topmost line of the configuration tree) and select the appropriate item in the context menu.
To edit an object module, select the object and select the Open object module item in the context menu.
To edit the manager module (if the object has such module), select the object and select the Open manager module item in the context menu.
To edit the command module, double-click the command or select the Open command module item in the context menu of the command.
To edit a common module in the Configuration window, in the General - Common modules branch, select the required module and select the Open module item in the context menu.
The process of editing the text of a program module is the same as the process of editing text documents. - You can use all the features of a text editor.
This section describes the specific modes of the text editor available when editing modules.

31.2.1.1. Syntax highlighting

For easy editing the texts of the modules, the text editor has the function to highlight the elements of the 1C:Enterprise language - keywords, constants (not configuration objects) of various types, operators, comments, and so on. The colors for different types of syntax structures can be set in the Parameters window, which is opened by the menu item Tools - Options.
The names of the built-in functions are not highlighted in color (the color matches the color of the identifiers).
In the general case, when a text editor is called to edit the text of a module, this function is automatically enabled. However, in some cases, the module text may be located in an external text file. Then, when opening such a file, Designer does not recognize the module and considers it as a plain text document. In this case, when editing text, highlighting the syntax structures by color and automatic formatting the text of the module are unavailable. You can indicate to the Designer that the edited text is a module, not a text document, using the Text - 1C:Enterprise language menu item, which works as a switch.

When it is turned on (a check mark appears in the menu to the left of 1C:Enterprise language text), the text editor reads the module text loaded into it and highlights the found syntax structures with color.

Fig. 307. Color highlighting is on

If this mode is enabled when editing a plain text document, the test is displayed using font set for the module text in the Designer (the menu item Tools - Options, the Texts tab, the Font attribute).
By setting the parameters of the 1C:Enterprise you can turn off the mode of highlighting syntax structures. In this case the choice of the 1C:Enterprise language item does not enable highlighting of the module syntax, but only the module font and tab positions are set.
If syntax highlighting is turned off, the colors of the operating system are used to display text.

31.2.1.2. Grouping

When modules or text documents are viewed in the 1C:Enterprise language mode, some syntax structures are automatically combined into groups. These structures include `If … Then … EndIf`, `While … Do … EndDo`, `Procedure … EndProcedure` and so on.

The text groups improve code readability and make it much easier to copy or move syntactic structures as separate text units. The display of groupings can be seen in fig. 308.

Fig. 308. Grouping in text editor

You can collapse and expand a group with the mouse. To do this, simply click the left mouse button on the group marker.
If you press the Ctrl key at the same time, you also collapse and expand subordinate groups (conditions, cycles, etc.).
To display collapsed text, move the cursor over the marker, as shown in fig. 308. If the text of the group is large enough, then only its beginning is displayed.
During the development , it may be necessary to group the module fragments. For example, you may need to select logically related procedures and functions,

or select fragments of an algorithm within a single procedure or function. To do this, use preprocessor command `#Region … #EndRegion`.

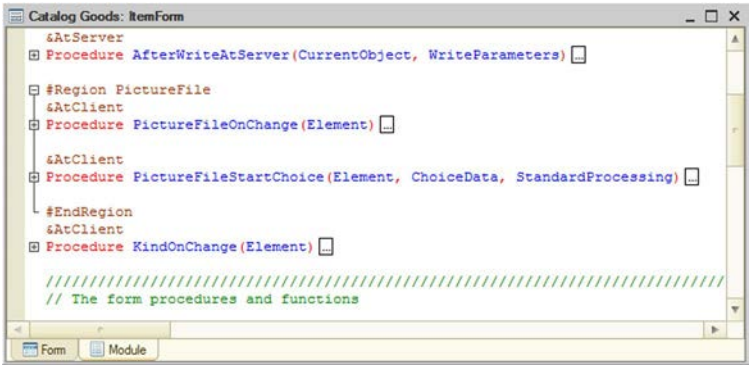


Fig. 309. Line grouping

All lines bounded by preprocessor directives can be minimized in the module editor (like other 1C:Enterprise language constructs). Each region must have its own name formed according to the rules of the 1C:Enterprise language. In the example shown in fig. 309, the name of the area is `ImageFile`. Regions can be nested, but cannot intersect with other syntactic constructs of the language. The following are correct and incorrect examples of the use of regions:

```
// Correct use of region
#Region Correct
…
#If Client Then
…
#EndIf
…
#EndRegion

// Nested regions
#Region CalculationAlgorithm
…
#Region Step1
…
#EndRegion
…
#Region Step2
…
#EndRegion
…
#EndRegion

// Incorrect use of region
#Region Incorrect
…
#If Client Then
…
#EndRegion
…
#EndIf

// Incorrect use of region
#Region Incorrect
Procedure GetData()
…
#EndRegion
…
EndProcedure
```

A region can be described by a comment before the region (`#Region`). This comment can also be grouped.
Using keyboard commands you can easily manage the status of groups.

Keys	Description
Ctrl + Num-	Collapses the group (the cursor can be anywhere in the group)
Ctrl + Num+	Expands the group (the cursor should be in the first line of the group)
Ctrl + Shift + Num-	Collapses all groups
Ctrl + Shift + Num+	Expandes all groups

Keys	Description
Ctrl + Shift + R	Renew grouping

When coding a module at the time the developer completes a syntactic structure, the groups are automatically created for all syntactic structures of any nesting level. Updating groupings is performed using the menu item Text - Grouping - Update grouping or automatically when saving the code according to the group display setting.
The first level is the description of procedures, functions and regions, the second level -is the syntax structures nested only in the body of a procedure or function, but not in the body of other syntactic structures. The mode of the group display can be customized.

31.2.1.3. Using bookmarks

When coding a module, the developer can mark any line of the text in order to have possibility to go to the marked fragment quickly. This mark is called **bookmark**.

To set a bookmark, use the command Text - Set / Clear bookmark. The bookmark is marked with a color mark in the column to the left of the module text.
It is also possible to go to the next bookmark (the Text - Next bookmark command) or previous bookmark (the Text - Previous bookmark command), and also to delete all bookmarks set (Text - Clear all bookmarks).
A list of all the bookmarks set can be seen in a specific list using the Edit - Bookmark list command.

Using this list, you can go directly to the desired bookmark, and delete specific bookmark or all bookmarks at once. Please note that you can go to a bookmark only using a special button. Pressing the OK button does not bring you to the marked bookmark.
It should also be noted that the system saves the bookmarks list between sessions of editing documents and Designer use.

31.2.1.4. Module formatting

The text editor of the 1C:Enterprise system includes a number of modes that facilitate the development of modules.

31.2.1.4.1. Formatting syntax structures

The best practice to write modules is to highlight the control structures of the 1C:Enterprise language using the syntax indentation - with leading spaces (tabs). See example of the module fragment below.

```
Procedure NextElement(Catalog, Selection)

  While True Do
    If Selection.Next() = False Then
      Selection = Catalog.Select();
      Continue;
    Else
      Break;
    EndIf;

    If Selection.IsFolder Then
      Continue;
    EndIf;
    Return;
  EndDo;

EndProcedure
```

In this fragment, the module lines located inside the structure operators `If ... Then ... EndIf` and `While ... Do ... EndDo`, are shifted to the right to emphasize their "nesting". Module text formatted using syntax indentation is easier to read and easier to debug.

The 1C:Enterprise text editor provides the functions for automatic formatting of 1C:Enterprise language control structures. To set up automatic formatting in the system settings mode (the item Tools - Options of the main menu in Designer, the Modules tab), you can select one of two types of indentation.

The syntax indentation performs automatic formatting of the module text, shifting the text located inside control structures of the type `If ... Then ... EndIf` and `While ... Do ... EndDo` and similar to the right. To shift test the necessary number of tabs are added to the beginning of the lines.

“Ordinary” indentation automatically aligns the text of the line to the left border of the previous line.

If auto-indent is disabled, no additional characters are added to the text.

In addition to automatic formatting of the module text, you can also format already entered text during the coding. To do this, select the block of text that you want to format, and select the menu item Text - Block - Format. The test editor analyzes and formats the text by shifting the content of each syntactic construction to the right with the standard tab indent, regardless of the original indents (leading spaces). Tabs are set into blank lines in accordance with the syntax.

A text block can also be shifted as a whole to a tab position to the right or to the left. To do this, select the text block and use the command Text - Block - Shift right (Text - Block - Shift left).

The 1C:Enterprise text editor automatically removes spaces at the ends of lines. This is done when writing a module.

31.2.1.4.2. Insert/delete comment mark

When debugging modules, it is often necessary to “turn off” some module lines temporarily so that they are not executed when the system runs. This is usually done by turning such lines into comments - by adding the comment mark “//” in the beginning of the lines. Later, in order to “turn on” the commented lines into operation again, the comment mark is deleted.
To turn off (and then turn on) large fragments of the module, it is convenient to use the automatic setting of comment marks for all lines of the selected block or for the current line (it is not necessary to select it).
To do this, select the block of text or set the cursor to the desired line and select the menu item Text - Block - Add comment (Text - Block -Delete comment).
When deleting a comment, if there are several comment marks in the beginning of the line, then only one mark is deleted.

31.2.1.4.3. Insert/delete line continuation character

When input the value of string constants, the symbol “|” is used as the line continuation character.
Text editor allows to easily add or remove this character in the selected lines.
To insert/delete a line continuation character for all lines of a selected block or for the current line (it is not necessary to select it), specify the region and select the menu item Text - Block - Insert line continuation character to insert or Text - Block - Delete line continuation character to delete.
The line continuation character is inserted into the first significant (not a space or a tab) position of each line.

31.2.1.5. Navigating to module procedures and functions

With a large number of procedures and functions definitions in a module, it is convenient to use the procedure search mode available in the 1C:Enterprise text editor.

The Text - Procedures and functions command opens a window containing a list of all the procedures and functions of the edited module.

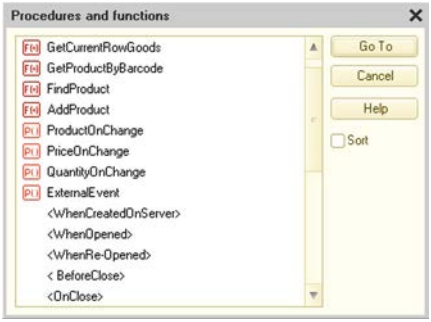


Fig. 310. Procedures and functions list

The names of the procedures and functions are listed in the order in which they appear in the module. If the Sort box is checked, the list is sorted alphabetically. Names of procedures and functions already placed on the form are marked by icon to the left of the name.

To navigate to the desired procedure or function, select its name in the list and click the Go To button.

The list of procedures and functions shows the names of events, which handler procedure is not yet created, in angle brackets. The created procedures and functions are marked by icon before the name (P() and F(x), respectively). The composition of the events that can be processed is determined by the type of object and the type of the form, as well as by the composition of the controls located in the form. When you select such a line, the text of the predefined procedure is added to the module, and a link to this procedure is added to the corresponding event of the Events- category.

IMPORTANT! Creating event handler procedures defined by the system for particular module must be performed in the properties palette for the corresponding form in the Events property category, or using the Procedures and functions window, or from the field in the Procedures and functions list. With simple copying event handler procedures from other modules, the event handlers for this form will not be initiated by the system, and the copied procedures will not be called to handle events.

When you move the cursor to the module line, the name of the current procedure or function is displayed in the procedure selection field of the Module toolbar (the Procedures and functions command). Using this list, you can also navigate to the desired procedure or function.

To go to the procedure, function or variable, just hover the mouse cursor on their name and press the F12 key. Such navigation is possible only for procedures, functions and variables located in the same module, or to exported procedures, functions and variables of the application module, common modules and object modules. In order to return to the point from which the navigation to the definition was made, you must press the key shortcut Ctrl + "-" (next to the "=" key).

If the entered expression consists of parts defined in different places of the configuration, a list of such objects is displayed before the transition to select desired destination. You can go to the definition of a variable, to the definition of a metadata object with type of the current expression, or to the definition of a procedure or function used in the expression (for example, in a object module with type of this expression).
For example, for the expression `Dir.FindByCode()`, where `Dir` is defined as the `Currencies` catalog, pressing the F12 key displays a list of navigation to the definition of the variable `Dir` and to the definition of the metadata object Catalogs - Currencies in the configuration objects tree.
It is possible to search for places where a variable, procedure or function is used. The search is performed across the entire configuration. To perform such a search, place the cursor “inside” of the name that usage you want to find, and call the Find usage command of the text editor context menu (or press Alt + F12).

After the search is completed, a window opens with search results. From this window you can go to the corresponding module by double-clicking the desired line.
The first character in the search results describes the accuracy of the definition of each found usage. The symbol "i" means that the usage is defined completely and unambiguously, and the symbol "!" means that the usage of a variable, procedure or function is not clearly defined.

31.2.1.6. Context tips for module text input

The text editor of the 1C:Enterprise system provides a tool for context input of expressions using system objects, their properties, methods, procedures and functions, names of objects defined in the configuration, as well as variables, procedures and functions defined in common modules, application modules and form modules. The list includes predefined elements of directories, charts of accounts, plans of types of characteristics and plans of types of calculations.
The list of the context hint may include text templates and keywords.

NOTE. Context hint for text is not supported by the system for text documents with the property 1C:Enterprise language set.

The hints are insert from the list displayed as context menu at the current cursor position (taking into account the proximity of the screen borders).
The list is called by pressing the key combination Ctrl + <Space key> at any stage of the expression input or automatically after entering the "." character after the expression, which is an object with properties and/or methods (if the contextual tips are enabled).
The list content depends on the context of the program module execution (see the "General language description" section in the 1C:Enterprise language help) and the text entered previously.
At the initial stage, when the text has not yet been entered or only the initial characters of the expression have been entered, the list content is determined by the execution context.
The list is presented in the form of text lines sorted alphabetically.
If when list is opened a text fragment is entered, then the list is positioned on the first line, the name of which includes the entered or selected text as much as possible (from the beginning of the name). If the typed text is not contained in the lines of the list, then the list is positioned on the line in which the typed text is most represented.
With the list open, you can continue typing. In this case, the list is sequentially positioned on lines that match the typed text.
The list can be viewed in a standard way. When you press the Enter key, the contents of the selected line are transferred to the module, replacing the selected or typed text.

NOTE. When text is transferred to the module, the method parameters are not inserted and the context of the client application is used.

In the left part of the lines there are icons indicating the type of object and the type of its location.

Icon	**Object**
line (black)	• The global context property, • System value sets, • Enumerations
line (green)	• Properties of objects of universal value collections, • Properties of interface objects, • The applied objects attributes, • The predefined elements
line (blue)	Exported variables of modules
line (red)	Local variables of a module
P() (black)	The global context procedures
P() (green)	Procedures of interface and application objects

Icon	Object
P() (blue)	Exported procedures of other modules
P() (red)	Local procedures of a module
F() (black)	The global context functions
F() (green)	Functions of interface and application objects
F() (blue)	Exported functions of other modules
F() (red)	Local functions of a module
Colored lines	Keywords (`If`, `Do`, `Try` etc.)
The templates picture	Text template

Text templates are only included in the list if they have the autoreplacement string specified.

When you enter a text or select it from the list, if the text forms an expression with properties or methods available, once you type "." (dot), the list of available properties and methods for this expression is displayed automatically.

For example, when you enter the text "Catalogs." a list is displayed containing the names of all the catalogs described in this configuration. After selecting a specific catalog and entering the "." Symbol, the list is displayed again, but now it contains the names of procedures and functions for this catalog, as well as predefined catalog elements. When choosing a method that returns a value of a certain type, which in turn has properties and methods, the context input can be continued (you must enter the opening and closing brackets at the end of the name). The context hint list contains only a possible set, determined by the type of the entered expression.

For module variables you can also use the context hint mechanism. When declaring variables, note the following feature: in order for explicitly declared variables are included into the context hint, each variable must be marked by its own `Var` operator. Otherwise, the contextual hints display only the first variable from each `Var` operator. The following example displays all declared variables:

```
Var TemplateName;
Var HideDiscount;
Var MutualSettlementsBalance;
```

In the following example, the context hints display only one variable: `TemplateName`:

```
Var TemplateName, HideDiscount, MutualSettlementsBalance;
```

The context hint can be used when entering the `New` operator, as well as for variables created using the `New` operator.

The context hint can be used when entering various keywords (for example, `If`, `For`, `Do`, etc.). Keywords are included in list called in the standard way by pressing Ctrl + <Space key> at any stage of typing words.
You can customize the display of the keyword list.
If for any variable or method this list consists of only one line, then pressing Ctrl + <Space key> immediately insert this line.
Context hint also allows to display a list of parameters of a method and its type. Parameters are displayed for both global context objects and 1C:Enterprise language objects, as well as for application methods, which description meets certain requirements. For 1C:Enterprise language operators, the context hint for the parameters is not displayed. Parameters are displayed after entering the characters "(" or "," (it can be customized) or after pressing the Ctrl + Shift + <Space key> key shortcut if it is pressed after the "(" symbol of a procedure or function call).

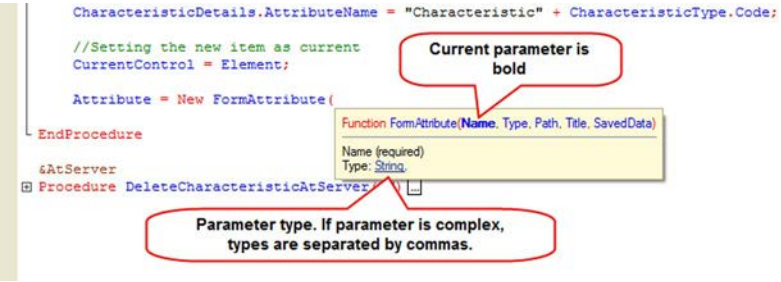


Fig. 311. Context hints for parameters

If a method has several variants of the syntax, then the first variant in order is displayed. However, before the description of the syntax there are the up and down arrows allowing to switch between the various options. You can also switch variants using the Ctrl + <Up arrow> and Ctrl + <Down arrow> keys.

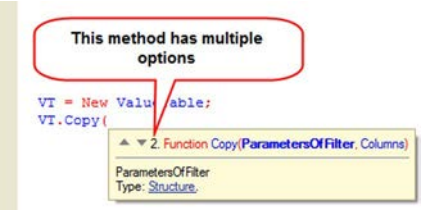


Fig. 312. Several syntax options in the context hints for parameters

Under the parameter description, the type of value that can be passed to this parameter is displayed. If the value can be of several types, these types are listed separated by commas. The type name is a hyperlink, which clicking opens the Syntax Assistant with information about the type. The context hint for parameters is closed after entering the "**)**" symbol or pressing the **Esc** key when the window of the context hint for parameters is opened.

In order to the context hint shows the description of the parameters and their types for the methods of the application, a comment must be placed before the method according to certain rules. In general, the description of a procedure or function consists of the following sections:

- The **Description** section - contains a brief description of the purpose and/or principles of the procedure or function, it can be the only section for functions without parameters;
- The **Parameters** section - describes parameters of a procedure or function. If there are no parameters, - the section is skipped;
- The **Return value** section - describes the type and content of the return value of the function. Procedures have no this section;
- The **Example** section - contains an example of using a procedure or function.

An overall format of a comment is as follows:

```
<Comment> =
  [<HeaderSection>]
  [<ParametersSection>]
```

```
  [<ReturnValueSection>]
  [<ExampleSection>]

<NewLine>  = Beginning of module line
<Name>           = 1C:Enterprise language ID
<Type>           = <Name> | <Name>"."<Name>
<TypesList>   = <Type>{","<Type>}
<TextLine> = Arbitrary text without line break
<Text>        = <TextLine> {<NewLine> <TextLine>}

<HeaderSection> = [<Text>]

<ParametersSection> = <NewString> ("Parameters:")
{<ParameterDescription> {<TypeDescription>} }
<ParameterDescription> = <NewString> <Name> [" " <TypesList>] " "
[<Text>]
<TypeDescription> = <NewString> " " <TypesList>] " " [<Text>]

<ReturnValueSection> = <NewString> ("Returns:") <ValueDescription>
<ValueDescription> = <NewString> [<TypesList> " "] [<Text>]

<ExampleSection> = <NewString> ("Sample:") <NewString> <Text>
```

As a result, a comment before a procedure or function, designed in accordance with the described format, causes the context hint of parameters of the methods to correctly show the hint for the application method.

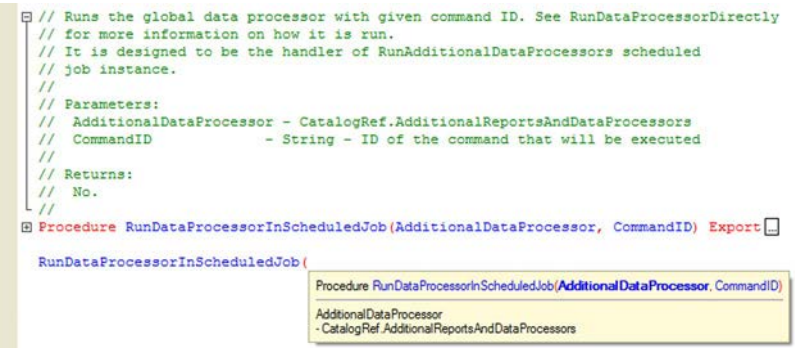


Fig. 313. Context hint of parameters of application methods

Thus, the context hint mechanism of the text editor of the 1C:Enterprise provides a way to quickly and correctly type texts of the modules.

31.2.1.7. Refactoring tools

31.2.1.7.1. General information

Refactoring - is the process to rework the 1C:Enterprise language code improve the code readability and its general internal structure with full and accurate preservation of the original meaning and behavior. Designer offers several built-in tools that can help with the refactoring of the application. These tools can be accessed in the Refactoring submenu of the context menu of the 1C:Enterprise language text editor or in the Text menu of the Designer main menu in the same name submenu.
Now we consider the refactoring tools in more detail. We will use the following terms:

- **Block** - is the line set between the keywords `Then`...`EndIf`, `Then`...`Else`, `Then`...`ElseIf`, `Do`...`EndDo`, `Try`...`Exception`, `Exception`...`EndTry`, `Procedure`...`EndProcedure`, `Function`...`EndFunction`;
- **Terminal call** - is the last statement executed by the method of the 1C:Enterprise language:
  - It is located before the keyword `EndProcedure`, `EndFunction` or `Return`;
  - It is the last statement in the recursive-last block of the method statements.
- **Asynchronous method**- is a method that can be used in call chains using notification handlers.

31.2.1.7.2. Extract fragment

This command converts the selected syntactically complete fragment of code into a separate procedure or function. The type of extracted fragment depends on the final scopes of the variables of the parent method and the structure of the fragment.
If the extracted fragment can act as a source in an assignment operator, the fragment is extracted as a function. In addition a function can be extracted based on a fragment initializing a single variable, which is used in the rest of the parent method. In other cases, the fragment is extracted as a procedure.
If the variables of the extracted fragment are used in the remaining part of the parent method, these variables are passed to the extracted method as parameters, with their preliminary initialization or declaration (if necessary).

If the parent method (**from** which the fragment is extracted) is preceded by some kind of compilation directive, then the extracted method is also preceded by the same compilation directive.
The code of the new method is placed immediately after the code of the parent method (from which the fragment is extracted) and is formatted properly.

31.2.1.7.3. Rename

This command allows to change the name of a variable or method. If the reference to the export renamed variable or method cannot be determined unambiguously, the renaming is not performed and a list of the use of the variable or method is displayed.
For export variables and methods of the form module, the tool analyze if the identifier within the whole configuration is unique and if there are no its synonyms among the methods and properties of the platform. If this conditions are met, the user is offered to replace the identifier globally in all places of its possible use.
Renaming a method parameter does not change the name of this parameter in the comments, including the description of the method (created in any way).
In the case of renaming methods, an additional check is performed to transfer their name as a string literal:

- into object constructors:
  - `Action`;
  - `NotifyDescription`.
- global context methods:
  - `SetAction()`;
  - `AttachDataChangeHandler()`;
  - `DetachIdleHandler()`;
  - `AttachIdleHandler()`;
  - `AttachNotificationHandler()`.

Renaming a handler method in a form module automatically replaces the reference to the renamed handler in the corresponding properties of form elements and commands. When you rename the handler method from the form element or properties command palette,- all calls to the given handler are automatically renamed in the program code (if they occur).

31.2.1.7.4. Create method description

The command creates a comment before the method declaration, which will be used for context hint when entering modules text.

31.2.1.7.5. Create notification handling

This command simplifies the use of asynchronous methods of the 1C:Enterprise language. The command becomes available after the method name and the opening parenthesis are entered (the cursor must be placed on the method name). The further behavior of the command is equivalent to the behavior of the refactoring command Convert call, with the exception of renaming the method to which the refactoring command is applied. If the method already passes object of the type `NotifyDescription`, the user is asked to confirm the command execution to continue.

31.2.1.7.6. Using legacy synchronous calls

General information

This section describes refactoring commands that allow to rework the application code to use asynchronous calls instead of synchronous calls.

Convert call

The command replaces the trivial and simple cases of synchronous methods use with their asynchronous counterparts.
A case is considered **trivial** if it transformation affects only one method and does not require operations other than selecting this fragment. The modal call is considered trivial in the presence of the following conditions:

1. The call is located in a procedure that is not called from other procedures and functions, with the exception of recursive-terminal calls. In the analysis for terminality, the belonging of the calling methods may be uncertain. If this is not observed, then the case may be simple or require conversion to an asynchronous procedure.
2. The call is located outside the cycle blocks, except for the case of the `Abort` or `Return` statements following on the same nesting level.
3. The call is outside the exception catch blocks.
4. The nesting level of a modal call does not exceed 2, with the exception of the recursive-last block statements. The nesting level of a modal call can be reduced using the fragment extraction command. If the extracted fragment contains the `Return` operator, then you should extract such a fragment using the command for extraction fragment into asynchronous procedure.
5. The parameters of the procedure do not change after the call of the modal dialog box -, neither in the procedure itself, nor in one of the nested calls.

A case is considered **simple** it it does not satisfy only the first triviality condition and requires, in addition, only to convert a procedure or function with the synchronous call into an asynchronous form. When such a synchronous call is found, and if such a conversion is possible, the system automatically offers to convert it into an asynchronous procedure.
The result of the convertion is:

- The initial procedure with the replacement of a synchronous call to an asynchronous call;
- Notification handler procedure. If the synchronous call is the last statement of the parent block,- the decision to create an empty procedure is taken by the user.
- The fragment-ending - is common code for the initial procedure and the notification handler procedure, when the call to the synchronous method is not included in a single sequence of code execution.

If the case does not meet one or more requirements, the user can perform a partial conversion: the maximum syntactically correct code fragment is extracted to the notification handler procedure, and an exception is thrown after the asynchronous method call. A comment is added to the module code with a description of the reasons for non-compliance.

Convert to asynchronous procedure

The command is used to convert a method to a form that allows to use this method in asynchronous calls. The conversion affects all already existing call sequences involving the selected method, but the methods that are called from the converted method are not changed. This command should be used when it is necessary to convert modal calls (one or several) within a function or procedure called from other methods. After executing this command you must convert the modal call into its blocking variant using the modal call conversion command.
Conversion of methods that alter their reference parameters is not supported in order to prevent hard to find logical errors. In addition the system does not support conversion of call sequences containing a direct call to the procedures for handling events of forms and metadata objects because the list of their formal parameters is strictly fixed. If the conversion involves multiple modules,- the user is asked for additional confirmation.
Let us consider an example of conversion a call to an asynchronous form.

NOTE. This example is only to demonstrate the conversion process and is not an example of any real activity.

Suppose a form module contains the following call chain:

```
&OnClient
Var ProcedureCalled;
&OnClient
Var FunctionCalled;

&OnClient
Procedure FormCommand (Command)
  ProcedureCalled = False;
  FunctionCalled = False;
  CallProcedure();
EndProcedure

&OnClient
Procedure CallProcedure()
  Result = CallFunction();
  ProcedureCalled = True;
EndProcedure

&OnClient
Function CallFunction()
  ShowDialogBox();
  FunctionCalled = True;
  Return True;
EndFunction

&OnClient
Procedure ShowDialogBox()
  DoMessageBox("Warning!");
```

EndProcedure

You should convert the method `ShowDialogBox()`.

The call chain starts when from the form command handler (procedure `FormCommand()`) the procedure `Procedure1()` is called. The actions performed in the `CallProcedure()` and `CallFunction()` methods in this example simply demonstrate the presence of some program code after calling another procedure.

The call chain looks as follows: `FormCommand()` -> `Call Procedure()` -> `CallFunction()` -> `ShowDialogBox()`.

Since at the very "bottom" of the call stack a modal dialog is called, then the entire chain of method calls needs to be converted to asynchronous form. The `DoMessageBox()` method can be replaced with an asynchronous analog, and for other procedures this analog will be created by the system.

We perform the conversion of the `ShowDialogBox()` method and refuse the suggestion to create empty notification handlers. The result is the following code:

```
&OnClient
Var ProcedureCalled;
&OnClient
Var FunctionCalled;

&OnClient
Procedure FormCommand (Command)
  ProcedureCalled = False;
  FunctionCalled = False;
  CallProcedure(Undefined);
EndProcedure

&OnClient
Procedure CallProcedure(Val Notification)
  Result = Undefined;
  CallFunction(New NotifyDescription("CallProcedureFinal", ThisObject,
New Structure("Notification", Notification)));
EndProcedure

&OnClient
Procedure CallProcedureFinal(Result1, AdditionalParameters) Export
  Notification = AdditionalParameters.Notification;
  Result = Result1;
  ProcedureCalled = True;
  ExecuteNotifyProcessing(Notification);
EndProcedure

&OnClient
Procedure CallFunction(Val Notification)
  ShowDialogBox(New NotifyDescription("CallFunctionFinal", ThisObject,
New Structure("Notification", Notification)));
EndProcedure

&OnClient
Procedure CallFunctionFinal(Result, AdditionalParameters) Export
  Notification = AdditionalParameters.Notification;
  FunctionCalled = True;
  ExecuteNotifyProcessing(Notification, True);
  Return;
EndProcedure

&OnClient
Procedure ShowDialogBox(Val Notification)
  DoMessageBox("Warning!");
  ExecuteNotifyProcessing(Notification);
```

EndProcedures

During the conversion, the call to the modal call `DoMessageBox()` in the `ShowDialogBox()` method was not changed, since modal call replacement is not in the scope of the described command. It should be noted that all converted methods now receive a notification description as an input parameter. Now the

call chain looks as follows: `FormCommand()` -> `CallProcedure()` -> `CallFunction()` -> `ShowDialogBox()` -> `CallFunctionFinal()` -> `CallProcedureFinal()`. The methods with the suffix `Final` are called by sending notifications.

Extract to asynchronous procedure

This command converts the selected syntactically complete fragment of code into a separate procedure or function, converting in addition the selected method to asynchronous form.

Find module calls

The command lists the use of legacy synchronous methods in the current module.

Convert module calls

The command converts all trivial uses of legacy synchronous calls into their asynchronous versions. Conversion is performed within the current module.

Analysis of legacy synchronous configuration calls

This command (Main menu - Configuration - Refactoring - Analysis of legacy synchronous configuration calls) look through the all configuration to search for legacy synchronous calls and to analyze the possibility of their conversion. If in some case the conversion is impossible,- the system indicates the reason for this impossibility.
During the analysis of legacy synchronous calls, the following features of the application are taken into account:

- All modules that can be executed on the client application side are analyzed.
- The analysis of modules that cannot be executed on the client application side is not performed:
  - web service modules;
  - HTTP service modules;
  - common modules that can only be executed on the server side;
  - common modules that can only be executed via external connection.
- The procedures and functions preceded by the following compilation directives are analyzed:
  - `&OnClient`;
  - `&OnClientOnServer`;

    - `&OnClientOnServerWithoutContex`.
- An analysis of the text fragments of procedures and functions that are enclosed in the following preprocessor instructions:
    - `Client`;
    - `ThinClient`;
    - `WebClient`;
    - `ThickClientManagedApplication`;
    - `ThickClientOrdinaryApplication`.

The check box Display methods of objects whose type cannot be defined tells the system that the search results should include methods that are synchronous, but it is impossible to unambiguously determine the object for which they are called.

If the Convert modal calls that are found check box is selected in the search settings dialog box, then only the trivial synchronous calls are converted, and the results of the operation are displayed.

31.2.1.8. Module syntax check

You can check the module being edited for compliance with 1C:Enterprise script syntax.

To check the syntax, click Text - Check syntax.

Syntax check is performed in the following sequence:

- Common modules
- managed application module;
- object module;
- form module.

Modules are checked if the module was not yet checked or was modified.

When checking a specific module, only those modules that are in the list before this module are checked. For example, when checking an application module, only common modules are checked. An external connection module is checked only when it is edited.

If there are errors, their list is displayed in a message window specifying the full address of the location and the description of the error. If you move the cursor over an error message, the pointer turns into a magnifying glass. To navigate to the module line that caused the error, double-click the error message. If the module containing the error is not open, it is opened automatically.

If no errors are detected, a message about the absence of errors in the module is displayed in the message window.

You can enable the automatic verification of the module in the configuration mode of Designer (main menu item Tools - Options, the Modules tab, the Check tab, the Check automatically attribute). In this case, if the module has

been changed, then at closing the module window or at saving the configuration as a whole, the syntax check of the module is performed.
The automatic check mode is convenient when debugging any configuration element.
For full syntactic control of all configuration modules in one pass, select the item Configuration - Module syntax check.
When correcting errors in the modules, you can get a hint on the 1C:Enterprise language by calling Syntax Assistant and finding in it a description of the desired language element.
A hint on a specific language element (operator, procedure, function, property, method) can be obtained by placing the cursor in the module on this language element and pressing Ctrl + F1 key shortcut. Syntax Assistant displays the description for the selected element of the 1C:Enterprise language.

31.2.1.9. Configuration modules access restriction

For some modules you can set an access password. The main purpose of the password -is copyright protection for configuration developers.
When setting the password the following limitations apply:

- the managed application module can't be protected;
- the form modules can't be protected;
- the command modules can't be protected;
- modules that include preprocessor directives can't be protected;
- client common modules operating in a managed mode (thin client, web client and thick client managed mode) can't be protected.

31.2.1.9.1. Setting access password

To set a password, open the required module and select the command Text - Set password. The item is available if the module is opened for writing.
If the module contains preprocessor directives, the program displays a warning: Protected module must not contain preprocessor directives. Continue? If you click the No buttun, the password is not set. If you press the Yes button, the password setting becomes possible, and it is assumed that in this process the directives are deleted. If the directives are not removed, then in 1C:Enterprise mode the methods of such a common module will be unavailable (error Object method <Method name> not found).

Dialog box for entering the password is displayed.

Fig. 314. Setting module password

You need to enter a password and repeat it. To set the password, press the OK button, to cancel it,- click the Cancel button.

31.2.1.9.2. Opening protected module

If an access password is set for the module, then when you try to open the module, a password input dialog is displayed.

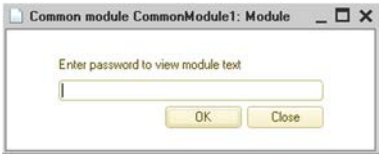


Fig. 315. Opening a module with a password set

If the entered password is correct, the module opens. If the entered password is wrong, the warning Incorrect password is displayed and the module does not open.

31.2.1.9.3. Changing password

To change a password, open the required module and select the command Text - Set password. Dialog box for entering the password is displayed. You should enter the old password. If the entered password is correct, the dialog for entering a new password is displayed.
To remove the password, you need to clear the password fields and, without entering the password values, click the OK button.

31.2.2. Editing text templates

If the Text Document is selected as the template type in the template designer, the text editor is opened in the text template input mode.
Text template editing mode is also available for text documents with the Text template extension set.

31.2.2.1. Text template format

The whole text of the template is divided into areas. Areas must follow each other and cannot intersect or be included in each other. In the template text the areas are highlighted as follows:

```
#Area XXXXX
#EndRegion
```

Where `XXXXX` - is the area name.

To specify end of the area is not nesessary. The declaration of the area beginning means the end of the previous area.

To store variants of a area in several languages, the the area name may be followed by the corresponding language code.

```
#Area Header1 RU
#EndRegion

#Area Header1 EN
#EndRegion
```

If the language code is not specified for some variant of the area, then this variant of the area is returned when the language code that is not specified in any variant of the area descriptions is passed to the `GetArea()` method.

Each area consists of service and text parts.

```
#Area XXXXX

[service part]
[text part]

#EndRegion
```

31.2.2.2. Service part of area

The service part of area has no explicit boundaries. It includes all the lines from the beginning of the area that start with the special character “#”. The service part describes the text template fields used within the area. If service part is not necessary, it may be absent.

The control structures of the text template are highlighted in color.

31.2.2.2.1. The structure of the service area

The service part of the area consists of declarations of some common parameters of the area and descriptions of the field formats included in the area. All descriptions made in the service part of the area are valid only within the region. If the field format is described in a certain area and this field is found in another area without a format description, the default format is used for it in that another area.

At the beginning of the service part of the area, the keyword of the common area `#ReplaceChar A B` may follow. It is used to replace characters within the lines of the area;

- `A` - is a character to replace;
- `B` - is a character replacing the `A` character.

The characters must be enclosed in single quotes.
For example, if you write `#ReplaceChar "@" "#"`, then inside the lines the "`@`" characters not in the field names will be replaced with "`#`".
It makes sense to use this keyword in cases where the design of the lines of the template should include the # character (its direct use would indicate that the text is a service text).

31.2.2.2.2. Field descriptions

The replacement description is followed by a description of the field formats used in the area.
A field is created using the keyword `#Field FFFF`, where `FFFF` -is the name of the field which format is described.
Then the following lines contain keywords to describe the field. The field description is active until the beginning of the next field description.
`#Align {Left | Right | Center | Justify}` - indicates the alignment of the field within its character space.

```
#Field OperationMode
#Align Justify
```

`#Format <Format string>` - specifies the format string for the field output.

```
#Field Date1
#Format "DF=dd.MM.yy"
#Field Tm1
#Format "ND=4; NFD=0; NDS=."
```

`#Block <Parameter>` - indicates the space reserved for the field is filled with symbols “`#`”. If `<Parameter> = True` and the contents of the field does not fit in the allotted space, the rest space is filled.

```
#Field ReportSummary
#Format "ND=18; NFD=2; NDS=."
#Block True
```

31.2.2.3. Text part of area

The text part of the area consists of the lines of the text template. It begins after the last line of the service part of the area, and continues to the area end. Inside the lines of the text template you can specify the name of the field: `[ПППП]`.
The number of spaces reserved for the field corresponds to the number of spaces indicated by the brackets (including brackets). If the field size is one character, one bracket is allowed. Example:

```
![Name ]! [Code ]! [Description ] !
```

If the field name is shifted to the left bracket, it is considered that the field has left alignment, and if the name is shifted to right bracket, then the right alignment assumed. If there are spaces to the left and right of the field name, it is centered within the character space of the field.
If the field name is greater than the character space allocated for the field, the field name is indicated using the `Fields` keyword.

Example:

```
#Field Number
  #Format "ND=3; NFD=0"
#Field Code
  #Format "ND=5; NFD=0"
#Field Remark
  #Format "ND=1"

![ ]! [ Code]![ Description ]! [! #Fields Number Remark
```

Parameters in the `Fields` keyword are specified only for fields with names not indicated in the body of the template in brackets.

31.2.2.3.1. Auto-wrap

If the text does not fit in the allocated field size, it may be necessary to use automatic text wrap to a new line. For this purpose you can use the indication of the location of the field in the template in curly and angle brackets:

- `{ПППП}` - область поля ограничена фигурными скобками. They show that the autotwrap of text stored in the `FFFF` field is possible onto this line, and indicate the area for the text that does not fit in the field `FFFF` before.

In this case, even if the line does not need to display any of the fields, the line is still displayed.

- <FFFF> - the field space is limited by the angle brackets. They show that auto-wrap of text stored in the FFFF field is possible, and indicate the area for the text that does not fit on the previous line in the FFFF field. In this case, if all the fields present in the line are indicated in angle brackets, but none of them are required for text output, the line is not displayed. If, when output text from a field to a line in which the field is placed in angle brackets, this text still does not fit, that line is duplicated until all the text from this field is displayed.

Example:

```
![ YY]! [XX ]![ ZZ]!
! ! {XX }! !
! ! <XX >! !
```

This example indicates that field XX is placed on the first line. The second line contains the part of the XX field that does not fit in the first line. The third line contains part of the XX field, which does not fit in the first and in the second line. If there is no text in the XX field for the third line, it is not displayed at all; and if this text remains, the third line is displayed up to the end of the text from the XX field.

31.2.2.4. Text template keywords description

Area

Description:

Indicates the beginning of the template area. If the previous area is not finished, then the end of the previous area.

Syntax:

```
#Area <Area name> <Language code>
```

Parameters:

<Area name> mandatory

Specifies the name of the area to receive it by the GetArea() method of a text document.

<Language code> optional

Specifies the language code of the area. If you intend to use the configuration in several languages, then you can specify several areas with the same name in the text template. In this case each area should contain a language code. If the language code is not specified, this area is selected when the area is requested with a language code not specified in the description of the areas.

EndOfArea

Description:

Indicates the explicit end of the template area.

Syntax:

```
#EndRegion
```

ReplaceChar

Description:

This keyword allows to replace characters within the lines of the area.

Syntax:

```
#ReplaceChar <Char to replace> <Replacing char>
```

Parameters:

<Char to replace> mandatory

Indicates the character in single quotes to be replaced.

<Replacing char> mandatory

Specifies character to replace `<Char to replace>`.

Field

Description:

Specifies the field for which you want to specify formatting keywords.

Syntax:

```
#Field <Field name>
```

Parameters:

<Field name> mandatory

The field name.

Align

Description:

Specifies alignment when displaying the field contents.

Syntax:

```
#Align <Align parameter>
```

Parameters:

`<Align parameter> mandatory`

Field alignment value. Can take values:

- `Left`,
- `Right`,
- `Center`,
- `Justify`.

Format

Description:

Specifies the output format of the field.

Syntax:

```
#Format <Format string>
```

Parameters:

`<Format string> mandatory`

Defines the format for representing the field value.

Example:

```
"DF=dd.MM.yy"
```

Block

Description:

If the value of the parameter is `True`, then filling mode is set. The "`#`" symbol is displayed in the rest designated space in a field if its value does not fit in the space allocated.

Syntax:

```
#Block <Parameter>
```

Parameters:

`<Parameter> optional`

Determines whether the field should be filled. Can take the following values:

- `True`
- `False`

Fields

Description:

Specifies a list of fields which names cannot be specified at their place.

Syntax:

```
#Fields <Field name 1> <Field name 2> … <Field name N>
```

Parameters:

`<Field name> mandatory`

The field name. It is necessary to specify all field names that are not listed in the text of the template area.

31.2.2.5. Example of invoice print

Let us suppose that the configuration object `Document` called Invoice has a template with name `PrintText` in the list of text-type templates:

```
#Area Header RU
  Invoice # [DocumentNumber ]
  From: [From ]
  To: [To ]
==================================================
| N| Description | Price |Qty| Amount |
#Area Line RU
  #Field Price
    #Format "ND=10; NFD=2; NDS=."
  #Field Qty
    #Align Right
    #Format "ND=4; NFD=0; NDS=."
  #Field Amount
    #Format "ND=12; NFD=2; NDS=."
|  +               +          +    +             |
|[]|[Description ]|[ Price]|[ ]|[ Amount]|#Fields Number Qty
| |<Description >| | | |
#Area Footer RU
  #Field TotalQty
    #Align Right
    #Format "ND=4; NFD=0; NDS=."
  #Field TotalAmount
    #Align Right
    #Format "ND=12; NFD=2; NDS=."
==================================================
 Итого [ ] [ИтогоСумма] #Поля ИтогоШтук

 Director [Director ]
```

This template contains the following areas:

- `Header` - to output the report header;
- `Line` - to output the table data;
- `Footer` - to output totals.

The document form contains the `Print` button. Pressing it triggers the `Click()` event. The handler procedure of this event is placed in the form module.

```
Procedire PrintToText(Button)

  TextDoc = New TextDocument();
  PrintText(TextDoc);
  TextDoc.Show();

EndProcedure
```

The procedure creates the text document `TextDoc` and calls the procedure `PrintText(TextDoc)`, which fills the text document based on the invoice data. The procedure is located in the document module. After filling out the document, it is displayed on the screen.

Text of the `PrintText()` procedure:

```
Procedure PrintText(TextDoc) Export

  // Get the template
  Template = GetTemplate("PrintText");
  // Set language code
  Template.TemplateLanguageCode = "RU";

  // Header
  Area = Template.GetArea("Header");
  Area.Parameters.DocumentNumber = Number;
  Area.Parameters.From = Format(Date,"DF=dd.MM.yyyy");
  Area.Parameters.To = Contractor;
  TextDoc.Output(Area);

  // Processing the table part "Components"
  LineTotal = New Structure("TotalQty, TotalAmount",0,0);
  For Each LineComponent From Components Do

    Area = Template.GetArea("Line");
    Area.Parameters.Number = LineComponent.LineNumber;
    Area.Parameters.Description = LineComponent.Products;
    Area.Parameters.Qty = LineComponent.Quantity;
    Area.Parameters.Price = LineComponent.Price;
    Area.Parameters.Amount = LineComponent.Amount;
    TextDoc.Output(Area);
    LineTotal.TotalQty = LineTotal.TotalQty + LineComponent.Quantity;
    LineTotal.TotalAmount = LineTotal.TotalAmount +
LineComponent.Amount;

  EndDo;

  // Footer
  Area = Template.GetArea("Footer");
  Area.Parameters.Fill(LineTotal);
  IR = InformationRegisters.Responsibles;
  Area.Parameters.Director = IR.GetLast(Date).Director;
  TextDoc.Output(Area);

EndProcedure
```

31.2.3. Editing template texts

In addition to the features of the modules text editor, the template text editor allows you to create and edit existing templates.

31.2.4. Query text editor

In this mode, the text editor in addition to the basic features has a number of additional features.
The syntax of the query language is highlighted in color.
Commands for setting and removing comments are added to the list of available text editor commands.
In 1C:Enterprise mode, users with administrative rights can call the query designer.
In addition to the query text, the query text can be edited for a text document with the Query language extension installed; and for a text document field with the Query Language extension installed.

31.3. Configuration command interface editor

The configuration command interface editor allows you to customize the initial order of sections in the section panel and the initial visibility of sections for different roles.

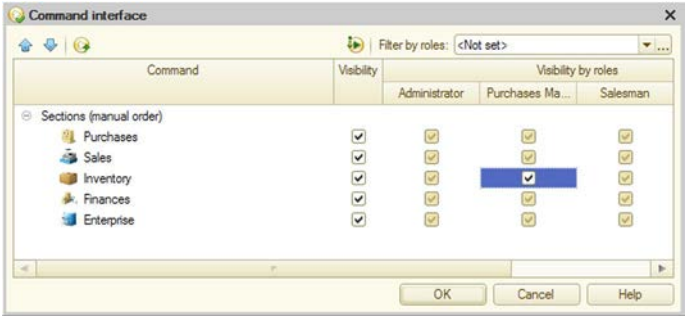


Fig. 316. Configuration command interface editor

In the Command column you can set the order of the sections.
The Visibility and Visibility by roles columns set the default visibility of the commands for opening subsystems.

31.3.1. General rules for setting visibility

The table field contains the general visibility control column (the Visibility column) and as many columns as there are roles defined in the configuration.
Initially, the order in which the commands are displayed and their visibility settings are set by the system. The user can still set up visibility for all commands as well as for specific commands selected by roles to which these commands are available.
Object visibility can be set up in the command interface using several methods:

- Editing the Visibility column. In this case command visibility is set for all roles. The command visibility for a particular role is defined by the Visibility column, if a special visibility check box status is selected for the role (see example of Sales subsystem visibility onfig. 316).
- Selecting a specific visibility value (checked or unchecked) for a particular role indicates whether the command is visible by default (or hidden - depending on the check box status) for the role. Common status (Visibility column) is ignored in this case. Thus, fig. 316the Goods inventory section visibility for the Administrator role is disabled (though common visibility box is checked) and command for navigation to the Enterprise subsystem for the Sales manager role is visible by default (though default visibility for this section is disabled)

Visibility status can be changed for multiple commands from the list.
To show only visible commands in the Command list press the Hide invisible by default button.

31.3.2. Filter by roles

In the Filter by roles field you can specify several roles that will define the current display of the command list. Only commands available to these roles will be included in the list. To disable filtering, select Not set. Use the selection list to enable one of the latest filters quickly.

31.4. Start pagework area editor

31.4.1. Using the start page work area editor

Use this editor to set up a general layout of forms on the start page, as well as the list of forms that can be displayed on the start page.

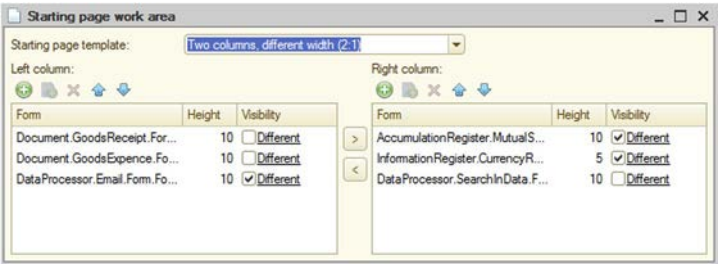


Fig. 317. Start Page Work Area Editor

The user can select a general layout of forms on the Start page in the Start page template field:

- One column-the forms will be displayed in a single column.
- Two columns of equal width- the forms will be displayed in two columns of equal width.
- Two columns of different widths (2:1)-the forms will be displayed in two columns and the left column is twice as wide as the right one.

After the required forms have been selected, the user can specify their sequence order on the start page and height for each form (Height column).
Visibility editor is similar to other role-based properties editors.
When developing the interface of the start page, special attention should be paid to ensure that this interface contains the most important forms. These are forms that are most frequently used by users with a particular set of roles.
Please remember that forms that cannot be viewed by the user due to insufficient rights are not displayed on the start page regardless of the Visibility column status.

31.4.2. Software

You can also configure the composition and layout of forms on the start page (desktop) from 1C:Enterprise script. To perform configuration, you should use

the methods GetForms() and SetForms() of the object HomePageSettings . Thus, it is possible to get the current status of start page (desktop) of the client application, modify the content as needed and set up it again.

Settings for the composition and layout of forms on the start page (desktop) are stored in the standard system settings repository. Settings key -General/HomePageSettings. To change the settings, first read them from the repository:

```
HomePageSettings = SystemSettingsStorage.
Download("General/HomePageSettings);
```

Then the settings of homepage (desktop) can be modified and added back to the repository:

```
SystemSettingsStorage.Save("General/HomePageSettings", "",
HomePageSettings);
```

After saving the new forms composition, you need to rebuild the interface of the client application by calling the RefreshInterface() method.

Among the various uses of the mechanism, there are several basic scenarios:

- It is necessary to invite the user to add a new form to the start page (desktop). This can be done using the program code that is called from the event handler OnStart. After changing the settings of the start page (desktop), you need to update the interface to make the changes available.
- For some users, it is necessary to maintain a fixed set of forms, which is determined not at the configuration stage, but by some data from the information base. In this case, it is recommended to customize the start page (desktop) in the event handlerSessionParametersSetting of session module. In this case, you do not need to call the RefreshInterface() method, since the settings of start page (desktop) are being read after the execution of this handler.
- It is necessary to change once the forms composition on the start page (desktop) for all infobase users. In this case, it is recommended to write a program module that will go over the start page (desktop) settings for all users and make the necessary changes. Please remember, that the changes of interface made in such way are available to the user on the next login only.

Consider an example, when it is necessary to forcefully add two forms on the start page at each start of the system: directory list form Products and the standard form of full-text search (SearchForm). To show an example, find a common module with Server and Server Call check boxes selected. Let the name of the common module is SetupInterface. The common module should include the following procedure:

```
Procedure StandardStartPage() Export

  HomePage = New HomePageSettings;
  FormContent = New HomePageForms;
  FormContent.LeftColumn.Add("Directory.Products.ListForm");
  FormContent.RightColumn.Add("SearchForm");
  HomePage.SetForms(FormContent);
  SystemSettingsStorage.Save(General/HomePageSettings", "", HomePage);
```

ProcedureEnd

The managed application module should include the event handler `OnStart`:

```
Procedure OnStart()
  WorkWithInterface.SimpleHomePage();
  RefreshInterface();
EndProcedure
```

As a result, regardless of the settings of start page (desktop) configured by the user in the previous session, each new session will start with fixed configuration of the start page (desktop).

When configuring the settings of start page (desktop), you can specify both the full names of the forms of configuration objects (for example, `Directory.Products.Form.FormName`) and the standard form names (for example, `Directory.Products.ListForm`). Please remember that the start page (desktop) program settings to not modify the template of the home area (desktop). At the program setup, the start page (desktop) template will be used, which is selected when setting up the start page (desktop) using the appropriate editor.

31.5. Main Section Command Interface Editor

The editor is used to set up commands in each command bar, their display order and visibility of command interface items based on roles.

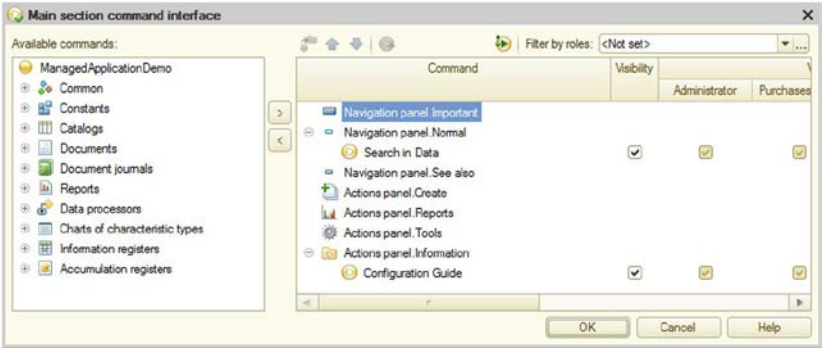


Fig. 318. Editor "Command Interface of the Main Section"

The editor has two tabular fields. The right one is used to edit the command interface, while the left one contains a hierarchical list of available configuration commands that can be added to the main section interface. Commands on the Available Commands list meet the following criteria:

- they have no parameters;
- The group that is specified for the command refers to the navigation bar or action bar.

The Command column of the interface editor table field displays structure of commands grouped by action panel groups (Important, Normal, See also), navigation panel groups (Create, Reports, Service) and groups defined in the metadata tree branch Common- of Command groups . The table field contains the Command overall visibility column. The system also adds commands visibility columns for each role defined in the configuration.

To select a command, expand the appropriate branch in the available commands table and select the command. To move the command to the command interface select it and click Add command to main section (alternatively, press Enter or double-click the command line). The selected command is moved to the panel and group defined for the command during setup of the current metadata object (Group property), the command of which is being moved. If the Add command to the main section button is not available, the selected command cannot be

moved to the main section, e.g., the Use standard commands property is not set for the enumeration.
Initially, command display order in the desktop command interface and their visibility settings are set up by the system. The user can set up visibility for all commands as well as for specific commands selected by roles to which these commands are available.
Use the Move command button to move commands between groups. Commands can also be dragged to the required position within groups of a single panel. To set up command order in a group, use up and down buttons.
Pressing the Set default properties button replaces the command visibility settings and group membership to the system default settings.
If he configuration has no subsystem in it, the editor has a different look:

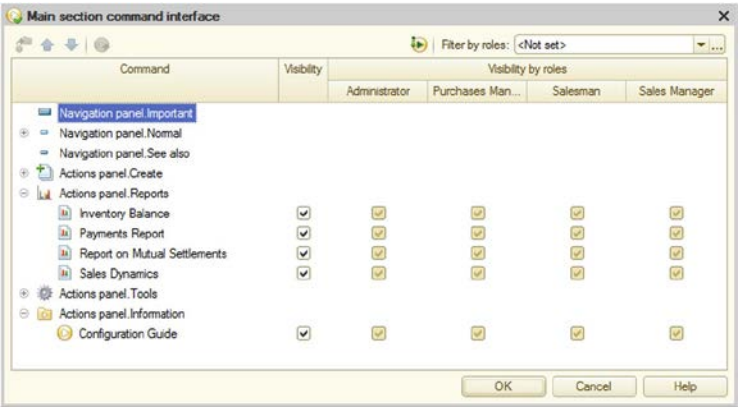


Fig. 319. No Subsystems

In this case, all system commands are available on the desktop, and the user only has to set their default availability.

31.6. Command interface editor

The command interface editor enables you to set up commands in each command bar, their display order and visibility of command interface items based on roles.

The table field in the first (left) column displays a complete list of commands grouped by groups (such as system groups Important, Normal, See also, Reports, Service, etc., and the groups defined in the metadata tree branch Command groups) of action and navigation panels. This is followed by the column that manages general visibility of commands.

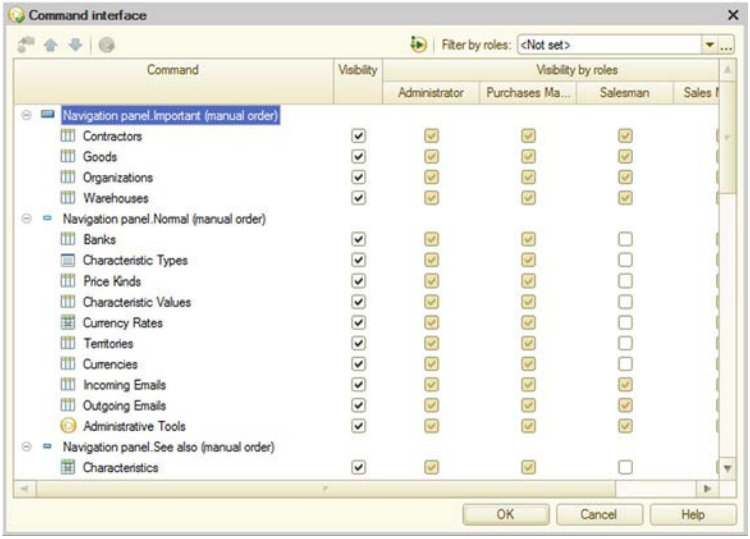


Fig. 320. Command Interface Editor

The system sets up the command order automatically. If you need to manually set up the command display order in groups, use the navigation buttons or drag commands. In this case, the line Manual command order is displayed next to the group. To undo the order changes, select Restore automatic command order in the context menu.

Use the Move command button to move commands between groups. Commands can also be dragged to the required position within groups of a single panel. To set up command order in a group, use up and down buttons.

Pressing the Set default properties button replaces the command visibility settings and group membership to the system default settings.

31.7. All Subsystems Editor

All Subsystem editor is designed to facilitate solving the following tasks:

- editing command interface for any subsystem
- setting contents and properties for any subsystem,
- setting up order of the subsystems
- editing contents of subsystems.

NOTE. The subsystem command interface editor can also be opened from the Command Interface subsystem property using the Open link.

The Subsystems list defines the order of subsystems in the configuration tree. This order does not affect the order of the subsystems in the sections panel.

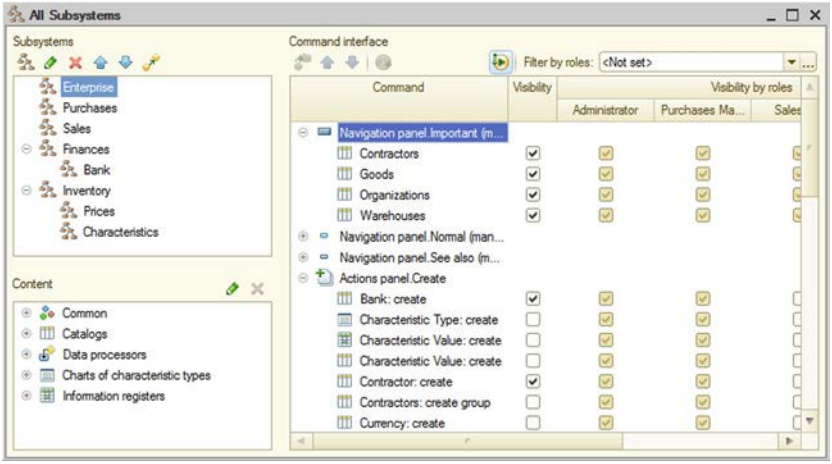


Fig. 321. All Subsystems Editor

To set up subsystem order in the sections panel, go to the desired element in the Subsystems list and perform the editing (subsystem order set up and relocation).
To edit subsystem properties, use the properties palette or subsystem editor (select subsystems and click the Edit current element button).
To move a subsystem (together with subordinate subsystems), select it and click the Move Subsystem button. In the object selection window that appears, specify the subsystem where the current subsystem will be moved. Transfer to subsystems included in the current subsystem is not allowed.

31.7.1. Subsystem Contents Setup

The subsystems list is followed by the Content hierarchical list that is used to define contents of metadata for the current subsystem. Please note, that the displayed contents do not include metadata of subordinate subsystems, if no metadata are defined in these subsystems.
To modify the contents, click the Edit subsystem components button. It opens the object selection window with the metadata tree. The checked objects are included in the subsystem.
In the selection window, specify the objects that are to be included in the subsystem contents. Pressing the OK button closes the selection window and generates the subsystem contents.

31.7.2. Subsystem Command Interface Setting

Subsystem command interface is configured in the table field located to the right of the subsystems and subsystem commands lists.

31.8. Query Wizard

The query wizard enables you to create a query text in the module and edit an existing query.
To call the query wizard, open a module, select a procedure to add the query to, click in that part of the procedure where the query code is or should be located and select Text -Query Wizard.

If there is no query, the user is prompted: The query text not found. Create new? Click Yes to open the Query Builder window. If the query is being edited, the Query Builder window containing the data of the current query appears.

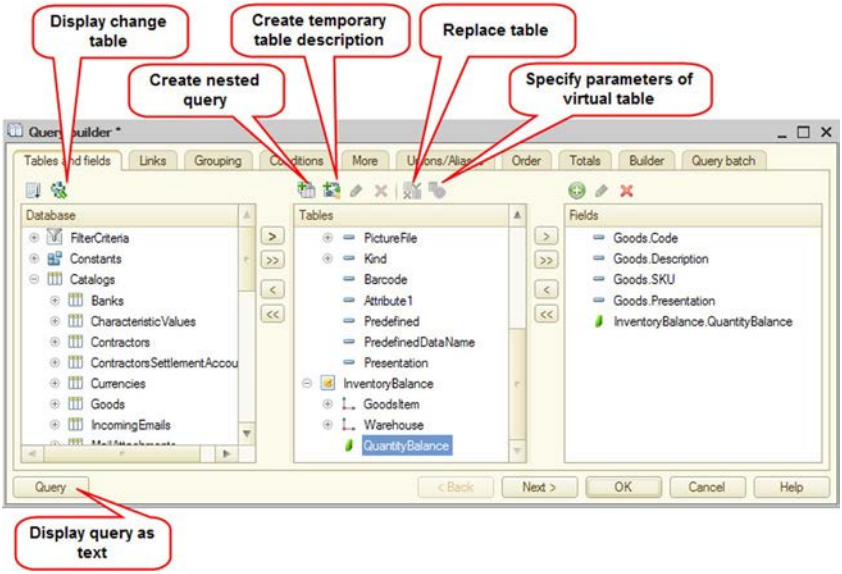


Fig. 322. Query Wizard in Output Form Generator

Click the Sort list button located above the Database list to sort the list of objects. To undo the sorting, click the button again.

Click the Show Change Tables button to show the tables of changes for configuration objects in the list of objects as separate branches <Object-Name>.Changes. To hide the branches, click the button again.

To create a nested query, click the Create a nested query button (see fig. 322). In the new query wizard, create the required query and click OK. This adds the nested query to the Tables list.

To use the created temporary table, click the Create temporary table description button (see fig. 322). In the displayed form, add the table name and names of the required fields and, if necessary, specify the field value type.

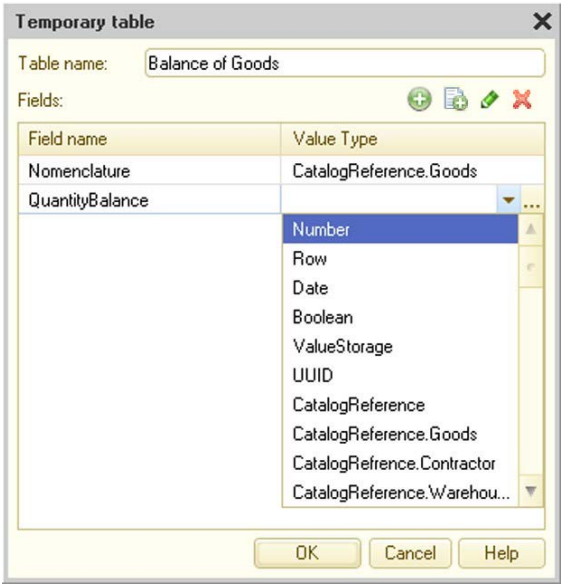


Fig. 323. Temporary Table Description

Using the Next> buttons in the query wizard, work your way through the tabs and select the necessary source data, specify groups and conditions, set the desired order and describe the resulting data. The wizard will generate a form and template in the appropriate branches. You can open a window with text generated on the basis of the specified query data at any time using the Query button.
To correct the data, use the <Back button.
On the Tables and Fields tab, select the required objects and drag them to the Tables and Fields sections.

To specify additional criteria, you can use the custom expression generation mode in the query. To do so, select Add from the Fields list in the context menu. The Custom expression window appears on the screen.

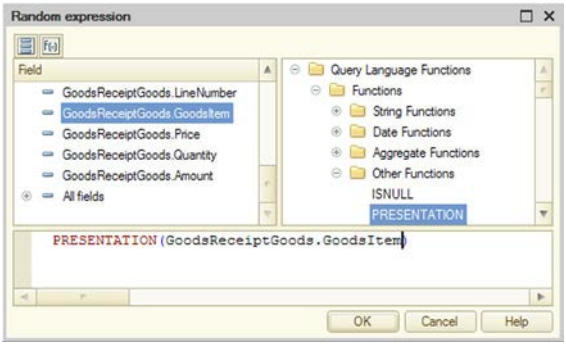


Fig. 324. Custom Expression Wizard

The expression text is generated in the bottom field. Alternatively, you can type the expression using the keyboard. You can drag the desired fields from the fields list and select the desired query language functions from the list by dragging them to the expression text box.
If several tables are specified, the Links tab is added to the query wizard form.

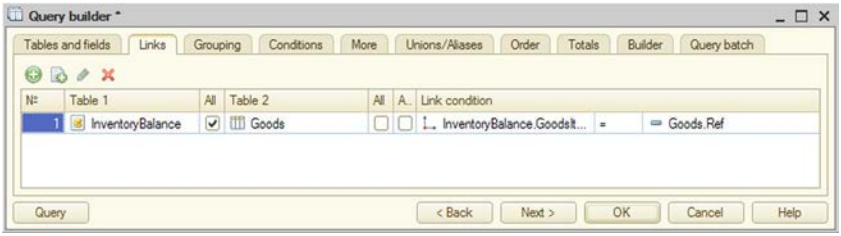


Fig. 325. Define Link Conditions for Tables

The Links tab enables you to specify the criteria for the links between the table fields. Click Add to add a new criterion and select one of the tables from the Table1 column; from the Table2 column select the table, the fields of which are linked with the first table fields. Controls that are used to create table link criteria are located below the criteria list.

On the Grouping tab, if required, select the attributes for grouping.

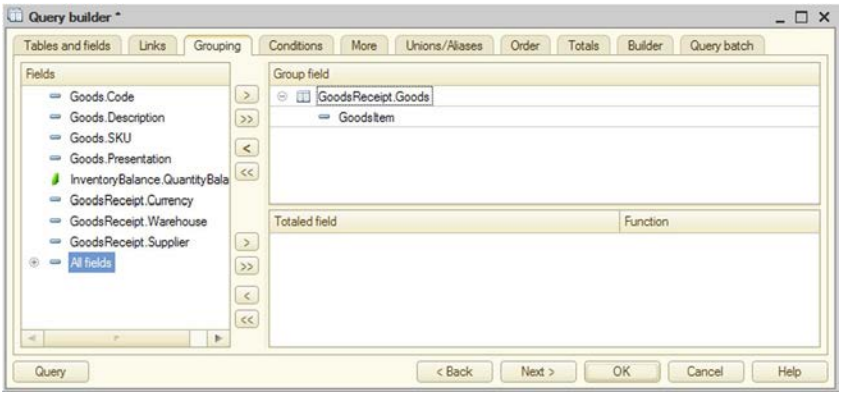


Fig. 326. Query Result Grouping

Specify conditions for source data filtering on the Conditions tab, if required.

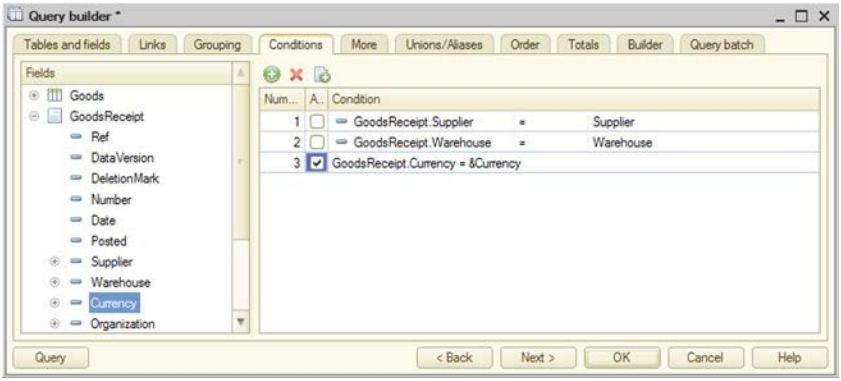


Fig. 327. Query Criteria

For each selected field, choose condition type (for a custom condition, check the box in the Arbitrary column). If the box is not checked, select condition type and specify parameter description. If Arbitrary check box is selected, you can use the custom expression generation window (see above.)

Click the More tab to specify additional criteria.

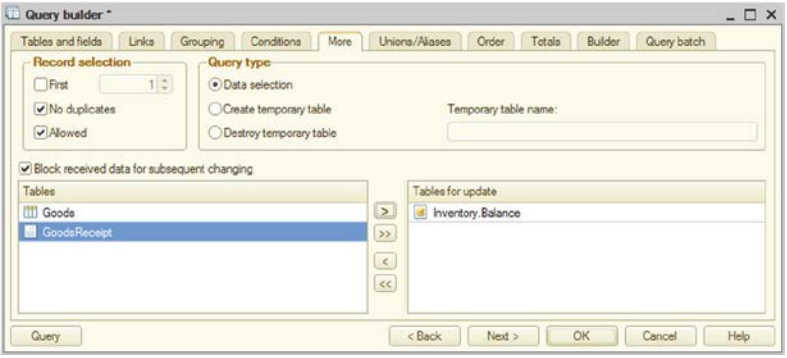


Fig. 328. Additional Query Parameters

If you selected to create a temporary table on the More tab, the fields for indexing can be chosen on the appeared Index tab.

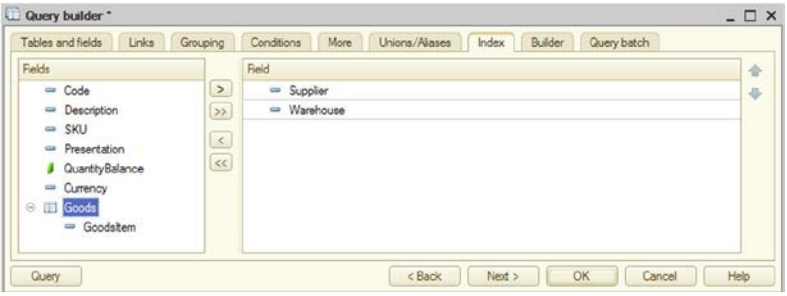


Fig. 329. Temporary Table index

On the Unions/Aliases tab, add filed aliases, if required.

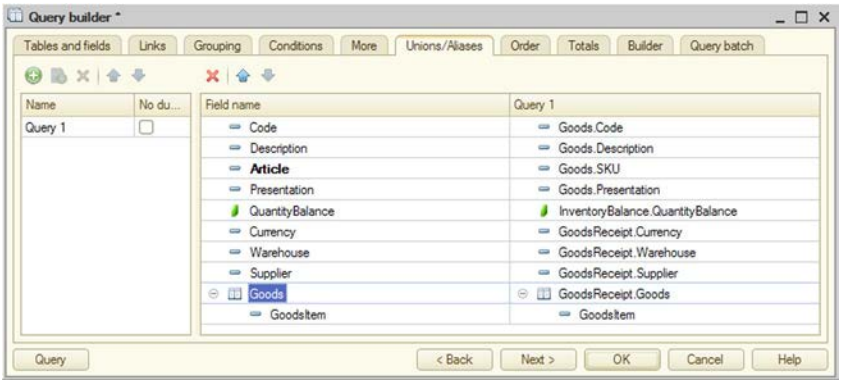


Fig. 330. Query Unions/Aliases

The table shows the mapping between the selected fields and source data. Field names and mappings can be changed. To change the name, select the field and press Enter, then add a new field name. To change the mapping, select the desired row in the Query column and press Enter. Select the desired value in the drop-down list.

If you want to select only unique values, check the box in the No duplicates column.

Aliases for fields that have been changed by the user, loaded from the query text or assigned mandatory aliases by the wizard at the alias generation step are displayed in bold.

On the Order tab, if required, specify the result output order.

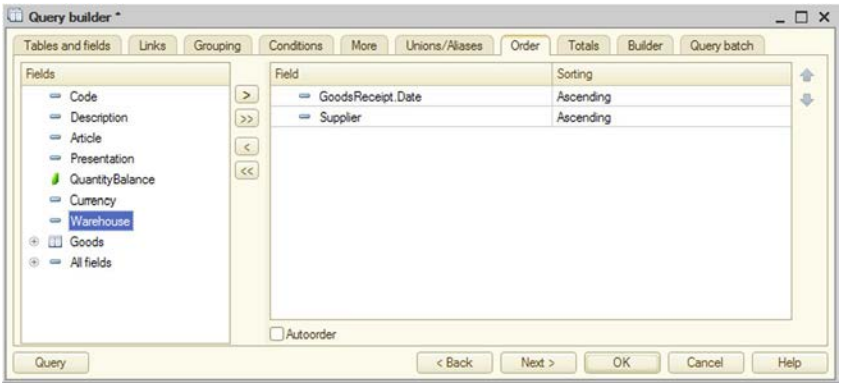


Fig. 331. Customizing the query result ordering

As shown in fig. 331, the output order for the data sorted by date is set, and the data is sorted by Vendor within a single date.

On the Totals tab, if required, specify which fields should be subtotaled and whether overalls should be generated.

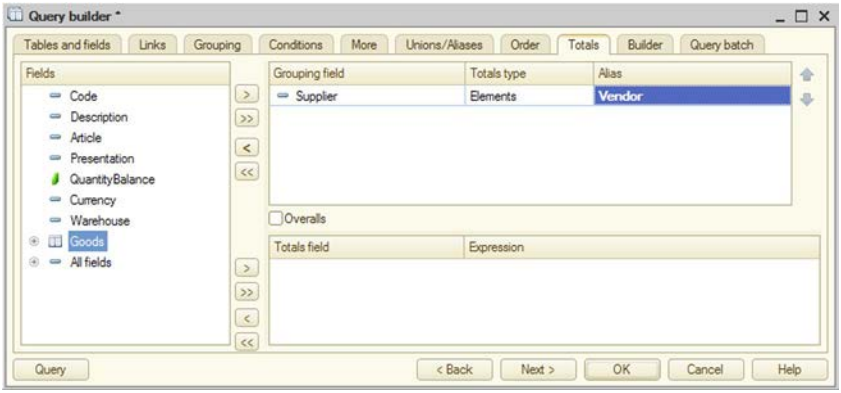


Fig. 332. Setting Totals for Query Result

If you press the >> button located next to the group fields, all reference fields are added to the group fields. If you press the >> button located next to the totaled fields, all numerical type fields are added to the list of totaled fields.

The report builder settings are provided in the Builder tab. Select tables and fields, specify conditions, order of presentation, and describe the total data.

- Use the Tables tab to edit settings of virtual table report builder and mark optional tables. To edit virtual table settings, select the table and call the Virtual table settings command. After the dialog box opens, add the table settings required for the report builder. To mark an optional table, clear Required check box against the table name. For optional tables you can also specify a group number. Optional tables with matching group numbers and adjacent link listing are combined into one optional group. Optional tables with different group numbers are distributed into different groups.
- The Fields tab can be used to select the fields that the report builder uses as available fields in the report output.
- The Conditions tab can be used to select the fields that the report builder uses as available fields for filtering.
- The Order tab can be used to select the fields that the report builder uses as available fields to sort the results.
- The Totals tab can be used to select the fields that the report builder uses as available fields for report grouping.

The Query batch tab enables the user to create a succession of batch queries when working with batch queries. You can edit and configure each query in the same window, and all the previous tabs can be used to switch between the queries being configured.

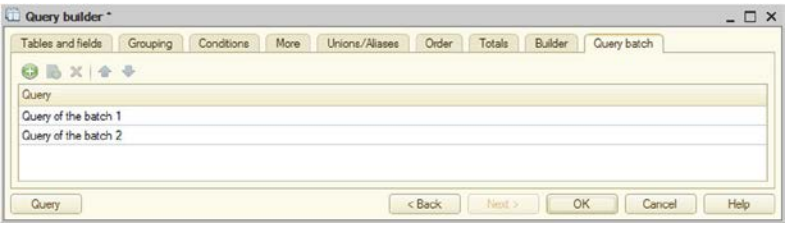


Fig. 333. Query batch

Clicking OK in the program module generates query text. If the wizard attempts to open invalid query text, the cursor is automatically placed in the query line containing the error and a troubleshooting message is displayed.

Depending on the source of query wizard call (data composition system, query with result processing, etc.), the wizard form can have new tabs that are described in the calling mechanism description.

31.9. Query Wizard with result processing

The wizard is designed to generate program code for query results processing. This wizard helps create the following options of query results processing:

- standard iteration over query results
- outputting query results to a spreadsheet document,
- outputting query results to a chart.

To call the wizard use the Query wizard with result processing context menu in the module editor.
When called, the wizard searches for a wizard-generated code in the current line. If the code is found, the wizard loads it (including templates names, etc.).
The query wizard window differs from the query editor described above.

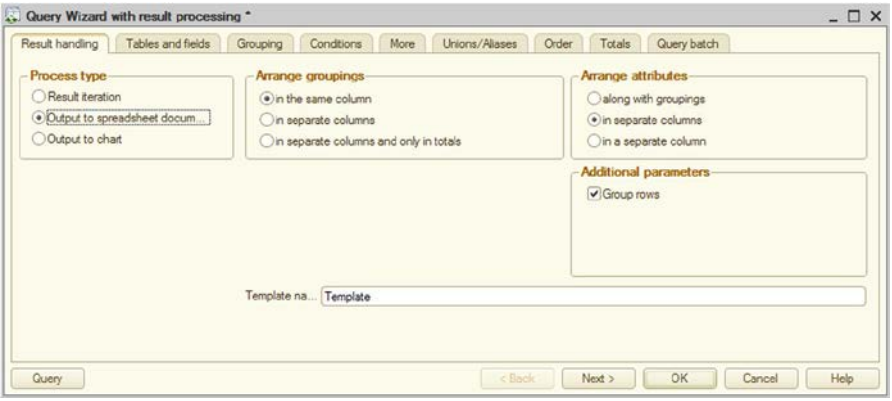


Fig. 334. Query Wizard with result processing

If the wizard is opened in this mode:

- Query result processing type is selected at the Result Processing tab:
  - Result iteration,
  - Output to spreadsheet document,
  - Output to chart.
- If Output to spreadsheet document or Output to chart is selected, parameters for output into these objects are displayed.

Clicking OK generates the program code to be inserted in the current module position. If the wizard has been called for the existing code, the old code is replaced.

If the results output to a spreadsheet document, a template is created in the metadata object, the object or form module of which has called the document. If you call the wizard again, the template is modified. This template is removed if the wizard was previously used to create a spreadsheet document, and is now called to generate another method of query results processing. If the wizard is called in a common module, then a common template is created.

NOTE. If the configuration editing mode is set for modes of managed application startup, the command to call the output forms generator is unavailable in the menu of metadata objects.

31.10. Register Records Wizard

Use the register records wizard for documents.

To launch the register record wizard, go to the Records tab in the document editing window, specify the list of registers that will be affected by the document and press the Register records wizard button. If a list of records for at least one register is defined for the selected document, this item is available in the context menu.

When launched the wizard asks the user to select a register for which the `Posting()` procedure is generated and opens the wizard window.

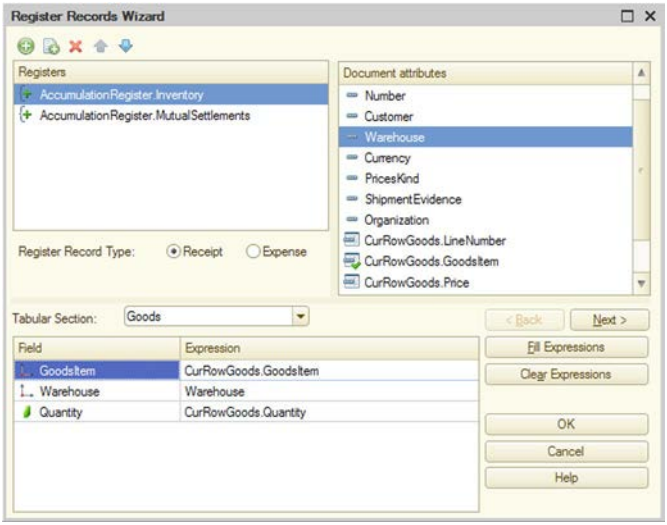


Fig. 335. Register Records Wizard

Initially, the list contains only the specified register. You can modify the list by adding registers from the list of registers marked on the Records tab of the document editing window.
For each register, depending on its type, preliminary settings must be specified: for balance accumulation registers, specify the type of register records; for accounting registers that do not support correspondence, specify the - correspondence type and the account; for supporting correspondence registers, specify the - debit and credit accounts.
If the document has tabular sections, and their data should impact the register status, then the incorporation of a tabular section is made in the selection list of the Tabular section.
Then, the attribute formulas are made for each register, according to the details of the document.
You should add to the list the formulas defining how to calculate the register records for the selected document attributes.
These formulas can be created "manually" as follows. The formula is added to the Formula column for each register attribute selected in the list. Formula can be manually entered or edited in this box. The wizard does not verify the formulas accuracy.
You can also add the corresponding attribute data in the formula by double-clicking in the list of the document attributes. The wizard does not check the matching of selected attribute types.
If at least one type belongs both to the attribute register and to the register dimension/ resource and their names match, Autofill mode can be used for these subordinate objects. When you press the Fill Expressions button, the editable attribute name is added in the Expression column.
The wizard creates a `Posting()` procedure in the object module. The wizard displays the following warning at the beginning of the procedure: This fragment was built by the wizard. All manually made changes will be lost next time you use the wizard!

31.11. Print Wizard

The print wizard is designed to create a template with specified named areas and print procedures for a configuration object.
The print wizard works with the following configuration objects:

- Catalogs
- Documents
- Document journals
- Charts of characteristic types

- Charts of accounts
- Charts of calculations
- Exchange plans.

Basic techniques for working with the print wizard are discussed below on the example of working with the catalog. For other configuration object types, there may be some differences in working with the wizard.

To start up the print wizard point to the desired object in the Configuration window. Select Wizards Print Wizard- in the context menu.

Because the result of the print wizard operation is a print procedure located in an object manager module, the wizard begins by checking the accessibility of that module. If the object module has restricted access, a password dialog box appears. After you enter the password, the wizard window appears.

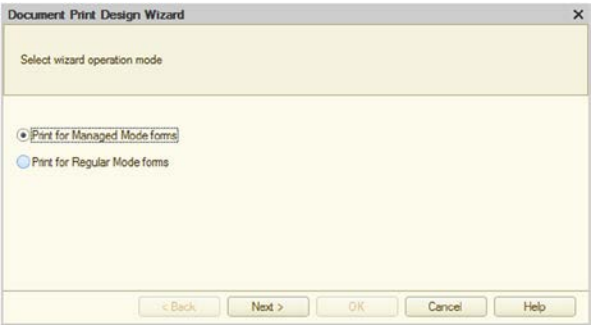


Fig. 336. Select Work Option

First, the wizard prompts to select the startup option for which the print module is to be generated.

Then it asks the user to create a new print command or modify an existing command. Name must be specified for a new command. The default command name is Print. The specified name is subsequently used as:

- object command name
- command module server method name;

- method name in an object manager that implements the print procedure.

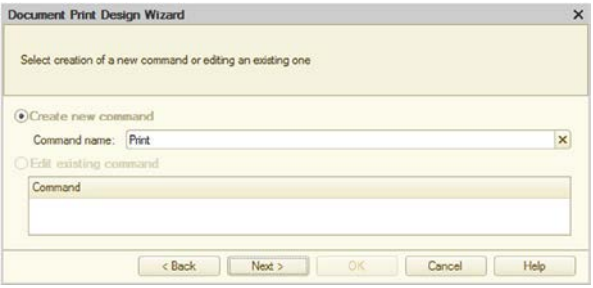


Fig. 337. Print Command Creation

If the user chooses to edit an existing command, the following texts are replaced in the command module: `CommandProcessing()` handler, server method with the command name (-`Print()` in this case), object manager module method with the command name and template.
After you press the Next > button, the wizard switches to selection of header attributes. Create the attribute list using the transfer buttons.

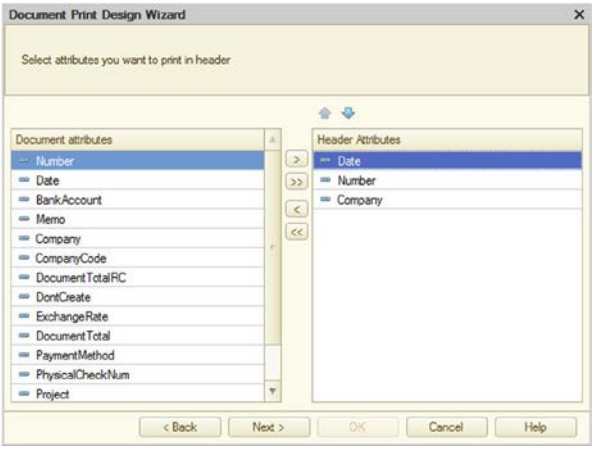


Fig. 338. Header Attribute Selection

If the configuration object has tabular sections, then after the Next > button is pressed the wizard switches to selecting attributes of the next tabular section. The print list is generated the same way as the header attribute list.
If the configuration object has tabular sections, then after the Next > button is pressed the wizard switches to selecting attributes of the footer.
Then the user should select the group for the created command and specify certain parameters of the generated document.

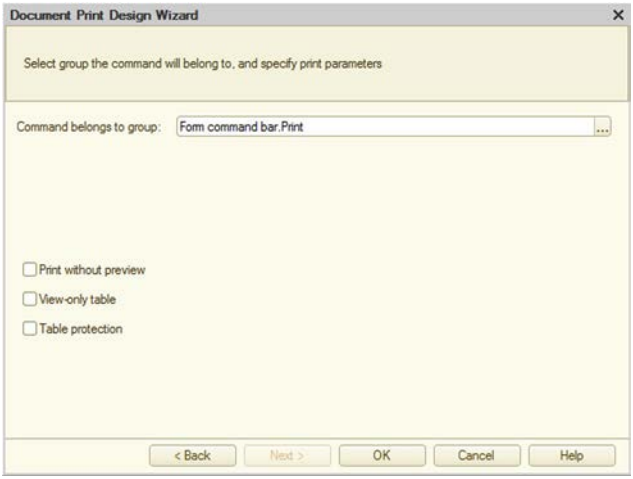


Fig. 339. Print Wizard Settings

Clicking OK closes the print wizard.
The wizard does the following:

- Template template with a print form is created for the selected object (form name matches the created command name).
- Command is created for the selected object; it prepares the spreadsheet document and calls the server procedure (with the command name) that fills the transferred spreadsheet document.
- The procedure is generated in manager module of the selected object, and its settings can be configured to print several objects (list) at once. The created command is assigned rights similar to the View right for the object that initiates wizard call.

31.12. Create based on Wizard

The Generation settings wizard makes it easier to develop a procedure to be used to generate a new object.
This wizard can be used for the following objects:

- catalogs,
- documents,
- charts of characteristic types,
- charts of accounts,
- charts of calculation types,
- exchange plans,
- business processes
- tasks.

To launch the wizard point to the desired object in the Configuration window. Select Wizards - Generate Wizard in the context menu. If Generate mode is defined for the selected configuration object, this item is available in the context menu.
When you launch it, the Create Based On... Wizard window opens.

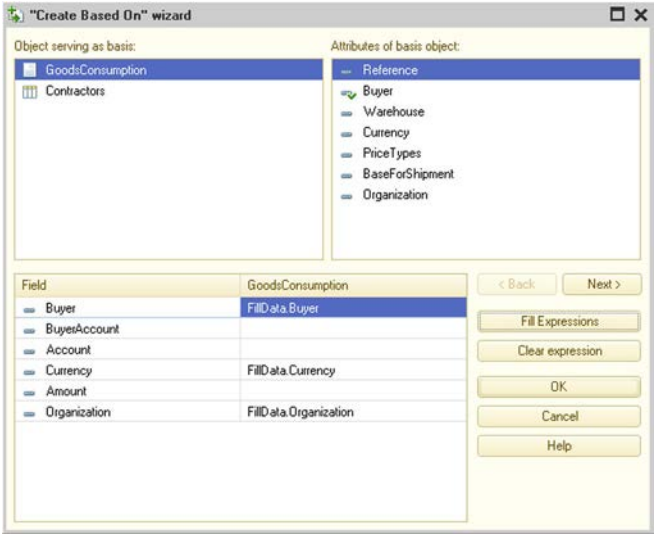


Fig. 340. Create based on Wizard

In the upper part you can see a list of base objects and a list of attributes for the base object selected in the first list.
The lower part displays a list of attributes for the resulting object. The object attribute list must be supplemented with expressions to specify how the document attributes are filled from the selected attributes for the base object.
These formulas can be created "manually" as follows. An expression is defined in the attribute entry Formula text box for the attribute of the object selected in the list. Formula can be manually entered or edited in this box. The wizard does not verify the formulas accuracy.
By double clicking in the list of attributes of the base object you can also insert correctly represented data of the corresponding attribute in the attribute Formula text box. The wizard does not check the matching of selected attribute types.
You can also combine these two methods: when added in the input field, data from the list of attributes of the base objects do not replace all of the information already there, but only the selected characters.
Clicking the AutoFill Expressions button after request and confirmation creates filling formulas for base object attributes. Auto-filling does not change the previously filled formulas. Document attributes are matched with base object attributes according to the attribute names, IDs and types.
Clicking the Clear Formulas button after request and confirmation clears all formulas created both automatically and manually.
When formulas are created for a base object selected from the base object list, they are retained in memory when a different base object is selected. So without ending the wizard session, you can create "Create-on-basis" procedures for several base documents. You can also resume editing the procedure for any base object by selecting it in the list again.
To end your wizard session use the OK or Cancel buttons (with and without saving your changes, respectively).
The wizard creates a `Filling()` procedure in the object module. The wizard displays the following warning at the beginning of the procedure: This fragment was built by the wizard. All manually made changes will be lost next time you use the wizard!
The next time the wizard is used, all formulas previously generated for any base document appear in the document attribute list when the corresponding base document is selected in the base document list. The wizard makes formulas generated in its previous session available for viewing and editing, both through Fill formulas automatically and manually. Moreover, it also takes formulas entered manually into the document module into account if they are added in the procedure it generates. The wizard does not verify the formulas accuracy. For example, if filling with two different base document attributes is specified for a document attribute, the wizard selects one for display (by alphabet). This

is the one that remains in the module after the wizard results are updated. The second line corresponding to the same attribute is deleted.

31.13. Configuration Object Form Wizard

The Form Wizard is launched whenever a new form is added to any configuration object that can include a subordinate Form type object. This special wizard is used to select the form type and set the configuration object attributes. The wizards have a great deal in common, despite substantial differences between the types of objects for which forms are created. Therefore, the use of form wizards will be shown using the document form wizard as an example.
When creating a new form object, the screen displays the form wizard.

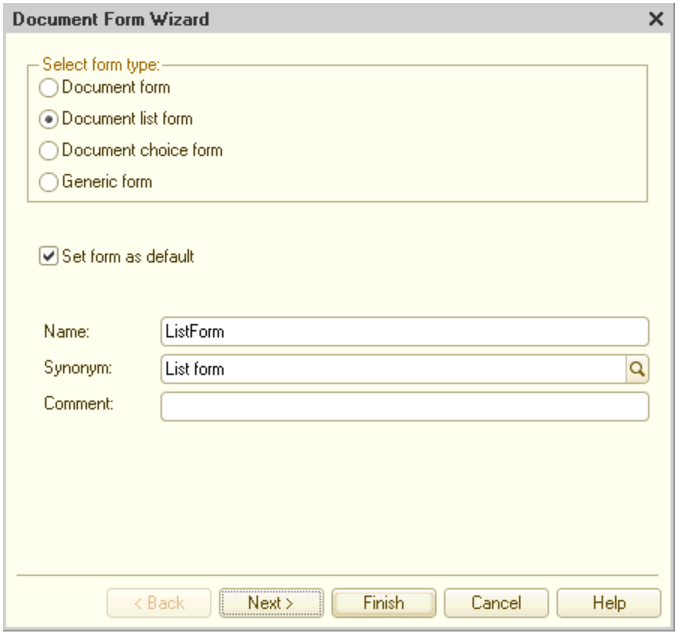


Fig. 341. Form Wizard

The selections available under Select form type depend on the type of object for which the form is being created. The following forms can be created for the `Document` configuration object:

- document form;

- document list form;
- document selection form;
- generic form (blank form).

The number of forms for each type is unlimited. If the object has multiple forms of the same type, you can choose one of them as the default form. When you open an object without specifying explicitly which form to call, the default form opens. To set a form as the default select Set form as default check box. The default form selection can be changed later, on the Forms tab in the object editing dialog box

Selecting the Generic form type does not result in creation of a main attribute. In this case the form behaves in the standard way. Selecting another form type results in creation of a main attribute and sets a different form type behavior. These exceptions are defined in objects known as form extensions (see descriptions of relevant object in the 1C:Enterprise script help).

Form type selection determines the main attribute type and consequently the entire form behavior, including the available toolbar commands.

Specify the form name, synonym and comment.

The functions of the command bars are determined by the action source and type of the form main attribute.

When the Finish button is pressed, the wizard automatically places the controls in the form and opens it for editing.

If form attributes have to be changed, click the Next > button. The composition of attributes depends on the form main attribute.

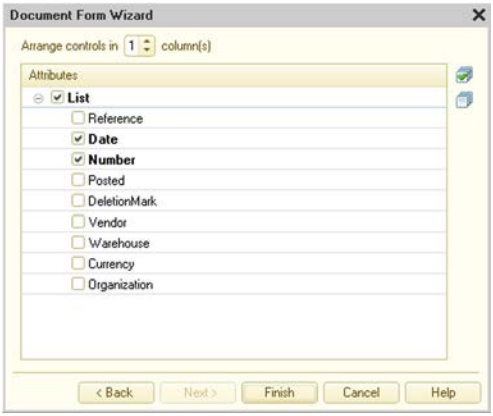


Fig. 342. Form Attribute Selection

The attributes to be placed in the form are chosen at this point in the wizard. Selections are made in the Attributes column by checking the box to the left of the entry.

IMPORTANT! The form wizard does not include non-displayable attribute types (such as `ValueStorage`) in the list.

31.13.1. Constant Form Wizard Features

Each constant can have its own editing form. To create this form select Create Constant Form in the context menu of the relevant constant.

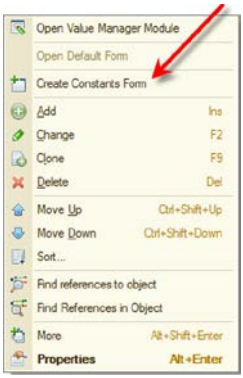


Fig. 343. Creation of Constant Editing Form

Executing this command opens the common form wizard that lists all system constants as data, and the current constant is the only selected attribute (by default).

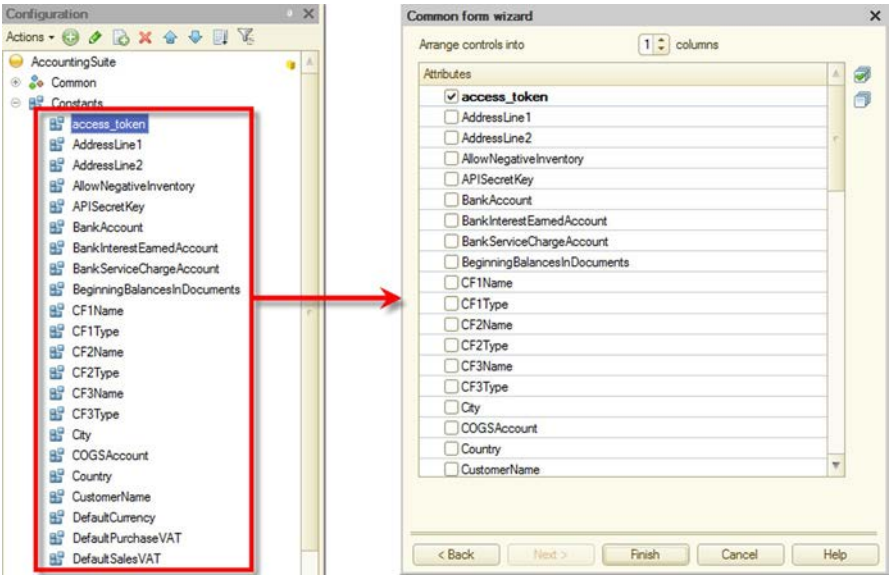


Fig. 344. Selection of Editable Constants

The same form is auto-generated by the system, if the constant has its Main Form property non-filled and its Use standard commands property is set.
To create an editing form for multiple constants, create a common Constants Form first and then select the constants to be edited on the attribute selection page.
Constant forms is included to the command interface of the subsystems that include the constant (if the Use Standard Commands property is set for the constant) and the common constant editing form (if the Use Standard Commands property is set for the common form).
If an editing form is created for the constant with the property Use Standard Commands set, the common form wizard clears the Use Standard Commands property for the form. Therefore, the constant editing forms are only included in the command interfaces of the subsystems that include the constant.

31.14. Template Wizard

Use the template wizard to create configuration object templates and common templates.
The wizard window appears when you select Actions - Add if the branch of the subordinate Templates or Common templates object is selected.

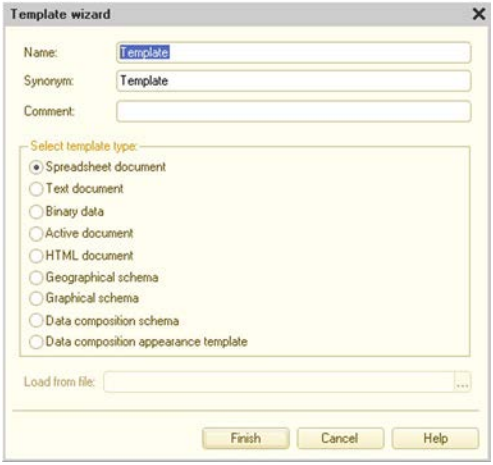


Fig. 345. Template Wizard

Specify the name, a synonym and a comment and choose the template type
The Spreadsheet document type assumes you use the standard 1C:Enterprise method to create and use spreadsheet document templates.
Text document template type assumes the use of text documents specially prepared as templates. For a text document, the Extension property means the Text template.
Selecting the Binary data type means that the configuration developer knows how to work with this type of object.

The Active document type enables you to use the OLE Active document technology. If you press Finish, the wizard searches for available Active document types and offers to select one of those it finds.

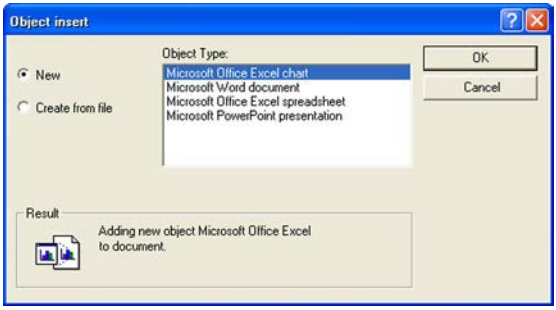


Fig. 346. Active document Insertion

You can select a ready Active document and use it as a basis for template creation (load template prototype from a file). If you selected a file, the type of which is not on the list of supported Active documents, the wizard displays a warning. Selected document is saved in the configuration. Working with the template, you must use properties and methods provided by the selected platform.
If you select the HTML document type, the HTML template editor opens. This mode supports all features of the HTML editor. In addition, the HTML template editor enables you to use pictures from the picture library, from a file, from clipart or from "internal" pictures. Internal images - are those already selected from an image file and stored "inside" the template.
To add a picture in a template, select Elements- Picture...

Fig. 347. Adding pictures to HTML documents

To select a picture press the selection button. Select the required picture in the window that opens.

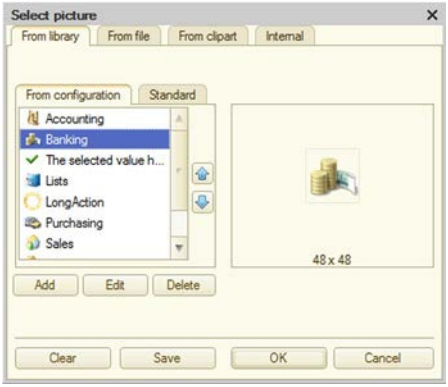


Fig. 348. Picture Selection

Select the Geographical Schema type to create templates using documents prepared with the geographical schema editor.
If the Graphical schema type is selected, you can use graphical schemas prepared in the graphical schema editor or download schemas from files and use them as templates.
The Data composition schema type opens the data composition schema wizard.
Selecting the Data Composition appearance Template type opens the appearance template wizard window.
Select the Add-in type to download the add-in in zip-archive. The archive should include the files of external components and a special manifest. External component loaded in this way can be used both in a regular application and in a mobile application.

31.15. Format String Wizard

Use the format string wizard to write expressions using numeric, date and logical expression representations.
To open the wizard place the cursor over the required module text area and choose the Text - Format String Wizard command. The editor displays the following message for the new string: Format string not found Create new format string? Clicking Yes displays the wizard window.

The wizard can also be called from Format and Edit Format properties of form element. In this case click the selection button in the appropriate property.
Window tabs correspond to the data type of the format string to be generated in the window.
Controls for selecting the presentation format are located on three tabs, according to the data types:

- number
- date
- logical values

Language (Country) field (L parameter) defines information presentation based on the regional settings. Values specified in the following tabs take precedence over country settings.
Select the Number tab for numbers.

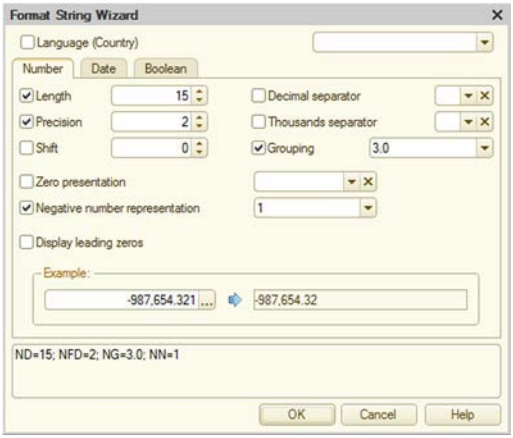


Fig. 349. Format String Wizard for Numbers

Length field (ND parameter) specifies the total number of decimal places (for the integer and fractional parts). Precision field (NFD parameter) states the number of decimal places in the fractional part.
Shift field (NS parameter) specifies the digit shift. If the field value is positive, then the number is divided by 10 to the appropriate degree, and if the field value is negative-, multiplication is performed.
Decimal separator field (NDS parameter) specifies the decimal separator.

Digit Group Separator field (NGS parameter) specifies the group separator. When separator is set to an empty string, non-breaking space is used as a separator.

Grouping field (NG parameter) specifies the option for grouping digits in the integer part of a number.

Zero presentation field (NZ parameter) is used to select how zero values are represented. This parameter can be used in a format string, but it is not used for text boxes.

Negative number representation field (NN parameter) specifies how negative numbers are represented.

Leading zerosare displayed when Display leading zeros (NLZ parameter) is selected.

TIP. The bottom part of the wizard window displays format string results (Example group) and the format string itself.

Select the Date tab to set the date.

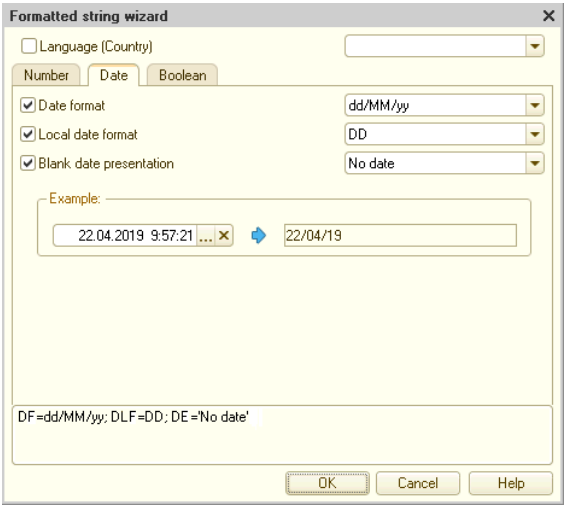


Fig. 350. Format String Wizard for Dates

The date presentation option is selected in the Date format field (DF parameter). It is recommended to use this format in the case when you need to get exactly one or the other date format, regardless of the localization parameters. For example, for the format string DF = yyyy-MM-dd, the date presentation always looks like 2015-12-01.

The local date presentation option is selected in the Local Date Format field (DLF parameter). No selection of DD parameter value is allowed for text boxes. It is recommended to use this date presentation option when you need to take into account the current localization parameters. Localization parameters can be specified both directly in the format string and can be generated from the regional settings of the infobase.

Example:

```
Format('20151201', "L=ru;DLF=D") = 01.12.2015
Format('20151201', "L=en_US;DLF=D") = 12/1/2015
Format('20151201', "L=ru;DLF=DV") = 01.12.2015 0:00:00
Format('20151201', "L=en_US;DLF=DV") = 12/1/2015 12:00:00 AM
```

The blank date representation option is selected in the Blank Date Representation field (DE parameter). This parameter can be used in a format string, but it is not used for text boxes.

TIP. The bottom part of the wizard window displays format string results (Example group) and the format string itself.

For Boolean type data, select the Boolean tab.

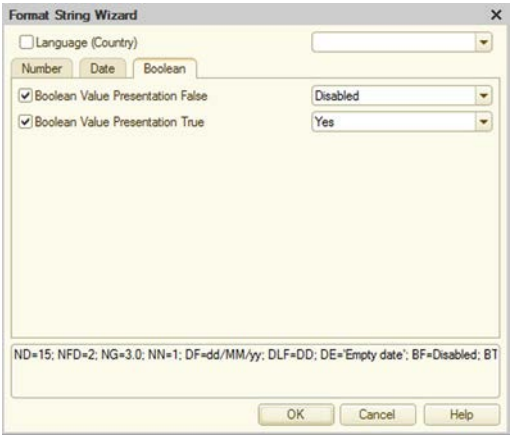


Fig. 351. Format String Wizard for Logical Value

If a value is not selected, default settings are used.
For details on format string parameters see description of the FormatString parameter for the Format() method in the 1C:Enterprise script description.
If the format string wizard is used, the generated format string is inserted into the text (or text box) when clicking OK .

TIP. The text editor can be used to place the cursor in the format string and edit it by selecting Text- Format String Wizard in the menu.

31.16. Wizard for Strings in Different Languages

Use this wizard to edit strings in all configuration languages and other languages, if their codes are included in the edited set, but not in the configuration.
The wizard produces the following string:

```
<language_code1> = <String_1>; <language_code2> = <String_2>;…
```

It is convenient to use the wizard to create the `InitialString` parameter for the global context function `NStr()`.
To open the wizard place the cursor over the required module area and choose Text - Multilingual string wizard. The editor analyzes the module text where the cursor is located and, if the construct is found, opens the wizard window. If the construct is not found, the wizard informs the user about it.

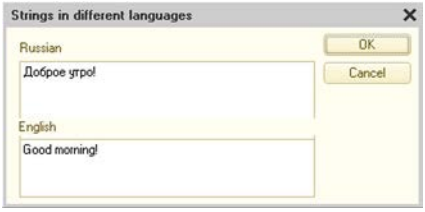


Fig. 352. Wizard for Strings in Different Languages

The wizard window contains as many text boxes as there are languages in the configuration.
Add text and press OK. The wizard produces the following string:

```
ru = 'Доброе утро!'; en = 'Good Morning!'
```

31.17. Spreadsheet document editor

The Spreadsheet Editor in 1C:Enterprise is used to create various print forms and forms for viewing and adding information in spreadsheet documents.
For a table of shortcuts for the spreadsheet editor see Help menu.

31.17.1. What is a spreadsheet document in 1C:Enterprise

Although 1C:Enterprise spreadsheets may be used for adding, processing and displaying information like standard spreadsheets (using spreadsheet documents in a form), they are mainly used to view processed information, e.g., a description of a printable report form. Program modules in the 1C:Enterprise language are used to process information and store it on the spreadsheet for most configuration objects.

The 1C:Enterprise spreadsheet editor is used to work with spreadsheet documents and print form templates.

Generated spreadsheet documents can be saved to disk in various formats. If a document is saved in MXL format, it can later be viewed in the file manager which is available at: http://v8.1c.ru/metod/fileworkshop.htm. It is not recommended to include OLE objects in a saved spreadsheet document if it is to be used in a web client or other that Windows operating systems.

Templates are stored in the configuration. There are common templates (located at Common - Templates branch, e.g., templates of standard payment documents printed based on different documents) and templates for specific configuration objects (e.g., a fixed asset card). A configuration object may have several print form templates.

Spreadsheet documents may also be placed in a form. For this purpose, use the Spreadsheet document field control. In this mode other controls can be inserted into a spreadsheet document.

Templates and spreadsheet documents in terms of work practices are almost the same.

Spreadsheet documents (files and templates) may be compared and merged.

31.17.2. Templates

Templates can be created in the template wizard and the query wizard with result processing.

31.17.2.1. Designing templates

Designing a template involves creating its components and building blocks - (named areas) that constitute an output form - report. Since nearly all business documents have a rectangular structure, it is best to create templates for these documents in an editor that supports rectangular elements.

This feature is provided in the 1C:Enterprise spreadsheet editor. When creating a template, you may add texts into spreadsheet cells; configure formatting settings (for the entire cell); modify row height and column width; add drawn elements - lines, rectangles and other graphical elements to templates: pictures, OLE objects and charts, as well as various controls; specify appearance for the entire spreadsheet or separate cells and cell groups.

When creating a report template, follow these guidelines.

Basically every report has a header with the name of the report and source building parameters. To generate such data, a named area should be created, which is often called the Header. If a report contains a tabular section, the header usually contains column names.

Create named areas (Row in the example on fig. 353) for outputting information to spreadsheet. It is normally impossible to know the number of spreadsheet rows in a template in advance; however, the structure of information is always the same. Therefore, the same areas should be used for each row description, when creating a report. Various parameters (in the example shown, they are Account, Name, Price, Amount, etc.) should be assigned to cells for outputting specific information. When displaying the next row, the contents of each output row are assigned to these parameters, and then the generated area is included in the report. This procedure is repeated until all information on the spreadsheet is output. Normally, the report ends with the output of resulting data and details of the responsible persons. This data is usually located in the area called Footer.

A finished template consists of several rectangular areas representing parts of the report: report header area (name, date, etc.), tabular section header area, etc.

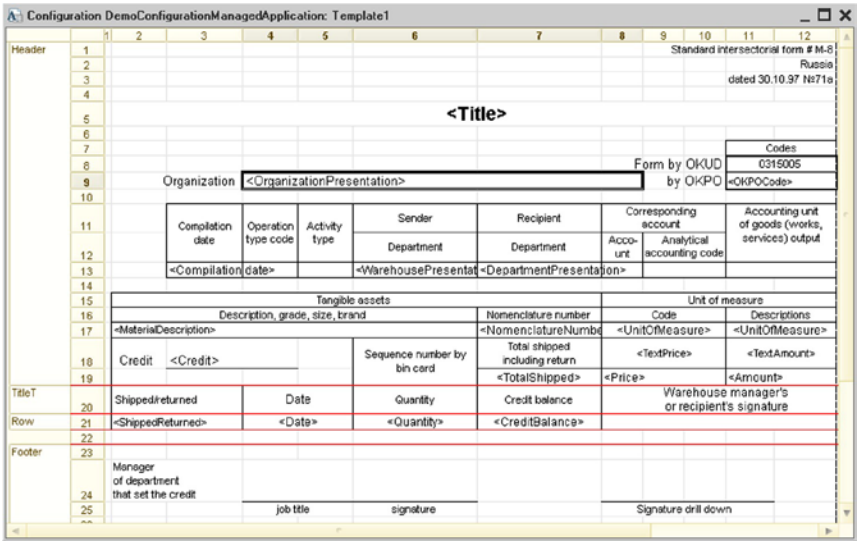


Fig. 353. Sample Template

The report generation procedure is described below.

- Basically, reports are always created as blank spreadsheet documents. Please note that a template is not directly linked with the finished report. The template is a kind of wizard or set of areas used by the program module to generate a report.
- When the report algorithm is processed, named areas are extracted from template in the required order, translated (parameters are replaced with values) and copied to a spreadsheet document of the report. Reporting language features allow to expand reports both horizontally and vertically.

31.17.2.2. Template Cells Properties

Template Property Category

The Template properties category is shown if the Template option is enabled in the spreadsheet document properties.

Fill - the Fill property items specify what kind of information is added into a cell. This option is shown, if the Values category does not have the Contains Value option enabled. It does not modify cell appearance and is used only during template processing when the final spreadsheet document is generated.

This list items have the following meaning:

Data format	Description
Text	The cell contains text information which is to be moved from the template into the final spreadsheet document without any changes.
Parameter	Cell information is a parameter with the name specified in the `Parameter` property.
Template	Information in the cells is the text with the parameter names enclosed in square brackets. When creating a spreadsheet document, variables will be calculated and included in the text. The space allocated in the text to display the parameter values is determined by the length of these values. Sample template Director Director [Директор]

Information in the cells of the final spreadsheet document is transformed to the `String` type.

Parameter-the name of parameter for cell contents. This property is displayed if the Values category contains the Contains Value property or when the Fill property of the Template category has the Parameter value set.

Decryption Parameter- specifies the parameter for processing cell value decryption. It is reasonable to use this property when the resulting spreadsheet document is located in a form (Spreadsheet Document Field control).

Fig. 354. Cursor Types in Spreadsheet Document

When a finished spreadsheet document is opened in the Read Only mode and the mouse pointer is moved over a cell with the Decryption Parameter property filled, the mouse pointer can take the form as shown on fig. 354(cursor of the-Hyperlink type is only displayed if the Hyperlink property is set for a link cell). This means that you can view the details (decryption) of a spreadsheet document data. Now, if you double-click this cell (or make it active and press Enter , or simply click it if it is a hyperlink), the field value will be displayed:

- String, Number, Date and Enumeration values will be opened for viewing.
- For Document values, a corresponding document will be opened for viewing and editing.
- If a value is a catalog item, this item is opened for viewing and editing in a dialog box. If editing in the list feature is turned on in the catalog properties, a catalog list form will be opened and the spreadsheet field pointer will be moved to the required catalog item.

This decryption processing is called standard, and does not require any additional settings.

If you need a special decryption processing, specify the name of the procedure that handles the event occurring when you select a cell with decryption, in the spreadsheet document properties in the Events category for the Decryption Processing event. In the module form in the body of this procedure, you will need to describe the decryption processing using 1C:Enterprise language.

Let's give an example. A form and PrintTemplate template have been created for a report. The template includes named areas Title, Header, GoodsHeader and Goods. The Goods field contains a cell with the Decryption decryption parameter. The form contains the Table form element associated with the Spreadsheet form attribute of the SpreadsheetDocument type. The Decryption Processing property in the Events category refers to the cell selection processing procedure DecryptionProcessingResult() located in the report module.

To generate a spreadsheet document, a following exemplary scheme is used:

```
Spreadsheet.ShowGrid = False;
Spreadsheet.ShowHeaders = False;
Spreadsheet.Protection = True;
Spreadsheet.ReadOnly = True;

Template = Documents.GoodsConsumption.GetTemplate("PrintTemplate");
// Header
Area = Template.GetArea("Title");
Spreadsheet.Output(Area);

// Header
Header = Template.GetArea("Header");
Header.Parameters.Fill(ThisObject);
Spreadsheet.Put(Header);

// Goods
Area = Template.GetArea("GoodsHeader");
Spreadsheet.Output(Area);
GoodsArea = Template.GetArea("Goods");

For Each CurRowGoods From Goods Do

  GoodsArea = Template.GetArea("Goods");
  Spreadsheet.Put(GoodsArea);

EndDo;
```

Implementation of the decryption selection processing procedure sample:

```
Procedure DecryptionlProcessingResult(Item, Decryption,
StandardProcessing)
  StandardProcessing = False;

  DecryptionType = TypeOf(Decryption);
  If DecryptionType = Type("CatalogRef.Nomenclature") Then

    ReportFormName = "ReportForm1";

  Else

    ReportFormName = "ReportForm2";

  EndIf;

  ReportForm = DataProcessorObject.GetForm(ReportFormName);
  ReportForm.ExecuteReport(Decryption, PeriodFrom, PeriodTo);
  ReportForm.Open();

EndProcedure
```

`ReportForm1` and `ReportForm2`- are the forms developed specifically for detailed reports (usually these forms contain a spreadsheet document field where the result is output).

Decryption Use - define the area of the decryption use. When selecting a cell, decryption is possible only for the specified cell; when a string is selected, the decryption is valid in each cell of the current string. If you selected Not to use and specified the decryption, it becomes not valid: cursor does not change its appearance (for possible cursor types (see fig. 354) and the selection in the cell is not processed.

Specifying value properties

Contains Value - if this option is set, a cell contains a value. Setting this property affects the properties of other categories.

Value Type - type of cell value. For spreadsheet documents, this list contains the `Number`, `String`, `Date` and `Boolean` types. For the Spreadsheet document field form control, the list also contains types defined for the current configuration, including -documents, catalogs, enumerations and others, in addition to primitive types. The `Arbitrary` data type may be added as well. This data type may be determined using 1C:Enterprise script tools during the form filling.

Control - a control is selected for editing cell contents. The list of possible values depends on the selected content type. For example, if `Number` data type is selected, Text box or Check box may be used as an editor. A row with the editor type Text box or Check box is added to the object list (Cells, Spreadsheet Document) for the specified editor type. Contents of the properties palette for this object depends on the type of selected control.

Format - sets the format string that is to be used when displaying the value. If the display format is not configured, it is selected from the infobase regional settings.

Editing format - sets the format string that is to be be used to display the value when editing it. If the display format is not configured, it is selected from the infobase regional settings.

Format can be entered manually or using the format string editor.

NOTE. When a spreadsheet document is edited in a thin client, the following cell properties are not available: Contains Value, Value Type, Control, Editing format.

31.17.3. Spreadsheet Document in Data Input Mode

Another way to build a report is to add data into the designated cells of a a spreadsheet document placed into a form element.
During the input process, this data is processed by procedures written in 1C:Enterprise script located in the form module. Calculation results may be used to calculate values for other cells of a spreadsheet document.
After data entry, you can print a report and save it for future work.
The overall procedure for working with a spreadsheet document in the data input mode looks as follows:

- Spreadsheet document template is generated; it is prepared in a special way and used for input of data.
- A form containing the Spreadsheet document field for data input is generated.
- The Spreadsheet document form element handlers to process user-entered data (if necessary) are generated.

Let us review the steps listed above using the generation of a goods delivery form example based on the `GoodsDelivery` document. The target form is demonstrated on fig. 355. In this form, the underlined fields are to be filled in by user. Delivery date field value is automatically calculated by summing up the Issue date and Delivery period values. Issue date is the document date.

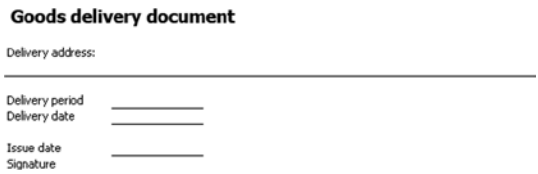


Fig. 355. Delivery Form

31.17.3.1. Spreadsheet Document Preparation

To represent a table box in the data input mode, a Spreadsheet document template must be created (with the `DeliveryForm` name). The generated template is then to be opened and its Template property is to be set to `False`.

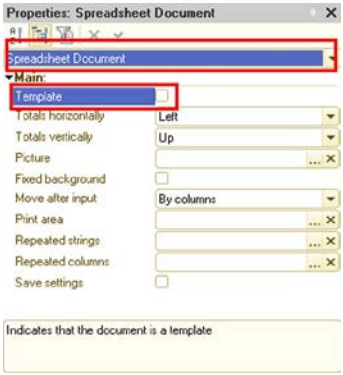


Fig. 356. Editing spreadsheet document properties

Then the required structure must be created for the document to be generated. Create the required fields:

- Delivery address. Merge several cells and define the following properties:
  - Name - `DeliveryAddress`;
  - Protection - `False`;
  - Contains Value - `True`;
  - Value Type - `String`;
  - Control - Text box.
- Delivery term. Define the following properties for the cell:
  - Name - `DeliveryTerm`;
  - Protection - `False`;
  - Contains Value - `True`;
  - Value Type - `Number`, length: `3`, precision- `0`, non-negative;
  - Control - Text box.
- DeliveryDate. Define the following properties for the cell:
  - Name - `DeliveryDate`;
  - Protection - `False`;
  - Contains Value - `True`;

- Value Type - `Date`, date content - Date;
- Control - Text box.

- Document Date. Define the following properties for the cell:
  - Name - `DocumentDate`;
  - Protection - `True`;
  - Contains Value - `True`;
  - Value Type - `Date`, date content - Date;
  - Control - Input field;
  - Format - DLF=DD.

The resulting spreadsheet document should look as shown on fig. 357

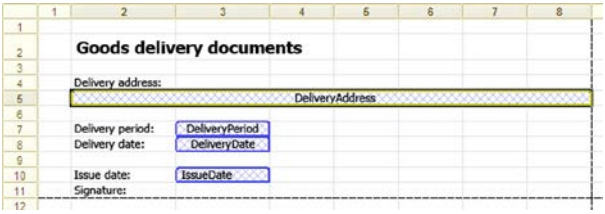


Fig. 357. Finished Template

The Show Named Cells mode is enabled for the template (select Table - Names - Show Named Cells in the menu).

NOTE. Properties that can be set for the text box in the template are neither saved nor used by the system to generate text boxes in spreadsheet documents in the data input mode.

31.17.3.2. Form Preparation for Data Input

After the spreadsheet document template has been finished, generate the form to be used for data input and implement a procedure to call it from the document. Create `ExecuteDelivery` command with the following code in the document:

```
&OnClient
Procedure ExecuteDelivery(Command)

  DeliveryParameters = New Structure("DocumentDate", Object.Date);
  OpenFormModal("Document.GoodsDelivery.Form.DeliveryExecution",
DeliveryParameters);

EndProcedure
```

The command should be placed in the form.

Now you need to create a form to be used for data input.

To do this, create a custom `DeliveryExecution` form in the `GoodsDelivery` document. Create the following form elements:

- `DocumentDate` form parameter of `Date` type;
- `DocumentDate` form parameter of `Date` type
- SpreadsheetDocument `form attribute of` SpreadsheetDocument `type` ;
- `SpreadsheetDocument` form element associated with `SpreadsheetDocument` form attribute.

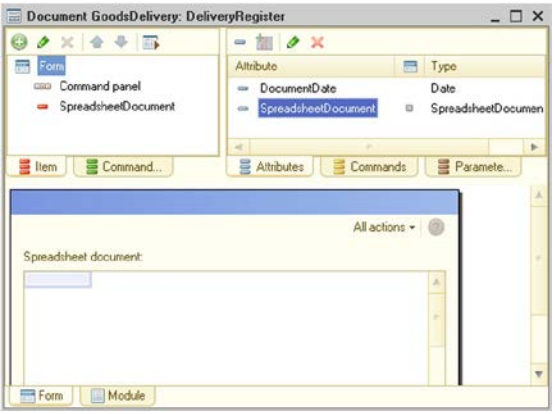


Fig. 358. Delivery Execution Form

Then implement the following handlers in the form:

```
&OnServer
The OnCreateAtServer(Denial, StandardProcessing) procedure

  // Save document formatting date
  DocumentDate = Parameters.DocumentDate;

  // Set finished template
  SpreadsheetDocument =
Documents.GoodsDelivery.GetTemplate("DeliveryForm");

  // Set document creation date in the spreadsheet document field
  SpreadsheetDocument.Area("IssueDate").Value = DocumentDate;

EndProcedure

&OnClient
Procedure OnOpen(Cancel)

  // Activate spreadsheet document area
  Items.SpreadsheetDocument.CurrentArea =
SpreadsheetDocument.Area("ShipToAddress");

EndProcedure
```

As a result, the following form filled with data is acquired:

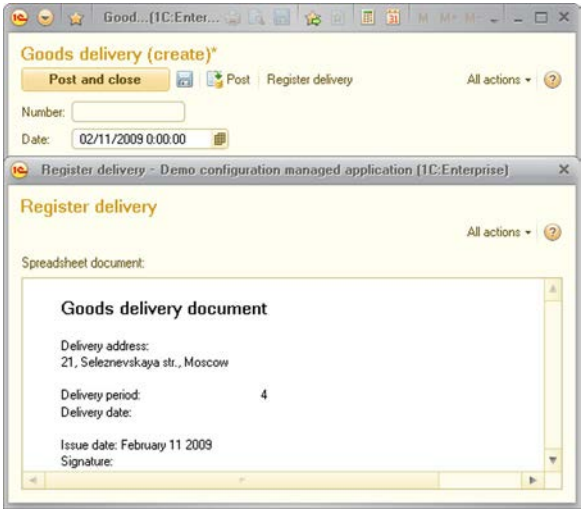


Fig. 359. Delivery Form

31.17.3.3. Implementation of a response to spreadsheet document cell changes

The acquired form has no automatic recalculation for Delivery period and Delivery date fields, if they are changed.

To make this recalculation possible, you shall implement the corresponding event processing for the form attribute associated with the spreadsheet document. You need to add the `OnChangeAreaContent` handler for the `SpreadsheetDocument` form item. The handler should contain the following program code:

```
&OnClient
Procedure SpreadsheetDocumentOnChangeAreaContent(Item, Area)

  SecondsInDay = 24 * 60 * 60;
  If Area.Name = "DeliveryDate" Then

    SpreadsheetDocument.Area("DeliveryTime").Value =
(SpreadsheetDocument.Area("DeliveryDate").Value -
BegOfDay(DocumentDate)) / SecondsInDay;

  ElseIf Area.Name = "DeliveryTime" Then

    SpreadsheetDocument.Area("DeliveryDate").Value =
BegOfDay(DocumentDate) +
SpreadsheetDocument.Area("DeliveryTime").Value * SecondsInDay;

  EndIf

EndProcedure
```

Now the spreadsheet document has obtained the desired interactivity.

31.18. Flowchart editor

Flowchart-is a schematic representation of the sequence of actions for a business process. It is a special case of the graphical schema.

The main difference between a flowchart and a graphical schema is the -special processing of special schema items (start point, end point, actions, conditions, etc.).

Flowchart is also an instruction for the application on business process actions execution sequence, an illustration for the user of this actions structure and a tool used to display the current state of a business process (description of the`GetFlowchart()` method is available in the 1C:Enterprise language help).

Flowchart is represented by a rectangular screen area that in the most general case, contains different items, for example, start point, end point, decorations, connecting lines, etc.

31.18.1. Flowchart editing

To edit the flowchart, in the business process edit window on the Other tab press the Flowchart button.
Flowchart editing is done by placing various map items on the flowchart, editing their properties and joining them using the connecting lines.
Flowchart editing is done with mouse or keyboard (some operations can only be performed with a mouse- for example, joining flowchart items using the connecting lines). When a flowchart is opened, the Inserting flowchart items toolbar buttons and the Flowchart upper level menu item become available on-screen.
Flowchart properties are configured using the properties palette.

31.18.2. Flowchart items

31.18.2.1. Design recommendations

The best approach for drawing flowcharts is using the vertical orientation of the map. The flowchart assumes vertical arrangement (top to bottom). For example, when adding the flowchart Activityitem ("element" hereinafter in this section) it is created with an outgoing connecting line directed downwards.
When creating labels, it is desirable to use identical fonts. Using a different font is an additional way of attracting attention (for example, in a decoration header highlighting a group of items).
Use the flowchart editor to place various items into flowcharts, as well as specify their size and level borders, etc.
To simplify item placing, flowchart editor provides various service tools. These include the use of a grid and performing various actions on an item group (leveling, flowchart distribution, size setting, etc).
As an additional feature, you can use the individual item design. Using the Design category features, you can customize text and background colors, text font, use pictures, tooltips and other design techniques. Properties' content depends on the item type.

31.18.2.2. Specifying graphical schema item order

Unlike the regular graphical schemas, certain types of items in the flowchart retain the order characteristic of the item type, even after the order changes. Precisely:

- Decorations are always located on the background (below) (i.e. the order of decorations can be changed only relative to each other- they will always be lower relative to other elements).
- Following the decorations (above them) are the connecting lines.
- On the very top-all other items (visualising points of the business process). This way, items constituting the logic of the flowchart are always placed above the decorations and decorative connecting lines in drawing order.

31.18.2.3. Using the connecting lines

A flowchart is used to describe business process sequences- it is defined by linking items using the Connecting line item. Connecting lines cannot be inserted just for the sake of it -they are always connected to certain points in the business process and cannot exist by themselves. Connecting lines cannot be deleted, unless this feature is provided by the point in the business process (for example, splitting point and switch point).

You can link all items used for visualization of business process points, i.e. all items except Decoration, Connecting line and Decorative line items. By default, most items used for visualization of business process points are inserted into the flowchart with one outgoing line that cannot be detached or deleted. It can only be moved to a different port (**port**-item area where a line can be connected-this is usually the middle of the side of a rectangle that contains the item; On a diagram, an unoccupied port is indicated by a blue cross).

To operate with the Connecting line flowchart item, you need to select it by clicking on any segment (interval) on the connecting line or by using the Tab (Shift + Tab) key.

If the end of the line is not connected to an item, the rectangle at the end of the line is coloured in red and the arrow has a contour, i.e., is not filled inside. To connect, use the mouse to grab the end of the connection line (gray or red rectangle) and drag it to the port area of any item, drop it there. After this, the connecting line is automatically rebuilt. You can also reconnect the connecting line (by connecting it to a different item). The following limitation is applied: you cannot directly loop one item on the other (point1 →point2 and point2 → point1), if none of these flowchart items are a Condition point or Switch item item type.

When inserting an item into the flowchart, an attempt to connect the free ports of this item to nearby non-connected lines is automatically made. When moving or resizing an item, there is an automatic attempt to connect the free ports of this item to the closest available (unconnected) lines and to the outgoing lines of this items, with the line end not connected to free ports of other items located in close proximity.

Decorative line item is used to connect decorations and flowchart points. Any number of decorative lines can be placed in the flowchart.

31.18.2.4. Flowchart correctness check

The flowchart is checked for correctness when selecting Flowchart-Check item and automatically when saving the flowchart. The following incorrect situations are checked:

- cycling;
- presence of unconnected lines;
- presence of route points that are not a starting point and have no incoming lines at the same time;
- a flowchart that does not have a starting point;
- presence of route points without a path to the end point;
- a flowchart where some lines that did not enter the merge point, left the corresponding splitting point;
- a flowchart where parallelized branches (coming out from the split point) enter the same route points (before merge point);
- a flowchart with a cycle that does not contain a single Action point, Nested Business Process Point or Processing point point type (cycle- unique cycled route point sequence);
- a flowchart where any route point of Condition Point or Switch point type has no event handler (`ConditionCheck()` or `Switch()`, correspondingly).

31.18.3. Flowchart items

This section describes all types of flowchart items that are listed in an alphabetical order.

Items behavior is configured in the properties palette by setting and selecting values. Some item properties are shared by all or most items. Individual properties of each item are given in 1C:Enterprise script help.

31.18.3.1. Common Properties of Flowchart Items

There are many properties that are shared by all or most item types. Usually these properties have a common purpose and are edited in a uniform manner. You will find a description of these properties below. For more information on editing the unique properties of items, see the sections dedicated to individual item types.

Only the properties that distinguish the flowchart items different from graphical schema items are described.

For easy description, the properties are grouped by category of the properties palette.

Some properties that are described later in this document might be unavailable for specific items.

31.18.3.1.1. Specifying general properties

Task Description - description of the task to be generated in points of the type Activities or Sub-Business Process Point.

31.18.3.1.2. Events Property Category

This category properties determine the business process points behavior for specific actions, e.g., for interactive activation, task creation, condition check (for Condition Point items), etc.
When working with items, most actions can initiate the launch of procedures related to these actions. 1C:Enterprise script describes the event handling in the body of each procedure.

Fig. 360. Creating Event Handler

To initialize the creation of a procedure responsible for an event handling, click a special button on the properties palette located to the right of the attribute with the event name in the Events properties category (see fig. 360).
Procedures are created in a business process module that feature is editable map of the route.
When working with a business process, upon the occurrence of an event, the program execution control is passed on to the procedure associated with this event.
For a list of flowchart events or items along with conditions that invoke them, see the relevant object description in the 1C:Enterprise script help.

31.18.3.2. Connecting Line

Connecting line flowchart item is used to connect flowchart items visualizing the business process points (e.g. Start Point, Activity, Condition Point and other items). Connecting line item is automatically rearranged when you change the item positions. You cannot modify the connecting line creation algorithm.
Connecting line is inserted automatically when you insert other flowchart items and cannot exist on its own. Only the Splitting Point and Switch items have the insertion of additional connecting lines feature provided.
When drawing a line, an application attempts to create the shortest path consisting of vertical and horizontal line segments that do not cross other schema items.

31.18.3.3. Activities

The Activity flowchart item represents a business process point by which tasks are generated and executed.

31.18.3.3.1. Specifying addressing properties

Explanation - a string additionally describing the activity addressing. It is used when addressing attributes are inserted by 1C:Enterprise script and are not set in advance at the design stage of the flowchart.

Group - if this property is set to `True`, the tasks for this business process point are assigned to all group (department) members. Otherwise, one task is assigned to the entire group, e.g., Sales Department and only one person (the first who undertakes it) shall execute this task.

Addressing attributes - the number of these properties is equal to the number of addressing attributes for a task selected in the Task property of the business process. The properties palette can be used to select one of the predefined values of the type specified in the addressing attribute of the task (e.g., from the Departments or Performers catalogs).

31.18.3.4. Splitting points

The Splitting Point flowchart item represents a business process point where the workflow splits into several parallel threads. By default, it is placed is a flowchart with three outgoing connecting lines.

To add an outgoing line to this item, right-click it and then click Add Outgoing Connecting Line. To delete - an outgoing line, right-click the line and click Delete. You cannot delete the only remaining outgoing line.

31.18.3.5. Condition points

The Condition Point flowchart item is a business process point that has two possible exit points based on a logical condition.

By default, Condition Point is inserted into a graphical schema with two outgoing connecting lines from the left side and from the right side. The right thread is a process branch when returning the `True` value of the `ConditionCheck()` handler. You can swap the condition branches. To do so, drag a rectangle at the beginning of the Condition Point line to the opposite side of the Condition Point item.

31.18.3.6. End points

The End Point flowchart item represents the end point of a business process. A flowchart can have several items of this type.

31.18.3.7. Start points

The Start Point item represents the business process start point. If a business process has several points of the Start Point type, you should specify the required start point when you run the process. The Start Point item cannot contain connecting lines.

31.18.3.8. Joins

The Join flowchart item represents a business process point of joining of parallel execution paths that begin at splitting point. Until the task execution for all parallel paths reaches the join point, no transition to the point following the join is executed. One join point always corresponds to one splitting point (but not vice versa, since there can be splitting points without join points). You do not have to specify explicitly which Join item corresponds to the Splitting Point item. It is determined automatically.

31.18.3.9. Sub-business process points

The SubBusinessProcess Point flowchart item represents a business process point where a business subprocess starts to execute. Execution of the main (parent) process resumes only after the subprocess is complete.

31.18.3.9.1. Specifying Data Properties

Business process - reference to a business subprocess.

31.18.3.10. Data processor points

Data Processor Point flowchart item represents a business process point that is executed in automated mode and has no destination.

31.18.3.11. Switches

Switch flowchart item represents a business process point with several exits (options), only one of which is selected based on the Option return parameter value of the `SwitchProcessing()` handler.

31.18.4. Modules

Flowcharts do not have their own modules. Event handlers are added to a business process object module.

31.19. Image Editor

31.19.1. General information

Designer offers an editing tool for images and image collections.
Pictures are stored in the Common - Common images branch configuration or in the files on the local hard drive.
To create a new image, use the Actions - Add command in the Common - Common images branch configuration and then open the created blank image for editing.
To edit an image, select it in the Common - Common images branch configuration and run the Actions - Change command in the menu; then click the Edit button in the opened dialog box. If an image is stored in a format other than PNG, the user will be prompted to convert the image to PNG format upon its opening. If you decline to convert the format, the image cannot be edited.
Images of the formats WMF, EMF and SVG cannot be edited.
To create a new image located in a file on the local drive, select File - New and select Image in the editor type selection window. Edit the blank picture and save it by selecting File - Save or File - Save as...
To edit an image stored in a file on the local hard drive, select File - Open in the menu and then select the file from the list.
The editor has two modes of operation: image editing and image collection editing. The difference between an image and image collection is conditional. Image collection is an image consisting of image items of the same size. Any image can be represented as an image collection. Use the image collection to store and edit images of similar use, like icons, button pictures, etc. The image editing mode is a default mode of the editor.
Editing techniques do not depend on the selected image type.

31.19.2. Editing images

After selecting (or creating) an image, you can edit it using the image editor.

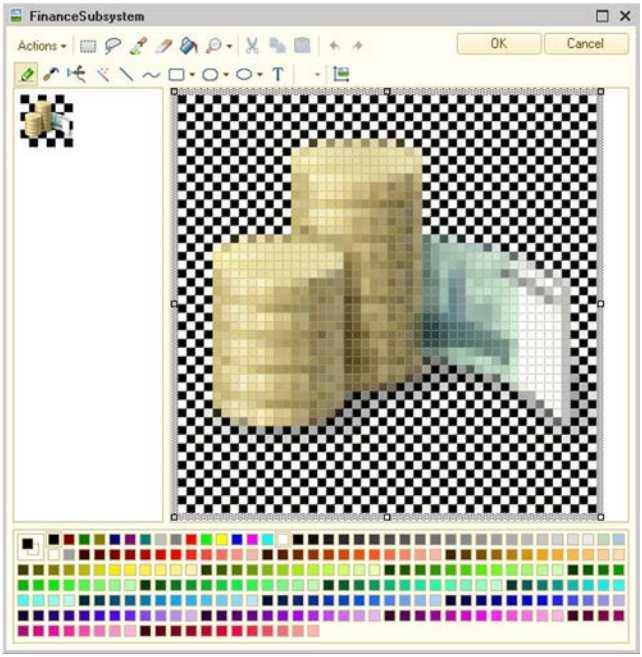


Fig. 361. Image Editor

The editor window consists of a toolbar, two fields for editing (the preview field on the left and the field for editing with an enlarged image on the right) and a color palette to select colors from.
The picture can be edited in both fields. You can perform the editing using the mouse. First you need to choose a tool or a shape to draw (pencil, brush, airbrush, line, rectangle or ellipse) and a color using the color palette.
The status bar shows the current position of cursor and the size of the area, as well as the scale.
Images represent a rectangular area consisting of points (pixels), which can be resized by dragging the bottom or right border marker, or the marker of the bottom right corner. Images can also be resized using Image Settings (see below).

Editing process comes to indicating a specific color for each pixel in the image. The number of colors is defined by the image resolution. The higher the resolution, the more colors you can use.

IMPORTANT! Using 24-bit resolution for large images results in an increase of the configuration size.

Colors in the color palette are selected separately for each mouse button. The selected color is indicated by a double frame.
The color composition can be changed. To do so, double-click the color you want to change. Select the color you need from color swatches or create a new color.
The image size can be changed. To do so, drag a marker at the side or corner of the image.
The pixel grid is designed to ease image editing. The grid is shown as a broken line. To setup the display of the grid Actions - Grid. This opens the following dialog box.

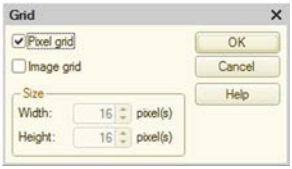


Fig. 362. Grid Settings

If Pixel grid check box is selected, the pixel grid is displayed.
If Image grid check box is selected, it is assumed that the image is a collection of images of the same size (about editing the collection (see section below). In this case, the fields for specifying the collection item size become active. Besides the pixel grid, the collection grid consisting of thin solid lines is displayed in the edit field. In this case, editing mode does not change.
If Image grid check box is selected, resizing an image is a multiple of the cell size. If this check box is cleared, the image is resized within accuracy of pixel.
Various tools and set of shapes are used for drawing. The set and use of tools are similar to the standard tools set in Paint in Microsoft Windows. Table of shortcuts available in the image editor (see Help menu).
However, there are some differences. By using the Scale button the image of the edit field can be scaled in the range from 1:1 to 20:1 with six selection levels. Clicking the button results in selection of the next scale level. If the 20:1 scale is reached, pressing the button again selects a 1:1 scale. The required scale can

be selected directly by pressing the scale selection button (with a small triangle pointed downward).
If you click the Scale button and move the mouse pointer to any of the image areas, a frame appears. This frame indicates the image area to be displayed when the user clicks on the mouse.
You can also change the scale by holding down the Ctrl key and scrolling your mouse scroll.
Shape buttons (rectangle, rounded rectangle and ellipse) also have selection buttons, by pressing which you can select the shape type (simple, outlined and filled, or filled).
Clicking the Image Settings button opens the following dialog box:

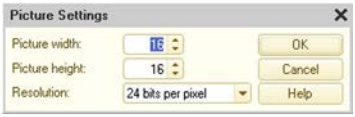


Fig. 363. Image Settings

The image dimensions and resolution (the maximum number of colors used) are set in this dialog box. Alpha-channel cannot be used for images with the resolution 1, 4 or 8 bit/pixel. Only one transparent color can be used. You can use alpha-channel if the resolution is higher than 8 bit/pixel.

NOTE. It is not recommended to use alpha-channel for the images larger than 40,000 pixels (e. g., image 200x200 pixels). The web client operating in Microsoft Internet Explorer 6.0 does not support correct display of these images. This rule does not apply to image collections with the items smaller than the specified limits.

Image editor supports standard clipboard commands. Use Ctrl + V to paste an image, Ctrl + C to copy it - and Ctrl + X to cut - (copy and delete). If an image dimensions exceed the current dimensions, the editor prompts to change it.
There is a a multi-purpose button in the toolbar that changes its purpose depending on the selected tool or shape. It is located in the second row on the right, in front of the Image Settings button. When the line, curve, rectangle or ellipse shape is selected, this button indicates thickness of the line used to draw the shape, in pixels. Pressing the button results in a consistent change of thickness (five levels). When you press the selection button, a drop-down menu opens where you can select the desired thickness of the line for drawing the shape. This button sets the size of airbrush, the size and shape of -brush and

the size of eraser -. When a selection or text tool is active, this button can be used to set the transparency mode.
Click the Text button to start entering text. The following dialog box opens:

Fig. 364. Text Input Dialog Box

Add text in the multiline text field. The font, size and other attributes are set using the Font button. The font selection is made from the list of system fonts or from the styles defined in the configuration.
Pressing the Color Replacement button replaces the color selected by the mouse pointer (indicated by a pixel) with the color set for this mouse button. Thus, you can quickly change the drawing color, using two colors at once on each mouse button.

31.19.3. Image Collections

Image collection - is an image consisting of individual items (images) of the same size. These items form a rectangular matrix, and each cell of this matrix represents a separate image.
Image collections are designed to ease the selection of necessary images for controls, column headers, etc. Using the collection guarantees the choice of images of the same size for the same type of use.
Image editor enables you to create and edit image collections. Select Collection Mode to enter the collection edit mode. Selecting this item you break the image in the preview field into cells. To edit an image item, double-click the cell - the image item is displayed in the edit field. Editing techniques are discussed in the section above.
Two buttons -Add Column and Add Row are added to the toolbar. By clicking these buttons you can add a new column or a row respectively.

Item size can be modified in the Image Settings dialog box only (resize markers are not available).

Fig. 365. Collection Settings

Image settings are set in the elements (cells) in the Image Settings items group. Cell size in pixels (dots) are set in the Collection Settings items group. Collection settings can also be configured with the relevant attributes of the manifest file of the image with options.
Image collection can also be edited as a normal image. To do so, it is recommended to pre-configure displaying the grid by images.

31.20. HTML Document Editor

31.20.1. General information

This editor offers the major features for editing HTML documents. It can also be used to edit help content for configuration objects.

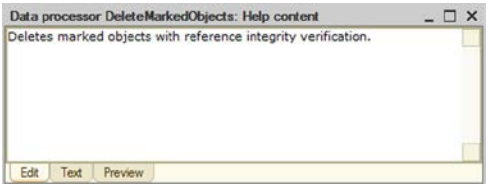


Fig. 366. HTML Document Editor

The editor consists of three tabs:

- Edit - use this tab to edit documents in visual mode.
- Text - use this tab to edit documents with HTML markup language. At that, the editor functionality matches to the text editor functionality.

- Preview - this tab can be used to preview the created document; however, editing is unavailable.

31.20.2. Visual Editing

31.20.2.1. Formatting text in HTML documents

You can format HTML document text, e.g., set its style (font, size, layout, indents, alignment, etc.).
When you copy text from other text editors, such as Microsoft Word, the original text formatting is retained.

31.20.2.2. Tables Inserting and Editing

You can add tables to HTML documents. To insert a table, click Table - Insert Table. In the dialog box, specify a number of rows and columns.
You can add more rows and columns later.
To edit a table (insert and/or delete rows, columns and cells), use the commands from the Table menu of the Designer main menu.

Menu item	Action
Insert Row	Add a new row above the current one
Delete Row	Delete the current row.
Insert Column	Add a new column on the right
Delete Column	Delete the current column.
Insert Cell	Insert a cell to the left of the current one. Shift the next cells to the right.
Delete Cell	Delete the current cell. Shift the next cells to the left.
Merge Cells	Merge the current cell and the cell to the right of the current one. The cell contents are also merged. To merge more than two cells, repeat this command as many times as required.
Split Cell	Split the current cell into two cells.

Tables can be nested. To add a nested table create a new table in a cell of the existing table.

31.20.2.3. Adding pictures to HTML documents

You can add pictures to HTML documents.
To insert a picture select the Element - Picture command. In the dialog box that opens, specify the image address (a file name and a path) or select a file in the

standard file selection dialog. In the Text field, add alternative text for the image. This text is displayed when you move the pointer over the image.
If necessary, specify image alignment (position in the document) in the Alignment attributes group.

Alignment	**Description**
Not assigned	Image is in the text
Left	Image is in a new row pinned to the left margin of the document
Right	Image is in a new row pinned to the right margin of the document
Bottom	Image is inside the text aligned to the bottom of the image
Center	Image is inside the text aligned to the center of the image
Top	Image is inside the text aligned to the top margin of the image

To add a border to the image, specify its width in the Border attribute of the dialog box. If the width is 0 or is not specified, the border is not displayed. To insert an image press OK. You can edit the image attributes in the image properties palette.
To move a picture, drag it. To do so, select a picture and drag it to the desired place of the document. The Alignment property does not changes, i. e., if the alignment is set neither to Left, nor Right, you can place the picture to a position inside the text. Otherwise the picture is placed to a new line.
When inserting pictures in SVG format, if the HTML page does not contain an item of the following kind: `<meta http-equiv="X-UA-Compatible" content="IE=*"/>`, then an item is added to the page containing the text `content="IE=Egde"`. This item disables the Microsoft Internet Explorer compatibility mode (by default, the page is rendered in compatibility mode with Microsoft Internet Explorer version 7). After adding such an element, the web browser reloads the page without compatibility mode (the current cursor position is lost).
If your computer has Microsoft Internet Explorer version lower than 9, the SVG format image will not be displayed in the HTML editor.
To copy an image, use the standard operating system tools (the clipboard or dragging while holding down Ctrl).
Picture can be positioned anywhere in the HTML-document (text, spreadsheet, or label).

31.20.2.4. Marquee

Marquee - is a special object of the HTML document. It represents the area where you can add some text, table, picture, or other labels. In the preview objects included in a marquee scroll continuously from right to left.

To insert a label, select the Element - Marquee command.

To resize a label, select it and drag its marker.

You can add labels to a text and tables.

31.20.2.5. Hyperlinks

Links are used to create a hyperlink in HTML documents.

To add a hyperlink, select a text or a picture object and click Element - Link. This opens the hyperlink properties dialog box.

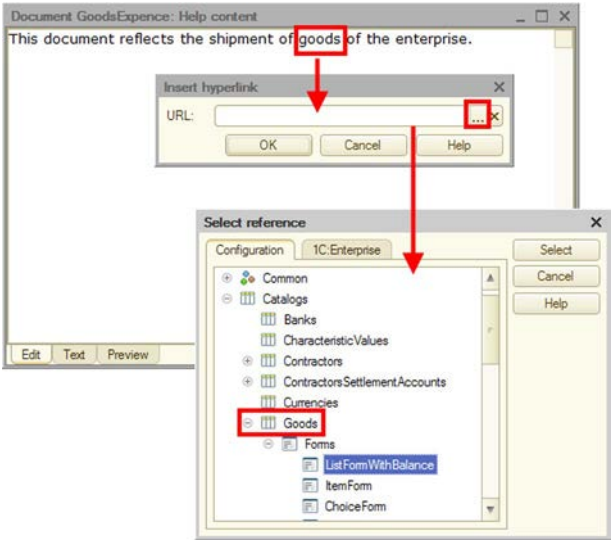


Fig. 367. Adding a hyperlink

The Address attribute should specify:

- The hyperlink address.
- Reference to help information on the configuration object.
- Reference to the system help section. If Managed application is selected as the configuration editing mode, only those help sections associated with managed application are displayed (together with the column that describes availability of this section in the web client). If Managed application and ordinary application is selected as the configuration editing mode, all

help sections are displayed with the information about the section availability in managed application, web client and ordinary application;

- Tab name (preceded by the "#" character).

To create a link click OK.
The selected object can be formatted in compliance with the style of the hyperlink. It is usually blue underlined text. Blue border is added to a picture.

31.20.2.6. Tabs

You can create bookmarks for internal references within the current HTML document. To do this, select a text or a picture object and go to Elements - Bookmark. Dialog box for entering the bookmark name appears on the screen. The formatting of the selected object is not changed.
Bookmarks are used to organize internal links.

31.20.2.7. Adding horizontal lines

Horizontal lines serve as separators.

31.20.3. Editing HTML document source code

Features of the HTML document editor listed on the Edit tab are limited. You can edit HTML document source code on the Text tab.
Please remember that manual encoding changes using the `charset` attribute of the `META` tag do not affect the editor behavior. When saving, the HTML document will be written in utf-8 encoding with the unchanged `charset` tag, and when read, the `charset` tag will be set to utf-8.

31.20.4. Previewing HTML documents

To preview the HTML document after editing, go to the Preview tab and test the document objects.

31.21. Localization of Configurations

Localization of configurations means translating the 1C:Enterprise system string values to languages specified in the Common - Languages branch of the configuration tree. String values can include names of interface items, names (synonyms) of objects, reference information, module texts, etc.
The most difficult part of this work is finding the places where the text is to be added in the required language. Therefore, it is convenient to use this mode even when only one language is set.
To start searching use the Edit - Edit interface texts command.
The following dialog box opens:

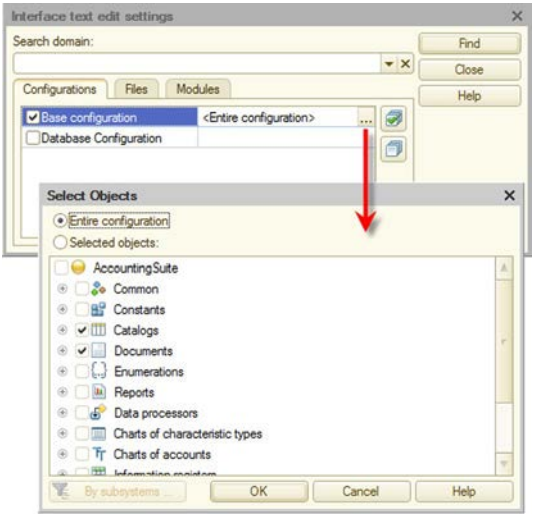


Fig. 368. Edit Interface Text

On the Configurations tab, select configuration objects for editing interface texts.
The configuration list includes all open configuration windows (in addition to the basic configuration, these can include database configuration, configuration files, repository and vendor configurations).
You can select Entire configuration line to create a full list of configuration objects containing interface properties. You can perform search only amid specific objects, if you select this object in the selection field.

Use the Files tab to select text and spreadsheet documents as well as external data processors stored in files.

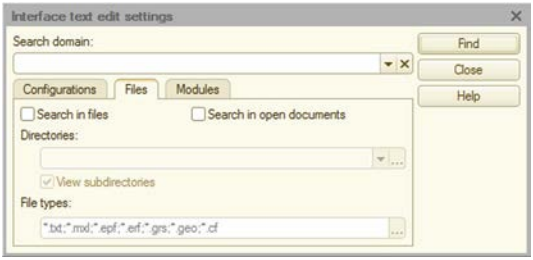


Fig. 369. Set Up Search in Files

On the Modules tab select the Search in the "NStr" functions in the modules if you want to determine where the `NStr()` operator is used in the modules. When selecting this check box, please remember that the `NStr()` operator is processed without parsing. Thus, the `NSTR()` operators in the comments will also be processed, and if the lines in different languages include line breaks, the comment mark (//) before such ("added") lines will not be added.
To remember the search parameters (list of configuration objects, files and open documents), specify a name for the current settings in the Search scope attribute. If you reopen the search window in the list of scopes, select the desired one and perform a search.
Select the objects you need and click Find. This opens the following window:

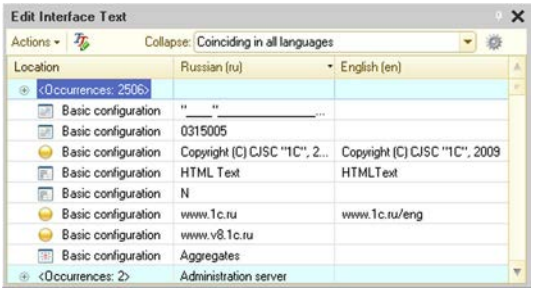


Fig. 370. Search Results Window

The window contains a table box, the first column of which shows the location of the text found. Other columns are usually used for the specified languages.

The table box can have more columns than the number of languages specified. Additional columns appear when new Language objects are created in the configuration process and then deleted, or their language code is changed (when the language code is deleted or changed, Designer does not reset the interface text for these languages).

Table box content is sorted by one of the columns. Click the column header to change the sorting. Clicking the header again changes the sorting direction.

Text can be edited directly in cells. To do this, select a cell and press Enter. Text box opens in the edit mode. Add the text you need and press Enter again. The modified text is shown in red.

For quick access to the interface text for a specific object, double-click the required line in the Location column. A form with this text is displayed. You can also review and replace the found text in the properties palette. If text has been modified outside the editing window, you can refresh the table box.

Above the table field, there are controls for performing various actions and window setup.

Use the Collapse selection field to specify the text collapse mode for different languages. If the value None is selected, the text is not collapsed. If Matching in sorting language is selected, then all items with the same text in a sorting column are collapsed. The first column contains the group icon (+) by clicking which you can expand the group. The first column will show the text <Occurrences N> (where N - is the number of item occurrences). If other columns of this group have different text values, then the cell in these columns will contain <Different values> text.

If Matching in all languages is selected, all items with the same text are collapsed.

You can perform group text replacement for collapsed rows. To do so, add the text into the appropriate language cell, as if it would be one string. The added text replaces the text at once in all the lines belonging to the group.

Use Actions - New search to open setup window to customize configuration sections to be searched.

Use Actions - Copy texts to copy all texts in one language to a different language. It is recommended to use this operation when words and phrases in different languages are basically the same.

Actions - Clear texts command clears all texts in the specified language.

Actions - Fill text command translates synonyms, headers, tooltips and interface texts in real time using a mapping file. This represents a spreadsheet document with several columns (according to the number of languages used).

First row of each column must contain a language code (for example, ru or lv). Other rows contain mapping samples. No ordering (sorting) requirements are applied. Rows can be duplicated.

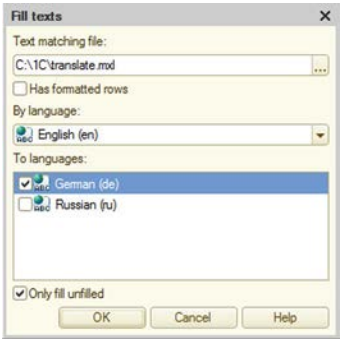


Fig. 371. Filling Texts

After selecting the mapping file, specify the default language (By language) and the languages for which translation is required (To languages). Selecting Fill blank only check box locks the overwriting of already translated strings.
Key language and languages for filling are selected from the languages that were found during interface text search. All languages selected for filling must be defined in the mapping file (it should contain columns with headers corresponding to these languages codes).

NOTE. If strings in only one language are found when you open the Edit interface text window, the Fill text command will be unavailable.

Click OK to fill the interface text. The texts are filled not only according to the key language mapping but also according to all language mappings that are present in the mapping file.
This can be used to specify a one-to-many correspondence between strings (so that identical strings are translated to the same language differently). To do so, in a two-language configuration add a third auxiliary language for commenting user interface texts and write different comments for the key language homonyms. For example, Account word may have different comments in Russian: "financial account" and "invoice". This can be used to specify correspondence between the first word and "Account" and the second word - and "Invoice".
Use the Actions - Export to spreadsheet command to export the contents of a table field to a spreadsheet document. Executing the command displays only unique strings.

Use the Actions - Settings command to set up a table field view mode and editors opening mode for group replacement.

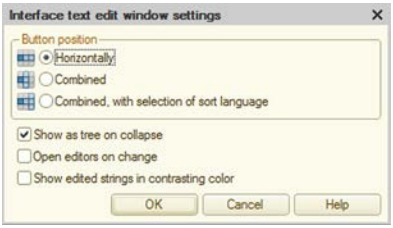


Fig. 372. Customizing Interface Text Editing Window

The Column position group is used to select a mode for columns arrangement:

- Horizontally -as shown on fig. 370.
- Combined - columns are arranged one below the other.
- Combined, with selection of sorting language - this option is used to arrange columns as follows: the column with the sorting language is to the right of the Location column, and the columns with other languages are one below the other further to the right When you click the header of a column, this column is sorted, and it is located to the right of the Location column. The column used for sorting replaces the selected column. If there are only two languages, then the columns simply change places.

If the Show as tree on collapse option is checked, you can access collapsed rows, and collapsed groups are shown as a tree.

It is recommended not to check the Open editors on group replacement option if there are many occurrences.

If Show edited strings in different color check box is selected, the modified text in the strings will be shown in a different color.

31.22. Centralized Configuration Verification

To verify a configuration, click Configuration Configuration Verification-. This opens the following window:

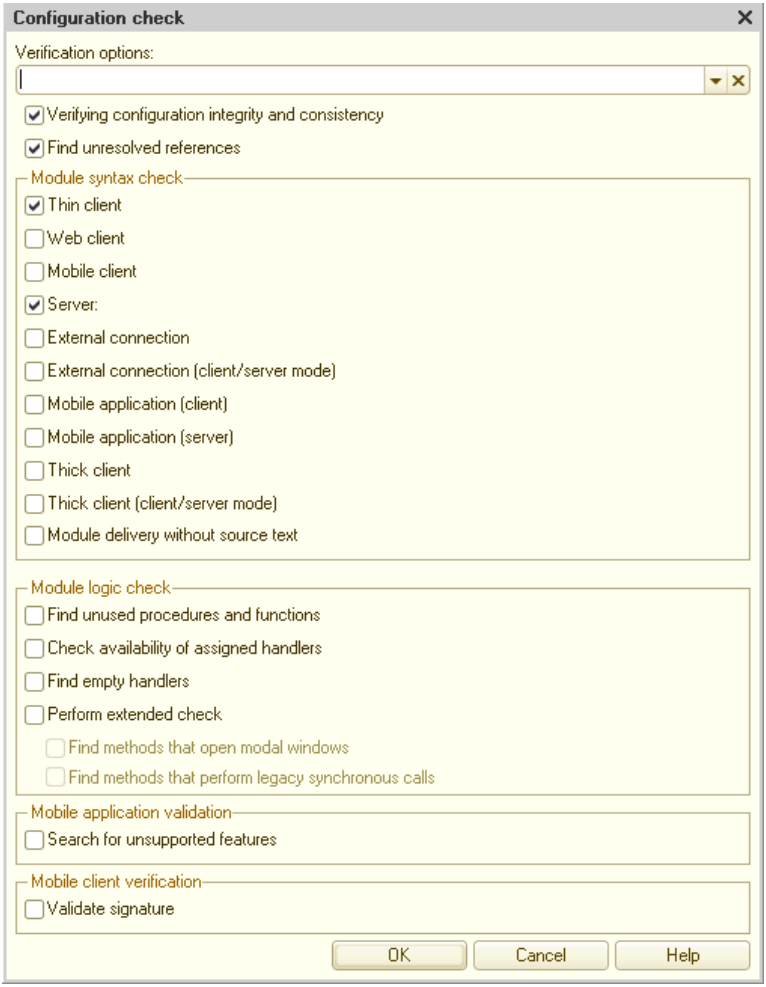


Fig. 373. Configuration check

Let's consider how configuration is verified in more detail. Any application code in 1C:Enterprise is executed in some environment. This environment is characterized by a certain set of defined preprocessor commands and a set of available objects. Respectively, when the configuration is checked, Designer defines preprocessor commands for every checked item and performs a validation. Please note, that if a client application is tested in the client/server mode, the entire application code is checked on the client side, and -only common modules available for calling from the client application are checked on the server side. If a common module exists in the application being tested and is delivered without source texts, then during the configuration check all method calls of this general module will be marked as a possible error.

Since there are quite a lot of various operating modes, validations are divided into two groups:

- A group of validations available for forced and automatic check when configuration is edited. These modes are available for selection on the tab Modules - Check of the Designer settings dialog box.
- Complete list of validations that can be called at the central configuration check (the mode is being described).

Next, it will be described which preprocessor commands are defined for every check mode, the mode features (if any), and check modes that are not related to determination of the environment of the source code being checked.

General parameters

Verifying configuration integrity and consistency.

It is the standard check (typically performed before updating the database). This check deletes incorrect data that could have been added to the configuration after editing the previous versions of the system.

Search for invalid references.

Search references to deleted objects. The search is performed in the entire configuration, including forms, templates, interfaces, and so on. It also searches for incorrect references to predefined data for charts of accounts (if the Extra Dimension Types property is set for the chart of accounts) and charts of calculation types.

Module syntax check

Thin client

Verifies the module compilation while checking the thin client environment in the managed application mode (executed in the file mode).

Preprocessor commands are defined: `Client`, `ThinClient`.

Web client

Verifies the module compilation while checking the web client environment in the managed application mode (executed in the file mode).

Preprocessor commands are defined: `Client`, `WebClient`.

Mobile client

Verifies the module compilation in the mobile client environment check mode.

Preprocessor commands are defined: `Client`, `MobileClient`.

Mobile client (offline)

Verifies the module compilation for an offline configuration. Two checks are performed with different preprocessor commands:

- First check: `MobileAppClient`.
- Second check: `MobileAppServer`.

Server

Verifies the module compilation in the 1C:Enterprise server environment check mode.

Preprocessor commands are defined: `Server`.

External connection

Verifies the module compilation in the external connection environment check mode (executed in the file mode).

Preprocessor commands are defined: `ExternalConnection`, `Server`.

External connection (client/server mode)

Verifies the module compilation in the external connection environment check mode (executed in the client/server mode).

Preprocessor commands are defined: `ExternalConnection`

Mobile application - client

Verifies the module compilation in the client application environment check mode executed on the mobile platform.

Preprocessor commands are defined: `Client`, `OnClient`, `MobileAppClient`.

Mobile application - server

Verifies the module compilation in the client application environment check mode executed on the mobile platform.

Preprocessor commands are defined: `Server`, `OnServer`, `MobileAppServer`.

Thick client (managed application)

Verifies the module compilation while checking the managed client environment (executed in the file mode).

If the Use ordinary forms in managed application configuration property is set to `False`, the ordinary form modules are not checked when you select this test.

Preprocessor commands are defined: `Client`, `ThickClientManagedApplication`, `Server`.

Thick client (managed application, client/server mode)

Verifies the module compilation while checking the managed client environment (executed in the client/server mode).

If the Use ordinary forms in managed application configuration property is set to `False`, the ordinary form modules are not checked when you select this test.

Preprocessor commands are defined: `Client`, `ThickClientManagedApplication`.

Module distribution without source text

If the configuration distribution settings for some modules specify distribution without source text, the capability to generate module images is checked.

Logical module check

Search for unused procedures and functions

Search for local (not exported) procedures and functions that are never referenced, Including unused event handlers.

The form module procedure / function is considered used if its name can be found in the form "ProcedureName" (including double quotes), i.e. if it is a parameter of any method, e.g., assignment of handler for an event.

Check availability of assigned handlers.

Checks availability of event handlers for interface, forms, controls and flowchart items.

Search for empty handlers.

Search for event handlers that perform no actions. Such handlers can impact the system performance.

Extended check.

For a restricted set of types, calls of object methods and properties are checked "over the period". Also, a check is conducted to determine whether the settings specified by a string are correct for a limited number of methods. Messages that

are formed as a result of the extended module check do not contain the line numbers with the detected problems.

Search for Modality use

This check box is only available if extended configuration verification is enabled. In this mode Designer searches in modules for methods to open modal dialog boxes. The Modality use mode configuration property does not affect the verification mechanism. If you select this check box, errors are generated even if the Modality use mode property is set to Use or Use with warning.

Search for use of synchronous calls

This check box is only available if extended configuration verification is enabled. In this mode, Designer searches in modules for code fragments in 1C:Enterprise language that use synchronous calls. The Synchronous calls to platform extensions and external components use mode configuration property does not affect the verification mechanism. If you select this check box, errors are generated even if the Modality use mode property is set to Use or Use with warning.

`Mobile application validation`

Search for unsupported features

This includes the following:

- Availability of configuration objects that are not implemented for the mobile platform
- Exchange plans with the property Distributed infobase set;
- Use of configuration objects that are not supported by the mobile platform:
  - In the Type properties of configuration objects attributes, constants and session parameters;
  - In the Command parameter type property of the `Command` object;
  - In the Type property of attributes and form attribute columns.
- Non-managed forms;
- Form items that are not supported by the mobile platform. This check is not performed for forms which Use purposes property does not include mobile devices)
- Verifying of that the start page consists of no more than one form.

`Mobile client validation`

Signature verification

Verifies digital signature of the configuration for the mobile client.

You can save the selected settings composition for further use. To do that, specify the name for the set of settings in the Verification settings field. To switch to a previously saved assortment of settings, select a name in this field.
All error messages are displayed in the message window. To interrupt the configuration check, press the Ctrl+Break key combination.

NOTE. Interrupting a configuration check can be processed with a delay at the initial stage of the check.

When the configuration connected to the repository is being checked, it is recommended to lock the configuration root object in order to avoid errors related to metadata information.
If the Edit configuration for startup modes parameter is set to Managed application and ordinary application, the following parameters are added to the dialog box:

- Thick client (ordinary application);
- Thick client (ordinary application, client/server mode).

At that, the dialog box looks like follows:

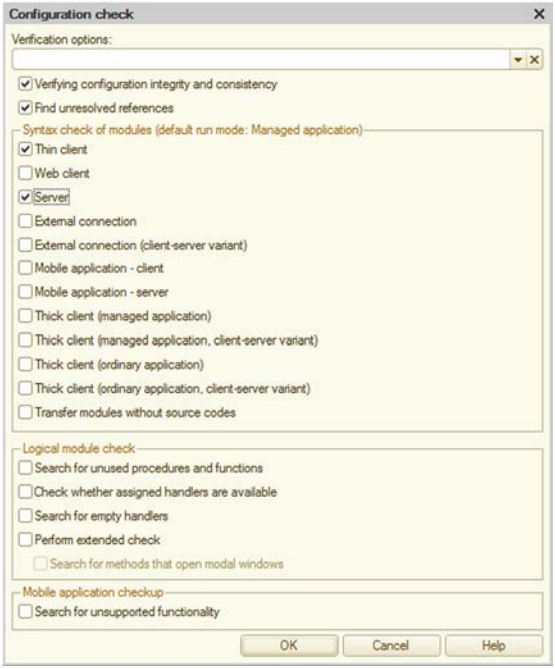


Fig. 374. Configuration check

These tests are described below.

Thick client (ordinary application)

Verifies the module compilation in the external connection environment check mode (executed in the file mode).

If the Use managed forms in ordinary application configuration property is set to `False`, the managed form modules and command modules are not checked when you select this test. The form module specified in the Default Constants Form property is checked if this test is selected regardless of the Use managed forms in ordinary application property value.

Preprocessor commands are defined: `Client`, `ThickClientOrdinaryApplication`, `Server`.

Thick client (ordinary application, client/server mode)

Verifies the module compilation while checking the client application environment, executed in the client/server mode.

If the Use managed forms in ordinary application configuration property is set to `False`, the managed form modules and command modules are not checked when you select this test. The form module specified in the Default Constants Form property is checked if this test is selected regardless of the Use managed forms in ordinary application property value.
Preprocessor commands are defined: `Client`, `ThickClientOrdinaryApplication`.

31.23. Usability analysis for solutions based on 1C:Enterprise platform

Launching a client application with the /logui command-line option creates logui.txt file in the application files folder %APPDATA%\1C\1Cv82\<Unique ID> on the user computer, and all interactive user actions (pressing keys or clicking the mouse) are registered in this file. Example:

```
"19.06.2008 16:22:49","Event FormActivate","Name
Document.GoodsReceipt.Form.ListForm","t=0"
"19.06.2008 16:22:49","Event FormActivate","Name
Document.GoodsReceipt.Form.ListForm","t=0"
"19.06.2008 16:22:49","Event Lclick","Form
AccumulationRegister.Settlements.Form.CurrentSettlemens", "Type
TableBox","t=0","beg"
```

Interactive actions are recorded for thick and thin clients and are not recorded for web clients.

Chapter 32. Debugging and testing of application solutions

32.1. General information

Debugging - is a phase in the application solution development process. It involves finding, localization and eliminating application errors.
Debugging can include using a debugger, performance measurement results, mapping server calls and simulating delays when calling the server. You can also run the automated testing scenario. In this case, **testing** refers to the process of testing a software product designed to identify situations where the application behavior does not comply with the specifications.
This chapter describes platform tools that can be used for application solution debugging and testing.

32.2. Debugger

32.2.1. General information

Debugger - is a tool that facilitates program module development and debugging in 1C:Enterprise. Debugger has the following features:

- debugging module execution in both file mode and client/server modes and background jobs;
- executing module step-by-step;
- setting breakpoints;
- interrupting and resuming module execution;
- debugging multiple modules at once;
- calculating expressions for variable status analysis;
- changing variable values during debugging process;
- viewing the call stack of procedures and functions;
- interrupting upon error;
- editing the module during debugging process.

Debugging can be performed using two different protocols: TCP/IP and HTTP. The capabilities provided by both debug protocols are fairly close.

The following text applies equally to both protocols, unless any particular features or limitations explicitly specified.
The debugger uses the notion of a debug item. **Debug item** - is the context of 1C:Enterprise language, characterized by the formal part:

- the user name, on whose behalf the code in 1C:Enterprise language is executed;
- the debug item type;
- session number used;
- The network name of the computer on which the code in 1C:Enterprise language is executed (for the TCP/IP debug protocol);
- the IP port number through which the debugger controls the operation of the debug item (for the TCP/IP debug protocol);
- the infobase name (for the HTTP debug protocol)

NOTE. If there are several workflows in the cluster in the client/server mode, and the session was reassigned to another workflow while debugging the code of the web client or server, then debugging becomes impossible.

Types of debug items include:

- **Thin client** - 1C:Enterprise script code executed in the thin client;
- **Web client** - 1C:Enterprise script code executed in the web client;
- **Thick client** - 1C:Enterprise script code executed in the thick client;
- mobile client - 1C:Enterprise script code executed in the mobile client;
- **Client (mobile application)*** - 1C:Enterprise script code executed in the client application on a mobile device;
- **Server** - 1C:Enterprise script code executed on the server;
- **Server (file mode)** - 1C:Enterprise script code executed on the server in file mode;
- **server (mobile application)*** - 1C:Enterprise script code executed on the mobile device:
  - in an application on the mobile platform, on the server or in file mode;
  - on the mobile client with offline mode and on a standalone server.
- **COM-connection** - 1C:Enterprise script code executed through an external connection;
- **Web-service** - 1C:Enterprise script code executed when calling web services methods;
- **HTTP-service** - 1C:Enterprise script code executed when calling HTTP-services methods;
- **Background job** - 1C:Enterprise script code executed in background jobs;

- **Background job (file mode)** - 1C:Enterprise script code executed in background jobs (in file mode);
- **Background job (mobile application)*** - 1C:Enterprise script code executed in background jobs (in file mode) on a mobile device;

Debug items marked by "*" symbol are used only if the HTTP debug protocol is applied.

During debugging, to indicate the system component addresses, both symbolic addresses (for example, pc-name) and IP addresses (for example, 192.168.1.1) can be used. During the operation, 1C:Enterprise subsystems are engaged in the conversion of symbolic names to IP addresses, for example, to obtain the debugger IP address, which is specified using a symbolic name. For this purpose, specialized services of the operating systems (DNS) are used. It is desirable that there are no ambiguities while resolving symbolic names.

Depending on the debug protocol used, 1C:Enterprise organizes the system components interactions in different ways. However, regardless of the protocol used, some ports of the protocol underlying the debugger operation are used. If the use of the necessary (for debugging) ports is not allowed on the computer (the ports are “closed”), then debugging is unavailable. The following may also be observed:

- increase of startup time of the client application;
- 1C:Enterprise server workflow (rphost) slows down, if the server is running in debug mode (-debug key).

The following ports are used when running the debugger:

- When using the TCP/IP debug protocol: 1560-1591 (by default). The range of ports can be changed. Ports are used on both the debugger side and the debug item side.
- When using the HTTP debug protocol: debug server port, which is explicitly specified at startup. When working in the client/server mode, the debug server uses port 1550 (by default). When working in the file mode, the port is assigned explicitly or selected by the Designer from the port range 1560-1591 (by default). The range of ports can be changed. The port is used only on the computer where the debug server is running.

When using the TCP/ IP debug protocol, the port range can be changed using the system components settings or configuration files (debugcfg.xml). Ports used (single or range) must be enabled for use (“open”) on computers that are involved in the debugging process. For the TCP/IP debug protocol, these are all the computers involved in debugging, and for the HTTP debug protocol -, this is the debug server computer.

32.2.2. Debugger Use

32.2.2.1. TCP / IP debug protocol

32.2.2.1.1. Overall scheme

To debug 1C:Enterprise script code you should run the application that executes the code in the debug mode. Debug mode requires the TCP/IP network protocol support enabled on the computer.
If 1C:Enterprise is not launched, select Debug Start Debugging- to start debugging. Designer launches the client application in debug mode.
If debug mode is enabled or debugging is set to begin upon startup in the Designer settings (you can open them by selecting Tools - Options 1C:Enterprise startup tab), you can also launch debugging in the run mode by selecting Tools - 1C:Enterprise. If you want to debug the code executed by a specific user, then in the settings form you can specify the user on whose behalf the debug mode is started.
The debugger and debug items use IP-address 127.0.0.1 when searching for each other. To ensure stable operation of the data transport service, it is necessary to determine the correspondence of the address 127.0.0.1 to symbolic name localhost. For this purpose, the following mapping can be written to the file C:\Windows\system32\drivers\etc\hosts

```
127.0.0.1 localhost
```

32.2.2.1.2. Application setup in debug mode

General information

In this section, it is assumed that the Designer is configured for debugging using the TCP/ IP debug protocol. Designer configuration setting is performed using the Designer settings dialog box. Further, the need to perform such a configuration setting is not specifically mentioned.

Debugging Client Application

To set the debug mode, you can use the following startup options:

- In the Designer mode, in the settings form (menu Tools - Options) on 1C:Enterprise startup - Additional, select Enable debugging mode check box and then attach the debug item. You can also select Begin debugging on startup; in this case connection is established automatically upon 1C:Enterprise startup. In this case, the TCP/IP protocol

should be used for debugging (the Tools - Settings dialog box, the Debug tab, Debug Protocol property must be set to TCP/IP Debugging).

- Open the infobase in the 1C:Enterprise mode by using the debug command-line option /debug -tcp (debug mode).
- If the client application is running, set the debug mode in the settings form (open via the Tools - Options menu). For this purpose, the Debug in the current session property should be set to Enabled (TCP/IP protocol). Note that you cannot change this property value after you have applied these settings.

Debugging server code

In the debug mode, the configuration objects are loaded as needed, and not at the system startup, as in the normal server operation mode. This speeds up the startup process of 1C:Enterprise when changing the configuration, that is, it accelerates the development process.
You should also consider the fact that system performance is lower in the debugging mode than in the normal mode. Therefore, it is not recommended to use the debug mode of the server for real user operations.

Server as an application

If the 1C:Enterprise server runs in an application mode (in any OS), you should stop the server and restart it to include the -debug parameter in the command line settings.

```
ragent debug <other command line settings>
```

Server as a Windows service

If the server runs as a Windows service, you need to do the following:

1. Stop the 1C:Enterprise server.

```
ragent  stop
```

2. Re-register a server agent as a Windows service to include the -debug parameter to the list of ragent command settings.

```
ragent  instsrvc  debug <other command line settings>
```

3. Start the 1C:Enterprise server.

```
ragent  start
```

Server as a Linux daemon

If you need to transfer the 1C:Enterprise server working as a Linux daemon into the debug mode, you need to do the following:

1. Stop the 1C:Enterprise server.

```
/etc/init.d/srv1cv83 stop
```

2. In the srv1cv83 configuration file set the value of SRV1CV8_DEBUG parameter to 1.

```
SRV1CV8_DEBUG=1
```

3. Save the configuration file.
4. Start the 1C:Enterprise server.

```
/etc/init.d/srv1cv83 start
```

Debugging External Connection

To enable debug mode for an external connection, use the settings located in xml file comcntrcfg.xml which is located in the 1C:Enterprise system configuration files directory. If the file cannot be found, the application runs in the standard mode.

Sample of comcntrcfg.xml file:

```
<config xmlns="http://v8.1c.ru/v8/comcntrcfg">
  <debugconfig debug="true" protocol="tcp"
debuggerURL="tcp://localhost:1560"/>
</config>
```

If the debuggerURL attribute is set for the debug item, then Designer should be available at the specified URL when accessing the external connection. Otherwise, the execution of the external connection will be suspended and the system will wait for the Designer availability. At the same time, to continue working in the Designer mode, you should attach the required debug item.

Debugging web service, HTTP service and access via the standard OData interface

To indicate that the service needs to be started in debug mode (only for the file mode of the infobase), use the settings that are located in the default.vrd file, which should be located in the virtual application directory. You need to

specify the debug item in this file, it's absence would mean the inability to debug the Web service.

Sample of the debug item from default.vrd file:

```
<debug enable="true" protocol="tcp" url="tcp://localhost"/>
```

If the url attribute is set for the debug item, then Designer should be available at the specified URL when accessing the service. Otherwise, the service will be suspended and the system will wait for the Designer availability. At the same time, to continue working in the Designer mode, you should attach the required debug item.
If you need to debug a service in the client/server mode of an infobase, you should enable debug mode on the server using the debug command-line option.

NOTE. To debug an HTTP service and to operate with standard OData interface, it is not recommended to use the Microsoft Internet Explorer 9.0 web browser, because there are URL coding errors in this web browser.

Debugging Web Client

To set the debug mode, you can use the following startup options:

- In the Designer mode, on the Tools menu -, click Options, click 1C:Enterprise startup tab -, click Additional tab, and select Enable debugging mode check box. Then attach the debug item. You can select also Begin debugging on startup check box. In this case connection will be established automatically upon 1C:Enterprise startup.
- Open the infobase in the 1C:Enterprise mode by using the debug command-line option (debug mode):

```
http://localhost/demo?debug=tcp
```

- Open the infobase in the 1C:Enterprise mode by using the debug command-line options debug and debuggerurl="IP-address":

```
http://localhost/demo?debug=tcp&debuggerurl="127.0.0.1"
```

- IP address - the debugger address. If the debugger located at 127.0.0.1 is launched, it will automatically attach the debug items (the debugger must be run at the specified address).

To enable the debugging of the server file mode of the infobase, you can use two methods:

- Start debugging directly from the Designer as described above. In this case, debugging of the server file mode on the computer with the web client starts automatically.
- Enable debug mode using default.vrd file, that should be located in the virtual application directory. You need to specify the debug item in this file, it's absence would mean the inability to debug the Web service.

Sample of the debug item from default.vrd file:

```
<debug enable="true" protocol="tcp" url="tcp://192.168.0.30"/>
```

You should also consider the fact that system performance is lower in the debugging mode than in the normal mode. So it is not recommended to use the debugging mode of the server file mode for real user operations.

When debugging a web client, you need to remember the following features:

- the feature to interactively enable the debugging mode is not supported;
- automatic search of debugging objects on remote computers is not supported when debugging a web client for the file mode infobase;
- Designer activation from the web client is not supported during debugging;
- debugging external reports and processes is not supported;
- in case of forced termination of the web client from the debugger, the process of the web browser that started when the web client was started from Designer is terminated. However, if the web client starts in the tab of an already running web browser, then the forced closing of the tab does not occur and a dialog box opens in the debug web client window with information indicating that the web client is terminated at the request of debugger.

IMPORTANT! To debug a web client (including client program code) in the client/server mode, the 1C:Enterprise server should be launched in the debug mode (debug key).

NOTE. It is not recommended to use web client debug in low-speed communication channels.

Debugging external data processor (report)

To debug an external report (processing), two conditions must be met:

1. External data processing (report) is opened in client application from a specific file mode location;

2. Designer that operates the debug item with debugged external data processing (report), has access to the external data processing (report) file that is opened by the client application.

If this conditions are not met - debugging of external data processing (report) will be impossible. In particular, debugging of external data processing (reports) located in the infobase are not supported. In this case, it is recommended to extract the external data processing (report) from the infobase for the time of debugging.

32.2.2.1.3. Attach Debug Items

To debug a module, the debug item should be attached. To manage attachment select the Debug - Attachment item. The debug item selection dialog will be displayed. The list includes only those debug items for which the following conditions are met:

- Debugger and the debug item have the same infobase ID:
  - For the file mode of infobase, it is generated at the first access to the database using the client application (including the Designer access). At the next first access the ID will be re-generated. The “first access” is not an absolute first access at the moment of creating the infobase, but the first call after the infobase has no more connections left.
  - For the infobase in the client/server mode, the infobase ID in 1C:Enterprise server cluster is used.
- The application must have the debug option enabled (for the server -, debug check box, for the client application -, the corresponding command line parameter, or the property of the client application settings dialog box, or a correctly configured configuration file).

Debug items are available for attachment throughout the time interval of the session, within which a specific background job, Web service or external connection are running. This interval can be very short, and in this case you can configure the debug items automatic connection.

NOTE. When a debug item is attached, the infobase users list should include users which can be authenticated both in the Designer and in the debug item.

Usually the list contains one line that refers to the configuration running in the 1C:Enterprise mode. If several 1C:Enterprise applications are running with this configuration, the list may contain several lines.

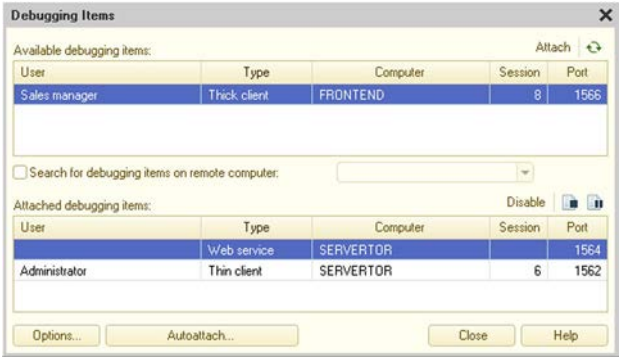


Fig. 375. Attach Debug Items

If Search for debug items on remote computer check box is selected, enter computer name or network address in the box to the right of the check box or select it from the list of previously entered names. The debug items found on the remote computer will be added to the available debug items list. The list of attached debug items contains the items already attached to the debugger.

Pressing the Attach button attaches the selected debug item to the debbuger. In the attachment window, this is displayed by the transfer of the debug item from the list of available debug items to the list of attached debug items.

To exclude a debug item, you need to specify it in the list of attached items and press the Detach button. For a detached debug item, debugging is no longer performed. In particular, this means that the breakpoints set in detached debug items do not function during walkthrough of them. In the attachment window, this is displayed by moving the debug item from the list of attached items to the list of available items. You can re-attach to it.

To close the debug item, press the Finish button, to stop at the execution point - press the Stop button.

To open range setup dialog box press Options. Specify the port range to search for debug items. The search is performed on the current computer or on the specified computer.

Fig. 376. Specifying debugger settings

The Debugger field of the dialog box contains the current debugger settings. You can use them, for example, in a command line during the client application startup as the /debuggerurl command-line option parameter or in an XML file containing debug settings for external connection or Web-service. To set auto-attachment for debug items on the 1C:Enterprise server running in the debug mode, use the Automatic attachment dialog box and select the required debug item types.

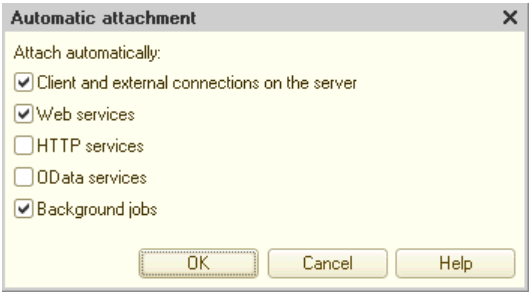


Fig. 377. Auto-connection Settings

The specified settings for automatic connection are valid only for the client/server infobase, except for Background jobs, which can be used to automatically connect background jobs, both in the client/server and in the file modes of infobase. If it is necessary to debug various services (WEB-, HTTP- and OData-) in the file mode of infobase, then debugging should be configured in the publishing dialog box on the web server.

Auto-connection of a background job run by the client application or a web server extension is indirectly managed in this dialog box. Background jobs can only be debugged if certain conditions are met:

- For thin or thick client:
  - Allow debugging in client application;
  - Allow auto-connection to background jobs in the auto-connection settings dialog box.
- For web server:
  - Allow debugging for infobase in the publication file (default.vrd);
  - Allow auto-connection to background jobs in the auto-connection settings dialog box.

32.2.2.1.4. Additional Port Range Settings

If all ports for connection within the standard range are occupied, you can specify additional range. This range can be specified in debugcfg.xml file located in the configuration files directory, on the computer where the Designer and debugged client application are located, or where an external connection is established. If a port range is changed on the computer with the debugged application, these ports should be set for the Designer where debugging takes place (if the debugged application and the Designer are located on different computers). If the file is not found, then ports from the standard range (1560-1591) are used for communication. Debugging items on the server use the same ports as the server processes: rmngr and rphost. Additional port ranges for debug items on the server are not required.

Example:

```
<config xmlns="http://v8.1c.ru/v8/debugcfg">
  <debugports range="1540:1550"/>
</config>
```

32.2.2.2. HTTP debug protocol

32.2.2.2.1. Overall scheme

To debug 1C:Enterprise script code you should run the application that executes the code in the debug mode. Debug mode requires TCP/IP network protocol support enabled on the computer and HTTP protocol operation is not blocked (various proxy servers, firewalls, etc.).
When using the HTTP debug protocol, a special **debug server** (dbgs) is used for debugging. Debug server acts as the debugging process coordinator:

- Stores lists of debug items available for use and lists of attached debug items.
- Translates debug control commands to debug items commands (stop, continue execution, etc.).

There is no direct information transfer between the debugger and debug items. All interaction is organized through the debug server. A message queue is organized on the debug server, through which the debugger and debug items transmit information to each other.

Both the debugger and debug items interact with the debug server using the HTTP protocol. The debug item location does not matter for the successful operation of the mechanism.

Interaction with the debug server is initiated by the debugger and debug items. During operation, two connections are established between the debug server and each process participant (debug items and debugger). One connection is used by the debug item to report its status to debug server (a breakpoint is reached, step is completed, etc.). Another connection is used by the debug item (or debugger) to find out whether the information has appeared for it on the debug server. And if it is there, -to receive this information.

Thus, there is a one-way interaction. Information is transmitted all the time from the debug server to the debugger and to the debug items. It should be understood that if the debug item is not running in debug mode, -the debug server will not be able to establish a connection with it. In particular, this means that it is impossible to debug the code on the 1C:Enterprise server if the server is not running in debug mode.

The debug server can handle debugging of an arbitrary number of debug items from different infobases. Debugger and debug items on the debug server is compared on the basis of the infobase name. These names are different and depend on the use case of the infobase:

- Client/server mode - the infobase name is used, as defined in the 1C:Enterprise server cluster.
- The file mode - the predefined name DefAlias is used (the possibility of change this name is not supported). Thus, in the case of a file infobase, the debug server should be used for simultaneous work only with the debugger and debug items from one file infobase. Upon the file infobase Designer startup, the debug server for the used infobase automatically starts, too. When you start the client application from the Designer, it is passed the address of the debug server running for this infobase. Thus, when two Designers work with two file infobases, two debug servers will be used, each for its own infobase. At the same time, when launching a client application not from the Designer, you must

manually specify the correct debug server address in the command line of the client application (if debugging is needed).

Designer provides the ability to customize the infobase name which the debugger uses when registering on the debug server. This feature is used in cases when one infobase is open in the Designer, but debugging is required for debug items from another server infobase with a known name on a known debug server.

Let us consider what happens when a breakpoint is encountered when using a debug server. After the debug server received a message stating that a particular debug item stopped at a breakpoint, it sends a message stating that all connected debug items should stop. Each debug item executes this command before starting the execution of the next “own” code line in 1C:Enterprise language. Connected debug items stop asynchronously: debug items of the current session (these are the client application and the server part servicing this client application) stop synchronously, and the moment of stopping the remaining debug items is generally impossible to determine. After the command is executed in the debugger, for example, take a step, the debug server informs all attached debug items that they can continue to perform their work. At the same time, the specified command is executed in the active debug item, and the “take a step” operation can be represented (with some approximation) as launching the debug item with setting a breakpoint on the next line. Once the active debug item has completed the requested operation, it reports this to the debug server and the process repeats.

Two important facts follow from this description:

1. It is impossible to simultaneously debug various debug items related to the same infobase (client/server mode) or the same debug server (file mode).
2. With a single step in the current debug item, it is impossible to predict (in general case) where the stop will occur in the other attached debug items. As part of this statement, it could be considered that the client and server debug items related to the same session number act as one debug item.

Also worth mentioning is the ability to set a password for access to the debug server. The password is set when starting the debug server and must be specified by the debugger when accessing such a debug server.

32.2.2.2.2. Application setup in debug mode

General information

In this section, it is assumed that the Designer is configured for debugging using the TCP/ IP debug protocol. Designer configuration setting is performed using the Designer settings dialog box. Further, the need to perform such a configuration setting is not specifically mentioned.

Debugging when starting from Designer

In order for the client application to start up from the Designer immediately in debug mode, you must start the client application using the Debug - Start Debugging command. In this case, the client application will be launched in debug mode, and all necessary components will also be launched with the correct settings or command line settings. This will start the client application that is selected in the Designer settings.
If you need to start a specific application in debug mode without changing the default settings, you can start it using the Debug - Start Debugging command. The menu lists client applications that can be launched in debug mode.
Starting a mobile application debug from Designer is supported only for devices running Android OS. In this case, the device must be attached to the personal computer that is running Designer using a USB cable and in the start settings of the mobile application the use of Android Debug Bridge to start the mobile application must be configured. Debugging a mobile application on an iOS device is only possible when the application is launched manually on the mobile device.

Enable debugging in a running client application

In case you need to debug an already running client application without debug mode enabled, you should perform the following steps:

1. Open a dialog box intended to set up client application parameters (Parameters in Service and - Settings).
2. The Property Allow debugging in the current session should be set to Enabled (HTTP protocol).
3. In the Debug Server property, specify the address of the debug server being used.
4. Click OK. After you enable debugging in the current session, you cannot disable it or change the debug protocol.

Enable debugging in a running client application is not supported when using a web client.

Debugging server code

In the debug mode, the configuration objects are loaded as needed, and not at the system startup, as in the normal server operation mode. This speeds up the startup process of 1C:Enterprise when changing the configuration, that is, it accelerates the development process.
You should also consider the fact that system performance is lower in the debugging mode than in the normal mode. Therefore, it is not recommended to use the debug mode of the server for real user operations.

Server as an application

If the 1C:Enterprise server runs in an application mode (in any OS), you should stop the server and restart it to include the -debug parameter in the command line settings.

```
ragent / debug http <other command line settings>
```

If necessary, you should also specify the values of the /debugServerAddr,/debugServerPort, and /debugServerPwd parameters in the server startup command line.

Server as a Windows service

If the server runs as a Windows service, you need to do the following:

1. Stop the 1C:Enterprise server.

```
ragent /stop
```

2. Re-register a server agent as the Windows service to include the -debug parameter to the list of ragent command settings.

```
ragent /instsrvc /debug http <other command line settings>
```

   If necessary, you should also specify the values of the /debugServerAddr, /debugServerPort, and /debugServerPwd parameters in the server registration command line.
3. Start the 1C:Enterprise server.

```
ragent /start
```

Server as a Linux daemon

If the 1C:Enterprise server runs in an application mode (in any OS), you should stop the server and restart it to include the -debug parameter in the command line settings.

```
ragent /daemon /debug http <other command line settings>
```

If necessary, you should also specify the values of the /debugServerAddr,/debugServerPort, and /debugServerPwd parameters in the server startup command line.

IMPORTANT! If Linux OS is configured to automatically launch the 1C:Enterprise server cluster at the OS startup, then the use of the HTTP debug protocol is not supported.

Debugging External Connection

To enable debug mode for an external connection, use the settings located in xml file comcntrcfg.xml which is located in the 1C:Enterprise system configuration files directory. If the file cannot be found, the application runs in the standard mode.

Sample of comcntrcfg.xml file:

```
<config xmlns="http://v8.1c.ru/v8/comcntrcfg">
  <debugconfig debug="true" protocol="http" debuggerURL="http://pc
name:1561"/>
</config>
```

If the debuggerURL attribute is specified for the debug item, then the debug server should be available at the specified address when the external connection is used. Otherwise, the external connection will work in ordinary mode and it will not be possible to debug it.

Debugging web service, HTTP service and access via the standard OData interface

To indicate that the service needs to be started in debug mode (only for the file mode of the infobase), use the settings that are located in the default.vrd file, which should be located in the virtual application directory. You need to specify the debug item in this file, it's absence would mean the inability to debug the Web service.

Sample of the debug item from default.vrd file:

```
<debug enable="true" protocol="http" url="http://pc name:1561"/>
```

If the debuggerURL attribute is specified for the debug item, then the debug server should be available at the specified address when using the service. If at the moment of launching the application, the debug server will be unavailable, debugging will be impossible, and the platform will work as usual.

If you need to debug a service in the client/server mode of an infobase, you should enable the debugging mode on the server using the /debug command-line option.

NOTE. To debug an HTTP service and to operate with standard OData interface, it is not recommended to use the Microsoft Internet Explorer 9.0 web browser, because there are URL coding errors in this web browser.

Debugging Web Client

If the web client is started not from Designer, then to debug the client, the following actions should be performed:

1. Determine the address of the debug server with which the debugger will interact.
2. Open the infobase in 1C:Enterprise mode, indicating the need for debugging:

```
http://localhost/demo?debug=http,attach&debuggerURL="addr"
```

   addr value is acquired on the previous step.
3. In this case, the debugging of the server application code will be unavailable.

To enable the debugging of the server file mode of the infobase, you can use two methods:

- Start debugging straight from Designer.
- Enable debug mode using default.vrd file, that should be located in the virtual application directory. You need to specify the debug item in this file, it's absence would mean the inability to debug the Web service.

Sample of the debug item from default.vrd file:

```
<debug enable="true" protocol="http" url="http://pc name:1561"/>
```

You should also consider the fact that system performance is lower in the debugging mode than in the normal mode. So it is not recommended to use the debugging mode of the server file mode for real user operations.

When debugging a web client, you need to remember the following features:

- the feature to interactively enable the debugging mode is not supported;
- automatic search of debugging objects on remote computers is not supported when debugging a web client for the file mode infobase;
- Designer activation from the web client is not supported during debugging;

- debugging external reports and processes is not supported;
- in case of forced termination of the web client from the debugger, the process of the web browser that started when the web client was started from Designer is terminated. However, if the web client starts in the tab of an already running web browser, then the forced closing of the tab does not occur and a dialog box opens in the debug web client window with information indicating that the web client is terminated at the request of debugger.

IMPORTANT! To debug a web client (including client program code) in the client/server mode, the 1C:Enterprise server should be launched in the debug mode (debug key).
NOTE. It is not recommended to use web client debug in low-speed communication channels.

Debugging external data processor (report)

To debug an external report (processing), two conditions must be met:

1. External data processing (report) is opened in client application from a specific file mode location;
2. Designer that operates the debug item with debugged external data processing (report), has access to the external data processing (report) file that is opened by the client application.

If this conditions are not met - debugging of external data processing (report) will be impossible. In particular, debugging of external data processing (reports) located in the infobase are not supported. In this case, it is recommended to extract the external data processing (report) from the infobase for the time of debugging.

Debugging mobile application

Debugging when starting from Designer

To start debugging from Designer, only the developer platform can be used. Starting a mobile application debug from Designer is supported only for devices running Android OS. In this case, the device must be attached to the personal computer that is running Designer using a USB cable and in the start settings of the mobile application the use of Android Debug Bridge to start the mobile application must be configured. In this case, the address of Designer, where the debugging starts from, is used as the debug server address. Debugging permission flag and the debugger address specified in the infobase property of the same name are ignored.

Debugging when publishing on a web server

To start debugging from Designer, only the developer platform can be used. The first start of the mobile application is performed on the mobile device manually. When started for the first time, the mobile application uses the properties of the infobase to determine the status of debugging. If the Restart from Designer property is set in the infobase settings, then the debug state upon restart will be determined by how this restart was initiated in Designer. The mobile application will be restarted in debug mode if the restart is initiated from Designer using the Start Debug command. In other cases, the restarted mobile application will not be available for debugging.

IMPORTANT! Selecting the Restart from Designer property in the infobase settings actually allows debugging of the mobile application. This feature should be used carefully, when debugging mobile applications with real confidential data in order to avoid unauthorized access to such data.

Debugging permission flag and the debugger address specified in the infobase property of the same name are ignored (except for the first start of mobile application).

Debugging assembled mobile application

To start debugging from Designer, only the assembled mobile application can be used.
The mobile application can be started only on the mobile device itself. Debug status is determined by the settings of the used infobase.

Setting-up debugging permission and debug server address

To configure the debugging control settings, open the list of infobases, find the required base in the list and perform the following actions in the properties of this infobase:

- Enable the Debug enabled radio button.
- Set the debug server address in the Debug address field. The address of the debug server to be used should be retrieved in Designer, in the Debug server property of the Debugger settings dialog box, that is available when executing the following command Main menu - Debug - Attachment -Settings.

You should keep in mind that the debug settings specified in the infobase properties can be ignored in some operation modes (described above in this section).

Debugging offline mobile application

No debugging of all components of an offline mobile application can be performed. You can either debug those running on the mobile device or interacting with the main server. Whenever offline application components are debugged, make sure that the following debug items are enabled: client (mobile application) and server (mobile application). Whenever primary application components are debugged, make sure that the following debug items are enabled: mobile client and server.
To switch in between these modes, use an infobase setting parameter on the mobile device.
Actions performed before debugging is started are similar to those typical of mobile application or application debugging on a personal computer.

32.2.2.2.3. Attach Debug Items

To debug a module, the debug item should be attached. To manage attachment select the Debug - Attachment item. The debug item selection dialog will be displayed. The Available debug items list contains debug items registered on the debug server used for the infobase specified in the settings of this debugger (Designer).

The following is excluded from the Available debug items list:

1. Attached debug items (displayed in the attached debug items list);
2. The debug items that do not match the filter set at the top right part of the window. Filter is intended only for the visual limitation of the list. Debug items that do not match the filter are still available for use and, after turning off the filter, will instantly appear in the available debug items list.

Pressing the Attach button attaches the selected debug item to the debbuger. In the attachment window, this is displayed by the transfer of the debug item from the list of available debug items to the list of attached debug items.
To exclude a debug item, you need to specify it in the list of attached items and press the Detach button. In the attachment window, this is displayed by moving the debug item from the list of attached items to the list of available items. You can re-attach to it. Moreover, breakpoints set in detached debug items will not “work” when passing through them.
To close the debug item, press the Finish button, to stop at the execution point - press the Stop button.

To open a dialog box with information about the debug server being used, you should press the Settings button. In the opened dialog box, the debug server address and the name of the infobase will be displayed.

To automatically attach debug items to the current debugger, you can use the Auto attach dialog box and configure the required filters in it.

On the left part of the dialog box (Types of debug items), you can specify which types of debug items will automatically attach to the current debugger straight after registration on the used debug server. You can select several types of debug items. If the type of debug item is marked as the Background Job, this means that as soon as the 1C:Enterprise server, which is running in debug mode, starts the background job, it will automatically be attached to the current debugger.
On the right part of the dialog box, you can specify an additional filter of debug items (Filters of debug items). The additional filter of debug items, in addition to specifying types of debug items, also includes the filter by users, as well as by the values of separators.

The debug item will be automatically attached to the debugger if it matches one of the enabled filters: this debug item matches the filter by item type, operates on the behalf of the specified (or one of the specified) users and the separators are set to specified values.
The filter that is configured in the Automatic attachment of Debug Items on the debug server dialog box, combines a filter by debug type and by debug item filters "by OR". I.e the debug item will be automatically attached if it matches one of the selected debug types OR matches one of the enabled filters.
The values set in this dialog box start working immediately and until the next change of settings.
A common list is used to filter available debug items. A list of debug item filters can be exported to a file and imported from a file.

32.2.2.2.4. Debug server

General information

Debug server (dbgs) - is an application that acts as an intermediary between the debugger and various debug items.
The debug server can be started in several ways:

- From Designer, when debugging an application in file mode.
- From server cluster, if debugging mode is enabled for the cluster.
- From application developer, if it is necessary to implement complex debugging scenarios. In this case, you can also register the debug server as an operating system service (for Windows OS) or start it as a "demon" (for Linux OS).

In case the debug server is started by other components of the 1C:Enterprise, all the necessary settings of the debug server are automatically configured by these components.
In cases when you need to manually start the debug server, you should use the command line, where the following settings can be specified:

```
dbgs --version|--help
  --service|--daemon
  --addr=<host>
  --port=<port>|--range=<range>
  --ownerPID=<PID>
  --password=<pwd>
  --notify=<file>
```

--version or -v

Displays the debug server version.

--help or -h or -?

Displays brief help information about debug server startup command line settings.

--service only for Windows OS

Registers the debug server as a Windows OS service.
--service key is incompatible with the --daemon key.

--daemon only for Linux OS

This key allows you to start the debug server in the "demon" mode of Linux OS, i.e in the background application mode that does not interact with the terminal from where this application is running. Starting a debug server with this key does not mean that the server agent will be automatically started for execution after the system is rebooted.

--daemon key is incompatible with the --service key.

NOTE. When starting the debug server as a service or “demon”, you should keep in mind that through such debug server you can debug infobase debug items in file mode and client/server infobase client debug items. Debugging server debug items of client/server infobases via such debug server is not supported.

--addr=<host> or -a <host>

Allows to specify the IP address that will be used as the debug server address. It makes sense if there are several network cards with different IP addresses installed on the computer. If the IP address is not specified, then the address of an arbitrary network card will be used.

--port=<port> or -p <port>

The port that will be used to connect to the debug server. If the specified port will be unavailable, the debug server will not start and debugging will be impossible. The port must be specified explicitly, there is no default value. When you start the debug server using the server cluster, the port 1550 will be used, which cluster selects by default and can be redefined using the server cluster start-up keys.

If the --port and --range parameters are specified at the same time, the --port parameter will be ignored.

--range=<range> or -r <range>

Specifies a range of ports to select the port from, which will connect to the debug server. If all the ports specified in this parameter are unavailable, the debug server can not start and debugging is impossible.

A range of ports is indicated by the symbol ":", several ranges (or specific ports) are separated by the symbol ",". Example: 2560:2590, 4567, 9000:12000. In this example, the ports from 2560 to 2590, port 4567 and ports from 9000 to 12000 are used to select the port. As a result, one port is selected, which will be used in the work. Information about the selected port (along with the address of the debug server) will be saved to the file specified by the --notify parameter.

If the --port and --range parameters are specified at the same time, the --port parameter will be ignored.

--ownerPID=<PID> or -opid <PID>

This parameter specifies the ID of the operating system process, the lifetime of which determines the lifetime of the debug server process. The debug server monitors the process with the specified id. After completion of this

process it also completes work. In this case, all debugging processes that are performed through this debug server will also be terminated.
When debugging the file infobase, Designer automatically sets its own process ID for this parameter, and when running from a server cluster, the cluster manager process ID is set in this parameter (rmngr).

--password=<pwd> or -pwd <pwd>

Password to access the debug server. The password is to be specified for all debuggers that want to be debugged using this debug server.

--notify=<file>

The full name of the file where debug server stores its address. Designer or cluster manager reads the information on the debug server address and deletes the file.

Getting the debugger address

To retrieve the address of debug server for interacting with the Designer debugger, you need to perform the following steps:

1. Open the dialog for attaching debug items Main menu - Debug - Attach ...;
2. Click Settings...;
3. Copy information from the Debug Server: line.

Test the debug server performance

The debug server performance can be tested as follows:

1. Retrieve the address of the debug server, e. g. in the Designer.
2. Enter the obtained address in the address bar of any web browser.
3. The debug server is functioning normally if the web browser displays the following information:

```
1C:Enterprise 8.3 Debug Server (8.3.7.1140) © '1C' 1996 2015 - it
works!
```

32.2.3. Breakpoint

A breakpoint - is a place in the program module where module execution stops, and control passes to the debugger.

The current breakpoint, break locations and breakpoint statuses are shown in the left column of the module text, with special marks.

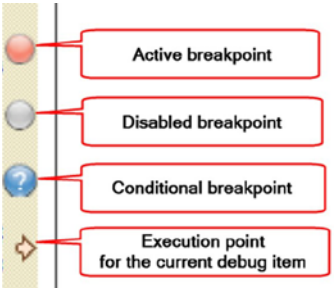


Fig. 378. Breakpoint types

You can set breakpoints in any module line at any time during debugging. If the line with a breakpoint does not contain any operators (e.g., is an empty line), or contains non-executable text (e.g., procedure or function header or variable definition), or is a continuation of the operator started on previous lines, the breakpoint position is automatically adjusted. Breakpoint position is marked with a special character in the left column of the module window. Different characters are used for enabled and disabled breakpoints (see fig. 378).

The mouse can also be used to set or remove breakpoints. To do this, double-click in the grey area of the line where you want to set the breakpoint.

Use the following commands of the Debug menu in the Designer main menu to manage breakpoints.

Command	Description
Breakpoint	Sets or removes a breakpoint on the line where the cursor is set
Breakpoint with the condition	Sets a breakpoint and opens the dialog box for adding a break condition - (a logical expression). If the condition has been added, a dialog box opens to edit the condition. If a breakpoint has been set on this line, a dialog box opens for setting a breakpoint condition for the breakpoint with the condition. Execution stops at the specified breakpoint only if the condition is met
Disable breakpoint	Enables or disables the breakpoint. The command is available if the current line has a breakpoint

Command	Description
Remove all breakpoints	Removes all previously set breakpoints in all modules
Disable all breakpoints	Disables all previously set breakpoints in all modules without deleting them
Breakpoint List	View and manage breakpoints
Stop on Error	If an error occurs, the debugger stops execution and proceeds to the module line that caused the error

Set breakpoints are saved when the configuration is closed. To view the list of breakpoints open the configuration and use the Debug - Breakpoint List command.

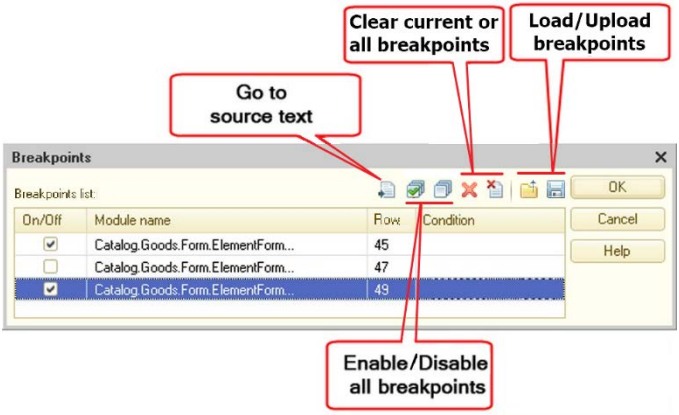


Fig. 379. Breakpoint List

Use check boxes in the On/Off column to manage the breakpoint status. If the check box is selected, the debugger stops execution at the breakpoint. For description of other buttons see fig. 379.

32.2.4. Step-by-Step Execution

In the step-by-step execution mode, the debug item completes each command and waits for a debugger command to continue execution.
When the first debug item is attached, the system adds debug management commands to the Debug menu.

There are several options for each step of module execution. Use the following commands from the Debug menu to select an option.

Command	Description
Step In	If the next executable module operator is a function or procedure call, it is incrementally executed, otherwise the debugger proceeds to the next operator.
Step Over	If the next executable module operator is a function or procedure call, it is executed completely (not step by step), and the debugger proceeds to the next operator.
Step Out	Interrupt incremental execution of a function or procedure and stop at the first operator after it is called
Go to Cursor	Interrupt step-by-step module execution, execute all statements up to the line where the cursor is set
Continue	Break the step-by-step module execution and continue free execution

When debugging applications, you need to keep in mind the following feature: if there is the `Execute()` operator in the debugged program code, then:

- step-by-step execution of the program code passed as an operator parameter (including procedure and function calls) is not supported;
- interruption of program code execution using breakpoints located inside procedures and functions and called from the program code passed as the `Execute()` operator parameter is supported.

If you debug multiple debug items, then there is a number of features of incremental implementation:

- if you stop a debug item, other debug items are also stopped when you execute the code;
- if you select Continue, it resumes execution of all debug items;
- if you select Step Over, it proceed to the next line for all debug items;
- if you select Step In (if the executable module operator calls a function or a procedure), it proceeds to the first operator in the call, while the Step Over command is performed for other debug items.

If you debug the client/server mode and the code is sequentially executed on the client and on the server (client and server debug items are attached), then:

- if you select Step In (if the executable module operator calls a function or a procedure on the server), it proceeds to the first operator in the call;
- if you select Step Out or Step Over for the last executable operator (if the executable module operator calls a function or a procedure executed

on the server and called from the module executed in the client application), it proceeds to the next executable operator in the call.

The current debug item can be selected in a special Debug Items toolbar. The toolbar has the only selection field displaying the current debug item. The selection field is only accessible when the debugger controls one of the attached debug items (e.g., after the breakpoint is triggered). The list of debug items only contains items that are currently controlled by the debugger, including the current debug item.

Use the immediate window and the Expression dialog box to get values for the relevant expressions. Call Stack can be used to trace the sequence of call procedures and functions.

If a step-by-step process is performed, the call stack and variable values (in the immediate window and the Expression dialog box) are displayed for the current debug item. When you change the debug item, it also changes the call stack and variable values.

IMPORTANT! If the client and server debug items are attached and the transition from the client to the server part is performed, then no calculations are made at the client levels of the call stack. These levels are highlighted in grey in the call stack window.

If you need to continue the module execution, select Debug - Continue Debugging to enable free module execution for the attached debug items (up to the next breakpoint). If a client application is attached for debugging, it is activated auto- matically.

To interrupt the entire debugging process (except for background jobs), remove all breakpoints from all modules and run the Debug - Continue Debugging command, if a breakpoint has been triggered. If you need to interrupt debugging and terminate the operation of the attached debug items, you should use the Debug- Finish command. In the latter case `BeforeExit()` and `OnExit()` procedures are not executed.

You can edit the current configuration and save changes during debugging.

IMPORTANT! Although during the debugging process it is possible to edit the module being debugged, the debugger does not compile the changed code -debugging is continued for the database configuration code (at the moment when debugger is started or attached). To debug the changes made to the configuration, you should update the database configuration.

If exclusive mode is set in 1C:Enterprise mode, you will not be able to save the current configuration until the exclusive mode is removed.

If a file mode infobase is in step-by-step 1C:Enterprise language code execution status or is standing on a breakpoint and there is an open transaction

(explicit or implicit), then executing some operations (e.g., an attempt to lock objects in configuration repository) can cause an error: Configuration repository operation failed. (Information collection error for configuration repository) Lock conflict when executing transaction. Could not lock FILES table. It is recommended to shut down the client application before retrying. To perform the operation during which an error occurred, you should continue 1C:Enterprise language code execution until the transaction is completed.
Table of key combinations for debugger operations can be retrieved in Help when using the application.

32.2.5. Debugging external data processors (reports)

To debug external data processor (report) modules you have to open the external data processor file in Designer using the File - Open item.
In the future, external processing modules (report) in debugger can be operated just like any other module.

32.2.6. Debug management

Debug - Restart command stops configuration execution and restarts it in 1C:Enterprise mode. If configuration has been modified, the screen prompts you to update the database configuration.
Debug - Finish command stops module execution and terminates the operation of the current debug item.
Debug - Stop command stops module execution on the current operator. This command can be used to start module debugging from the next executable string. This command is useful for module "cycling" analysis.
Debug - Stop on Error command opens the stop on error settings dialog box.
Stop on Errors check box status defines whether debugging should stop if an error occurs during execution. If execution stops on error, the following warning is displayed: Runtime error and an explanatory text about the location and cause of the error is given.
If Stop on errors containing text check box is selected, then debugging is stopped in the place of an error (if the error message contains the substring indicated in the table field). If this check box is not selected, debugging stops on an error regardless of the error message text. At the same time the specified substring is considered only if you select the check box next to it.

At the place of error, the current execution point of the current debug item is set, and the debug item where the error occurred becomes the current one. Viewing expression and variable values is used to determine the cause of the error.

32.2.7. "Expression" window

During the step-by-step execution, Expressions can be calculated using the Expression window. This window is displayed by selecting the Debug- Calculate Expression item.

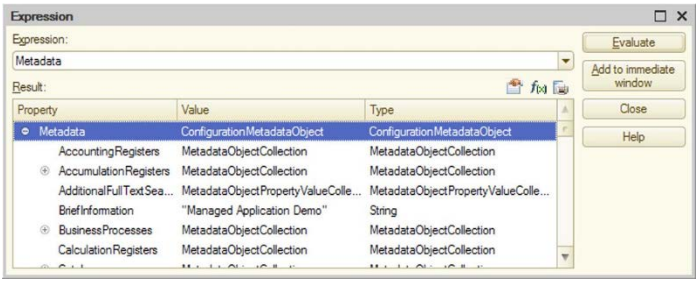


Fig. 380. Calculate expression

Add the expression in the 1C:Enterprise language into the Expression field of this window. You can add the expression "manually" or select it from a list of previously added expressions.

If the cursor in module window was placed over an expression or an expression was selected when calling the Expression window, the expression is automatically inserted into the Expression field.

Press the Calculate button to calculate an expression. The expression calculation result will be displayed in the Value field. It should be understood that calculation of expressions is performed in a real system, therefore various side effects are possible. For example, if the calculated expression analyzes the presence of some item in the infobase, and creates the necessary item in its absence, such expression will be executed “along the way” of creation only once. This is due to the fact that after the first execution, the necessary item will be created in the real infobase.

The Include in immediate window button adds the expression entered into the immediate window. This may be used to trace the change of calculation result of the expression in the process of module debugging. In this case, the expression is placed into a new line of the immediate window. If there is no immediate window on screen, it opens.

If an expression has a `String` type or is a collection of values or an array, the tools panel button Show Value in separate window becomes available.
For the convenience of viewing long strings, the View Expression value window is displayed. This window contains a control of Label type that displays the line value for viewing.
For the convenience of viewing a collection of values or an array, a window containing a table field is opened, columns in this window correspond to attribute names and rows contain attribute values. The window also shows the number of items in a collection and the first column indicates the index of each collection item.
If a specific value is also a collection of values or an array or a row, you can view these values in a separate window (as shown on fig. 381). To do this, select the required value in the context menu and select the Display value in separate window item or press the F2 key.

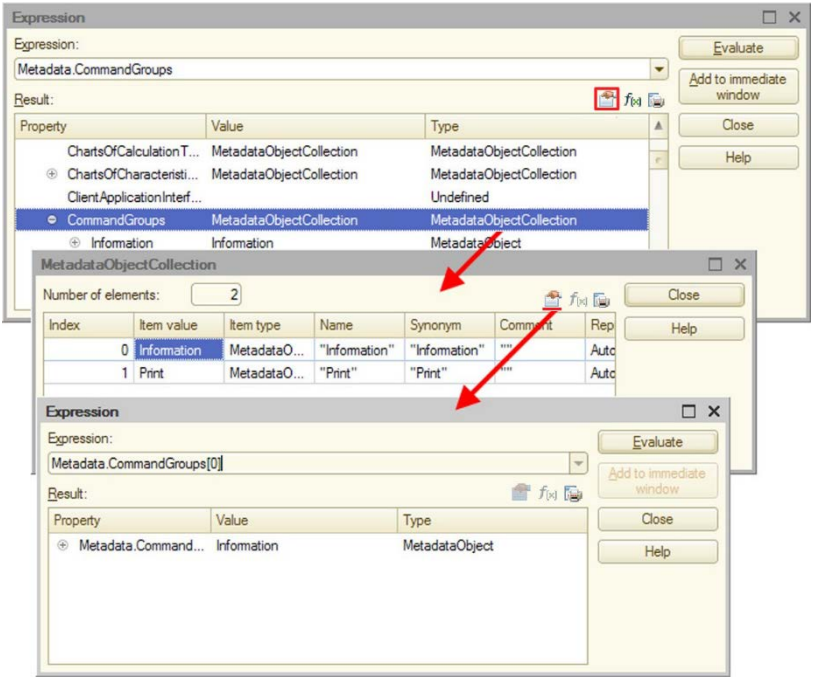


Fig. 381. Display values

For a string, a window is displayed with the string expression value in a multiline field. You can copy this expression to the clipbоard.

The contents of the calculation result table field can be exported to a text or a spreadsheet document using the Export list button. If the calculated value is represented as a tree, you can only export data from the opened strings of this tree.
The current value of the expression can also be viewed by moving the cursor over the expression. The current value is displayed as a short tooltip next to the variable. Viewing is possible if the value has a textual representation. This way can also be used to view the value of the property. If the property is represented as `<Object>.[<Object>.…]<Property>`(For example, `Controls.CommandBar.Buttons.Get.Enabled`), then viewing the value is possible by hovering the cursor over such text. The value is displayed if the text does not contain parentheses or square brackets. You can also view the value of the array item that has an explicitly indicated index (as a number or variable with value specified at the moment of viewing). To do this, you need to select the array ID and index.
When hovering the cursor over a single object, or a selected object, or a group of objects in the text of `<Object>. [<Object>. ...] <Property>` type, it's type is shown. For example, if selecting `Controls.CommandBar1` in `Controls.CommandBar1.Buttons.Get.Enabled` text, then the type of `CommandBar` will be displayed.

32.2.8. Immediate window

Immediate window- a specific window, that displays calculation results for variables and formulas that have been entered into it during debugging. Call of immediate window on screen is performed by selecting the Debug - Immediate Window menu item, available when debugging.

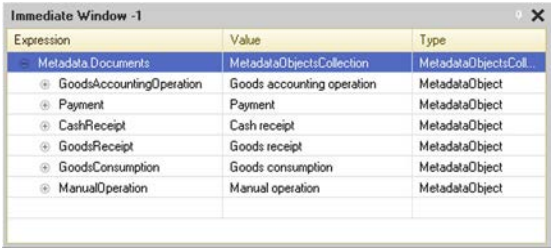


Fig. 382. Immediate window

Immediate window is a four-page form. Each page of the form contains a table box to add variables and formulas, the calculations results of which should be controlled. Formulas can include arithmetical expressions and expressions with 1C:Enterprise script functions, as well as the module functions of managed application and common modules.
Each formula is entered in the first table column, in a new row. The result of the formula is displayed in the Value column. The Type column displays the type of expression. If formula is invalid, the following error message is displayed instead of the result: Syntax errors detected!
Immediate window and calculation results are managed using the context menu commands.
The calculation result can be copied to the clipboard by selecting Copy result of the second column context menu.
For some data types (see the previous section) it is possible to view the values in a separate window.
Immediate window can also be output to a spreadsheet or text document.
If the formula data source changes, you have to recalculate the result. Select Recalculate or Recalculate All in the immediate window context menu.
The immediate window uses two modes for page selection: in the first mode, the pages are selected using the tabs below; in the second mode, - they are selected using the context menu. You may select the mode by checking or unchecking the Tabs option in the context menu.
To clear the contents of a line in the immediate window, select the line and press the Del key.
The immediate window may be displayed in different modes. After debugging is complete, the window closes automatically.

32.2.9. "Local Variables" window

During debugging, it is possible to open a special window that contains a list of local variables of the current method, where the current execution line of the 1C:Enterprise language is located. The window is called up by selecting the menu item Debug - Local Variables, which is available during debugging.

The local variables window contains the parameters of the method being debugged and all local variables used in the method at the current level of the call stack (by default, - at the top of the call stack). At the very beginning, the variables are in the uninitialized state. As they are assigned any values, - these values appear in the local variables window. The order of the variables in this

window is undefined. You can not edit the Variable column in this window. The rows composition in this window is not available for editing.
Switching on the call stack, changing the debug item, as well as updating the window contents will be made automatically after each action performed during debugging,
The calculation result can be copied to the clipboard by selecting Copy result of the second column context menu.
For some data types (see the previous section) it is possible to view the values in a separate window.
Immediate window can also be output to a spreadsheet or text document.
The local variables window can be displayed in various modes. After debugging is complete, the window closes automatically.

32.2.10. Variable value change

When the debugger stopped at a breakpoint, it is possible to change the values of variables and properties of objects available for writing. Access to the change function is possible from the expressions window, immediate window and the local variables window. In the expressions window, to change the value of a variable, you need to move over this variable and press the f(x) button; in the immediate window or the local variables window, select Set new value as an expression in the context menu of a specific variable. The term "variable" refers to not only the value of an ordinary variable, but also the property of the collection (arbitrary nesting).

After calling the value change function, a special window is displayed for entering the expression. In the upper part of the window, enter the expression which value will be set to the variable. Name of the variable is represented by the header of the expression input field. In the example in **Ошибка! Источник ссылки не найден.** it is proposed to change the value of the variable `Course`.
In the expression input window, you can enter constant values of various types, as well as any expressions that are correct in the 1C:Enterprise language. There is context help in the expression input window, that displays the context of the debugger current method.
After clicking the Install button, the installation result will be displayed at the bottom of the expression input window. This could be a phrase The value was successfully changed, or an error message if there is one.
The variable value can be changed directly in the window where the variable value is displayed. To do this, double-click the left mouse button on the text

in the Value column. Please remember, that in this way you can only set the value of primitive types.
It must be clearly understood, that the possibility of changing the values of variables during debugging can lead to various unpredictable consequences in the operation of the information system. So, if in the process of debugging, for an object recorded to the database (for example, in the `BeforeWrite()` handler), you can set the property values that cannot be there during actual system operation. However, these incorrect values will be written to the database. At that, other system algorithms may not work correctly. In this regard, you should cautiously use the option of changing the variable values and object properties during debugging.

32.2.11. Call stack

Call stack shows the sequence of procedure and function calls that led to the currently debugged module line.

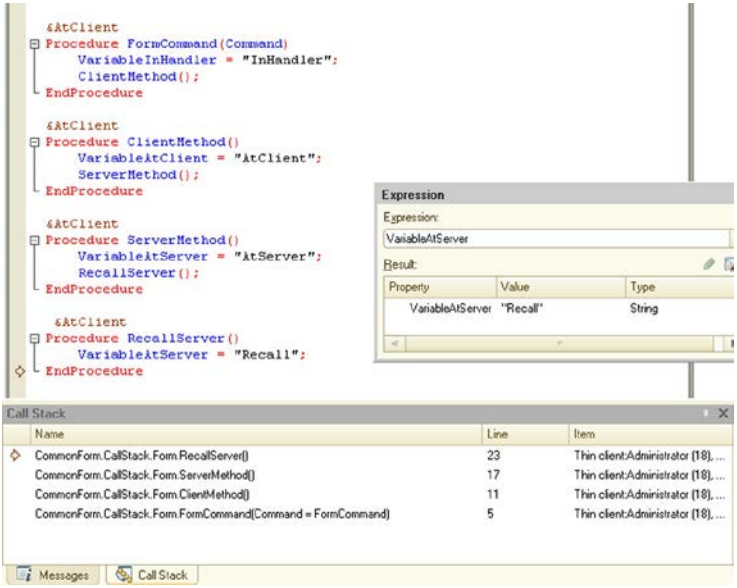


Fig. 383. Call stack

The window shows:

- executed method - the Name column;

- module line number - the Line column;
- the debug item where the module is located, -the Item column.

The first column of the window can have two icons:

- The yellow arrow indicates the top of the call stack.
- The green arrow (to the left of the CommonForm.CallStack.Form.ServerMethod() line) indicates a method in the context of which variables are calculated in the Expression window and the Immediate window. If the top of the call stack matches the current context, the green arrow is not shown. To change the context and to set the cursor position to the row indicated in the Row column, double-click the required row in the Call Stack window.
  In the example shown in fig. 383, the `VariableOnServer` variable has the OnServer value, since the debugger is switched to the `ServerMethod()` method context (as indicated by the green arrow next to CommonForm.CallStack.Form.ServerMethod()).
  If you switch the context to the top of the stack (double click the row with the yellow arrow), the value of the `VariableOnServer` variable is set to Recall, because the context switches.

Some rows in the Call Stack window are shown in grey (the last two rows in fig. 383). This means that you can switch to a module line, but you cannot evaluate expressions in the context of the called method. Thus, those stack items are selected that are "located" on the client, and the top of the stack is "located" on the server.

32.2.12. Debugging Protected Modules

If the module line calls a procedure or function located in a protected module, and you use the Step In command at the first attempt to go to the selected procedure or function, the debugger prompts you to enter a password.

If you enter the password correctly, the debugger opens the protected module and control passes to the first line of the called procedure or function. When you address to the protected module again in this session, no password is requested.

If you cancel the password entering, the debugger skips the protected code (but still executes it) and proceeds to the first code line following the protected module. Similarly, the command Step Out is executed.

The procedures and functions names are always specified in the call stack. If the protected module has not yet opened, then the first time you attempt to go to the selected protected module procedure, the debugger asks for an access

password. Upon subsequent transitions to a protected module from the call stack, the password is not required (in the current session).

32.3. Performance Snapshot

Performance snapshot can be used to evaluate performance of the entire configuration or its part that is running within a debug item of any type. You may measure the use frequency for specific code sections and speed of their execution; specify the code that was executed on the server or -on the client; and indicate the code lines that have initiated server calls. If there are several methods to solve a problem, you may implement them all, and then choose the fastest one.

Please remember that the comparison should be made under the same conditions. For example, if during the task performance using one of the compared methods, the computer processor was performing some other task, this may affect the accuracy of the comparison. There are other, less obvious reasons why the measurement conditions are different. Therefore, when you compare two methods with similar performance, it is recommended to perform several snapshots - for each method for estimating and averaging random scatter.

Use the Debug - Performance Snapshot command to measure performance. When you select this command again, the measurement stops and the measurement results window opens. Enabling and disabling the performance measuring affect all debug items currently attached to the debugger.

32.3.1. Available Action Options

If you need to measure the configuration performance, including the system startup section, you must first run the Debug - Performance Snapshot command, and then start 1C:Enterprise. The time elapsed between the start of measurement and the start of system operation is not considered in the measurement results.

If you do not want to include the startup section in the snapshot, first launch 1C:Enterprise and prepare it for execution of the required section. Then switch to the Designer and start the performance snapshot.

If you need to include in the measurement the section that is executed at the 1C:Enterprise shut down, then regardless of whether Option 1 or 2 was used to start the snapshot, you should shut down the program and then switch to the Designer mode. In this case, you do not need to stop the measuring manually. Once the measurement results are ready, they are displayed on the screen.

If no measurement is required for the section executed at the 1C:Enterprise shut down, then the performance snapshot should be stopped to view the measurement results. For example, to analyze the procedure for any document, you need to start the program, open the document and fill it, switch to the Designer mode, start the performance snapshot, switch to the 1C:Enterprise mode, post the document, switch to the Designer mode and finish the performance snapshot.

32.3.2. Measurement Results

Measurement results - references to specific module lines that specify the frequency and length - of execution. They are displayed as a table box with the following columns:

- Module - contains the name of the module.
- Line number - the module line number;
- String - the module line text;
- Usage count. - number of calls to this string during measurement;
- Time (net) - total runtime (in seconds) for this string during measurement;
- %(Time) (net) - percentage of the total runtime for this line to the total snapshot time (total snapshot time equals the sum of all time intervals of executing the configuration code). In this case, 100% is the execution of code on the client.

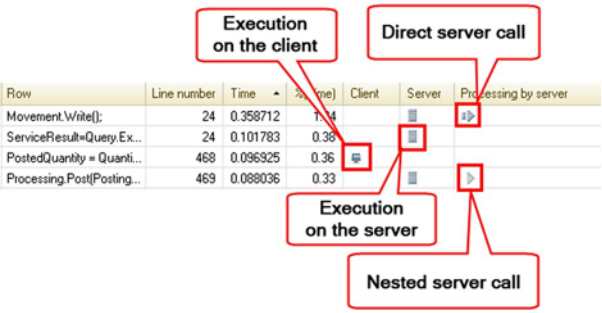


Fig. 384. Performance Snapshot Icons

- Client - this icon (see fig. 384) marks the code lines that run on the client;

- Server -this icon (see fig. 384) marks the code lines that run on the server;
- Processing by server - these icons mark the code lines that initiate server calls:
  - the server was called at the platform level, or procedures and functions executed on the server were called directly (Direct server call on fig. 384);
  - local call of procedure/function executed on the client, within which the server call occurred at the platform level, or procedures/functions executed on the server were called directly (Nested server call on fig. 384).

Runtime for each string in the performance snapshot is made up of runtime of string operators ("net time") and call time of the procedure (function), if there are any in the string. You can select Include runtime for procedure and function calls check box to choose the time to display: full time (as a sum of call time and "net time") or "net runtime".

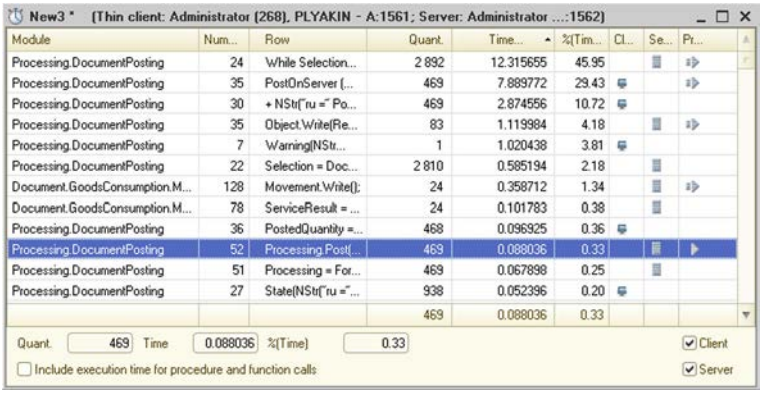


Fig. 385. Measurement Results

If a string contains at least one procedure (function) call, the runtime includes runtime of string operators and call time of the procedure (function).
If the check box is selected, the procedure (function) call time is included in the full runtime.
If this check box is cleared, the measurement results include only the code lines runtime, but not the runtime of procedure (function) that is called in this line. In this case the total runtime of this line (in the Time column) does not

reflect the actual time spent by the system to process this line. Please remember that execution of called procedure (function) may be time-consuming, and in this case this time is not included in the result ("net time").
The check box is selected by default and its state is saved between the sessions. If the state change, the time columns headers will change.
If Client check box is selected, you will see the measurement results of the code on the client.
If Server check box is selected, you will see the measurement results of the code on the server.
If Client and Server check boxes are selected, you will see the measurement results of the code on the client and on the server.
The check boxes are shown if server infobase is being debugged. The check boxes are available if you are debugging a server debug item.
If there are several open windows with the performance measurement results, when you hover the mouse pointer over the results column, you can see a tooltip with the URL of the file which data is shown in this column.
Double-clicking in the column with the performance measurement results in the module editor, it proceeds to the corresponding string in the file with the performance measurement results.
In addition to the special window, the measurement results can also be viewed in the module source code window. If a performance snapshot window is open in the debugger, a column appears in the module windows showing the number of calls for the current string and the percentage of its work to the total time.

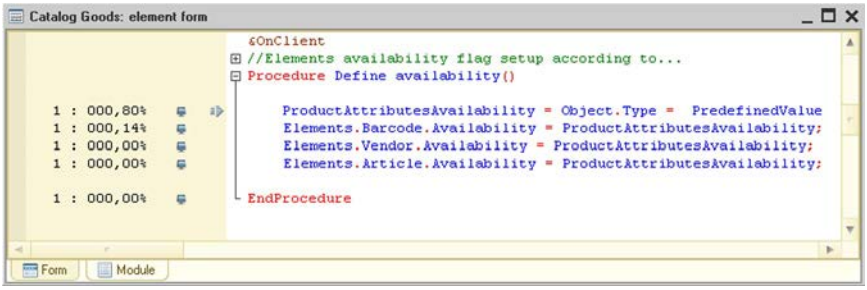


Fig. 386. Measurement Results in Module Text

The icons show the code execution on the client or server, as well as server calls (similarly to the measurement results window, description of icons (see fig. 384).

Double-click a line in the measurement results window to switch to the corresponding line in module window.
If several snapshots are opened at the same time, the module text windows contains the corresponding number of columns.
Closing the measurement results window removes columns with the number of calls and a percentage of working time from the modules.

32.3.3. Sorting Measurement Results

Measurement results can be sorted by any column in the report.
Sorting is performed by clicking on the header of one of the columns. Clicking on the Module or Line headers performs sorting by line number, Usage count -by the number of line calls, Time Or %(Time) - by operating time. If the code execution performance measuring has been performed on both the client and the server, sorting by these columns is also possible.

32.3.4. Selective Summarization of Measurement Results

Cumulative performance or combined characteristic may be useful for analyzing the measurement results. If several lines are selected in the results window, their summaries - total number of calls, total working time in seconds and percentages - are displayed in the bottom part of the window.
Use Include runtime for procedure and function calls check box to select one of two methods for approximate tracking of nesting levels for summarization. If this module contains both a line calling a certain procedure and lines of the procedure text, then you should not select both: since it will result in double counting of runtime. If you still need to select them (e.g., when it would take too much effort to track these items), you may uncheck this option to avoid duplicate counting. However, if all the called procedures - are external to the module, it is better to select this check box. Then the total runtime includes the runtime of these procedures providing more accurate results.

32.3.5. Performance Snapshot Features during Client/Server Infobase Operation

When the performance snapshot is produced, the results for the 1C:Enterprise script code executed for the same connection number on both client and server are summarized. In this case the results indicate which code lines have been run on the client and which - ones on the server; which lines have called procedures or functions executed on the server, as well as platform system calls (e.g., Execute operator for a query, or database object record operator) on the server.

When viewing the results, you can specify what to display in the snapshot: snapshot for client, for server or for client and server together. The mode is selected using check boxes in the lower right corner of the window with the performance snapshot results.

For server calls, the time to make the server call on the server is taken into account. The entire procedure can be compared to execution time of nested procedure and function calls for a client application.

32.3.6. Saving Results

Measurement results can be saved in a file using the File - Save and File - Save As commands. Select directory and specify file name in a standard save dialog box. The result file has an extension *.pff.

You can open a file with the measurement using the File - Open File command. To filter measurement files use the Performance Measurement (*.pff) filter.

32.4. Server call delay imitation

The purpose of the mechanism - is the simulation of the application operation under the conditions of significant time delays arising during interaction with the server.

NOTE. The mechanism can be enabled only in the thin client or in the thick client managed application mode.

The mechanism provides a simulation of time delays arising in the following cases:

- the server call delay: in seconds for each server call;
- delay in sending data to the server: in seconds per 1 Kb (1024 bytes) of data sent to the server.
- delay in receiving data from the server: in seconds per 1 Kb (1024 bytes) received from the server.

The mechanism can be enabled using the Designer configuration settings, as well as using the command line key /SimulateServerCallDelay [-CallXXXX] [-SendYYYY] [-ReceiveZZZZ], where:

- Call - parameter indicating the server call delay in seconds; it is equal to 1.45 seconds unless other value is specified.
- Send - parameter indicating the delay in seconds per 1 Kb of data sent to the server. It is equal to 0.45 seconds, unless other value is specified.
- Receive - parameter indicating the delay in seconds per 1 Kb of data received from the server. It is equal to 0.15 seconds, unless other value is specified.

Example:

```
/SimulateServerCallDelay  Call2.1  Send1.3  Receive1.2
```

NOTE. The maximum delay value can be set to 9,99 seconds. 0 means that this delay does not apply.

Server call delay simulation can also be enabled in the 1C:Enterprise mode. To do this, use the Tools - Options dialog box. Configuration of the server call delay simulation in the 1C:Enterprise mode is only valid in the current session and is not saved.

32.5. Server call history

This mechanism is used to debug client/server system and assess the amount of data transferred between the client and the server.
The mode can be enabled through the Designer settings cofiguration, or by using the command line key /DisplayPerformance. Displaying of performance indicators can also be enabled in the 1C:Enterprise mode by selecting the Show performance indicators check box in the Tools - Options dialog box. The mechanism is disabled by default.

If the mode is enabled, 1C:Enterprise displays performance indicators in the performance indicator panel at the bottom of the main application window.

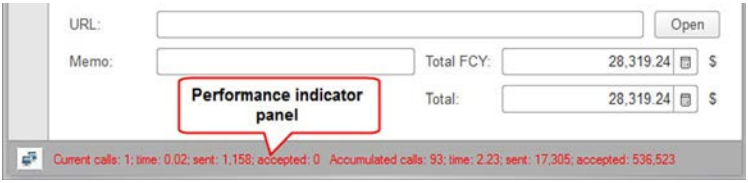


Fig. 387. Performance indicators panel

If the delay simulation mode is enabled when a server is called, the picture in the window showing server calls will look like this:

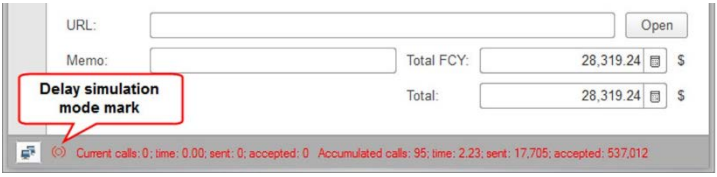


Fig. 388. Server calls in slow connection simulation mode

If you have not yet opened or have already closed the main application window, but the application code is running, then the performance indicators window looks like this:

Fig. 389. Server calls

If the server delay simulation mode is enabled, the performance indicators window looks like this:

Fig. 390. Server call delay imitation

When you change the data displayed in the server call window, it is displayed in red.

By default, the window consists of two counters: Current calls and Accumulated calls:

- Current calls - shows statistics of server calls since a certain point in time: the start of the mechanism or the last user action, within 0,2 s. after which there were no other user actions.
- Accumulated calls - shows statistics of server calls since the mechanism starts or its last forced reset is triggered from the context menu.

Performance indicators settings can be configured using the Performance indicator settings dialog box. You can open this dialog box by clicking the button in the performance indicator bar and selecting Settings in the menu.

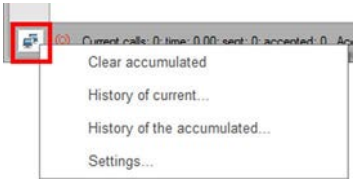


Fig. 391. Settings menu

You can specify the following parameters in the settings dialog box:

- Periodicity group - indicates the period for displaying the counter values;
- Indicators group - defines which indicators are to be displayed in counters.

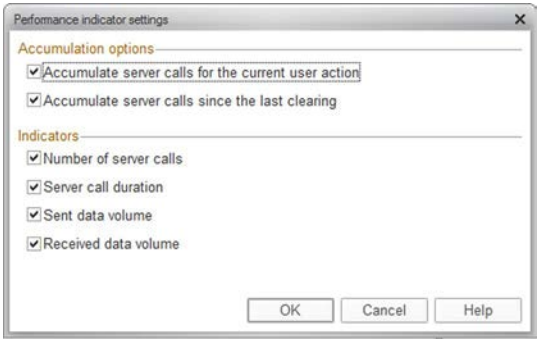


Fig. 392. Performance Indicators Settings

The context menu of this window contains the following commands:

- Clear accumulated - clears the accumulated calls counter;
- History of current… - opens the history for the current calls counter.

- History of accumulated... -– opens the history for the accumulated calls counter.

The settings are common for all users and all infobases of the computer. Setup of server calls window display in the 1C:Enterprise mode is only valid in the current session and is not preserved.

The following features of the mechanism should be considered:

- When working in the file mode (if directly connected to the infobase), the amount of data is displayed without compression, since compression is not applied in this case.
- when working in client/server or file mode with a connection through a web server using the SDC traffic compression mode, the indicator values may be affected by the sequence of actions, since compression is performed using the previous transmitted data.

In the web client, this mechanism has several features:

- Only synchronous calls are taken into account.
- The amount of data transferred is displayed in characters, not bytes. Measurement results can be used to compare various call options in the web client, but it is not recommended to use the results to compare the amount of transferred data for different client types.
- Performance indicators are not displayed until the main application window opens, while the values of the indicators accumulate. After the main application window is displayed, you can see the results of web client startup.
- The window with indicators has a larger size than on other clients, and is not displayed over the other windows.

32.6. Automated testing of applied solutions

32.6.1. General information

Automated testing is a simulation of interactive user actions and verification of the results of these actions.

IMPORTANT! Automated testing is only supported for managed applications.

The automated testing scenario is a program in the 1C:Enterprise script. The scenario describes a sequence of actions and validation of their results. There are special objects in the 1C:Enterprise language that can be used to simulate

interactive actions in the application being tested and provide access to the logical model of the client application interface and managed form items.
Two systems participate in the testing process: **testing manager** and **testing client**. The testing manager provides:

- Establishing connection with the testing client to run a testing scenario (test connection);
- Execution of the testing scenario;
- Evaluation of test results (if required).

The testing client provides:

- Execution of testing scenario commands passed from the testing manager;
- Transfer of data required for test results evaluation to the testing manager.

The test manager can be a thick or thin client, the test client can be a thick client, a thin client or a web client. One testing manager can be simultaneously connected to multiple testing clients, whereas a testing client can only be connected to one testing manager. For testing manager and the testing client interactions, a direct TCP/IP connection via a specific port (that has to be "open") is used. The testing manager and the testing client, that is a web client, interact using the 1C:Enterprise server (for client/server mode of tested application) or a web server (for file mode). Testing does not require any changes to the tested application.
The automated testing mechanism does not allow receiving and managing information about the structure of window objects for the system windows of a web browser. Therefore, it is necessary to avoid actions in the web client that could provoke the appearance of such windows, since this will make it impossible to automatically continue the test. An example of such action is an attempt to add text into the input field with automatic filling: in this case, the built-in dialog box will be displayed in the thin client, and in the web client -system dialog box. Therefore, when planning a test, it is necessary to keep this feature in mind and create the test accordingly.
The testing scenario can be stored either in the tested application or in a separate application. The scheme of interaction between the testing manager and the testing client does not change.
It should be understood that the test manager (and the executing test scenario) do not have direct access to the tested application. I.e, the testing scenario has access to all features of the application where the scenario is running (testing manager) and access to the logical interface model and forms of the tested application.

Running a client application in testing client mode does not affect performance and other client application settings, however, it allows any test manager to connect to it, knowing the name (or IP address) of the computer and the port, through which the interaction takes place.
The application also can be used to save interactive user actions to an .xml format file. You will be able to use this file later for automated testing. To perform such recording, a specific start-up mode of tested application is intended. Managing the recording of interactive actions is also available from 1C:Enterprise language.

32.6.2. Testing system start-up

32.6.2.1. General information

To run the testing scenario, both the testing manager and the testing client should be launched. Startup order is arbitrary. It is important that at the moment when you run the testing scenario, the testing client is running and available at the address used for connection by the testing manager.
Starting is possible both from the command line (this method will be discussed below) and from Designer. The application that you start from Designer can perform one of the roles (testing manager or testing client), depending on the settings specified in Designer settings dialog box.

32.6.2.2. Testing client start-up

You can start the testing client by using the `TestClient` command line parameter for client application startup. You can also specify the number of the TCP port to be used for testing manager and testing client interactions. To do this, use the `TPort` parameter of the `TestClient` key. If there is no specified value for this parameter, 1538 is used. This parameter should be specified if multiple testing clients run on a single computer.
Thus, the startup line of the client application that should act as a testing client, is as follows:

```
1cv8c ENTERPRISE /IBName "Test Application" /TestClient  Tport 1843
```

In this example, the thin client (1cv8c) will be used as the testing client for the Test Application infobase, and will use the TCP-port 1843 for interactions with the testing manager.
If a client application is started as a testing client, the text Testing Client - will be displayed in the title of the main application window. Similar text will be also displayed in the About program window.

In order to connect to the testing client represented by the thin or thick client, you should know two parameters: The IP-address (or computer name) where the testing client is started, and TCP-port number to be used for interaction.
If you start the web client, the situation is a bit more complicated. In addition to specifying the `TestClient` parameter, you need to specify one more startup command-line parameter: `TestClient<ID>`, that contains a UUID for the tested client application. To unambiguously specify the testing client, in the case of the web client, three parameters are needed:

- Address of the web server where the tested application is deployed. You do not have to know the infobase address!
- TCP-port number to be used by the web server to transfer data between manager and the testing client.
- The ID of a specific instance of the tested application running on the web client.

Since a single web server can serve several infobases and several connections with one infobase (also for testing purposes); you must specify the web server which TCP ports shall be used to interact with the test manager. This is done using the testcfg.xmlfile located in 1C:Enterprise settings directory, which serves the published (on the web server) testing client.

32.6.2.3. Testing manager start-up

To start the testing scenario, start your client application in the testing manager mode. Use the `TestManager` command-line key to run the client application. The web client cannot be used as the testing manager.
Thus, the startup line of the client application that should act as a testing client, is as follows:

```
1cv8c ENTERPRISE /IBName "Test Manager" /TestManager
```

If the client application is started as a test manager, the text Test Manager - will be displayed in the title of the main application window. Similar text will be also displayed in the About program window.

32.6.2.4. Start-up in recording the interactive action log mode

The client application is started in a mode that allows you to record interactive actions, either using the `UILogRecorder` command line key or using the appropriate command of Designer menu Service- Start to record the user actions log.

You can also specify the IP port number to be used by the testing manager to control recording of the interactive action log. To do so, use the `TPort` parameter of the`UILogRecorder` key. If there is no specified value for this parameter, 1538 is used. You can also select the location where the system wil save the log file. To do this, use the `File` parameter of the `UILogRecorder` key. If this parameter is not specified, after the file is recorded, the client application opens a text document window containing the saved log.
Thus, the startup line of the client application, which should allow to generate a log of interactive actions, looks as follows:

```
1cv8c ENTERPRISE /IBName "Test Manager" /UILogRecorder
```

With this launch, an appropriate command bar is added to the upper right corner of the main window of the client application that controls the journal entry.

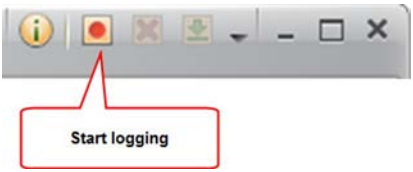


Fig. 393. Testing command bar

If you press the only available button on this panel, the recording of actions performed by the user will begin. The panel will also change it's type (all buttons will become available).

Fig. 394. Testing command bar during recording

You can use the command bar to finish interactive action recording, this will record the list of actions to a file (if the `File` parameter is specified) or will

open a text document window with a list of actions. There is also a feature that allows to interrupt or pause and continue log recording.

32.6.3. Description of features

In terms of the testing scenario, the tested application is a set of objects that is used to simulate interactive actions in the tested application and does provide access to the logical model of the client application interface and managed form items:

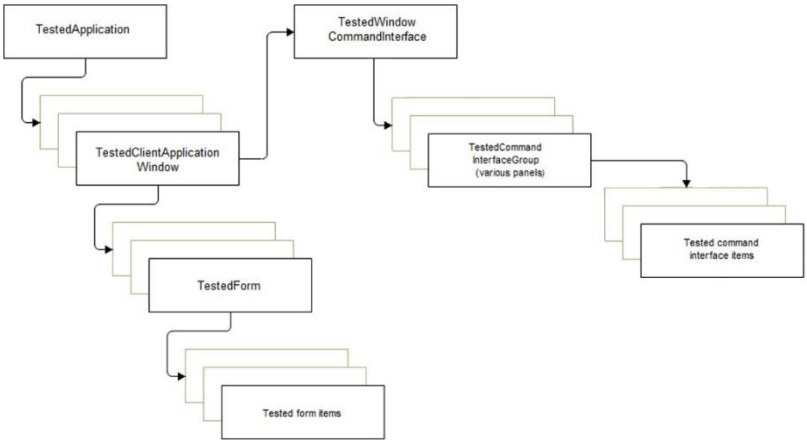


Fig. 395. The scheme of the tested application

The presented objects will be discussed in detail in the section below.

TestedApplication

Provides the following features:

- Connect to the tested application;
- Disconnect from the tested application;
- Get an active window;
- Get a list of tested application windows;
- Get current information about an error;
- Get the home page of the tested application;
- Find an object with specified parameters in the subordinate object hierarchy;
- Wait until an object with specified parameters is displayed on the screen;

- Get and clear the accumulated performance indicators of the tested application;
- Get the command interface of the tested application;
- Simulate operations with the file selection dialog box;
- Manage logging of interactive user actions in the tested application;

TestedClientApplicationWindow

Describes the client application window and provides methods for window management:

- Get a collection of subordinate objects for the window;
- Search for an object with specified parameters in the subordinate object hierarchy for the client application window;
- Determine that this test window describes the home page (desktop);
- Get the command interface of the tested client application.
- Operate with user messages.
- Close the tested form;
- Run a command using a navigation link.

TestedWindowCommandInterface

Describes the command interface for the tested form of the application. It has several subordinate objects `TestedCommandInterfaceGroup objects`:

- Sections panel (in case it contains sections);
- Navigation panel (except for Taxi interface);
- Actions panel (except for Taxi interface);
- Current section functions panel;
- Open items panel;
- Tools panel;
- Favorites panel;
- History panel;
- Functions menu (except for Forms in separate windows interface).

For auxiliary windows, the command interface of the form only has the form navigation panel. When opening forms in tabs, the `TestedWindowCommandInterface` object, related to the main application window, does not allow getting information about subordinate groups and items of the command interface.

When operating with a collection of objects that are acquired using the `GetChildObjects()` method, it is recommended to use the `FindObject()` or `FindObjects()` methods instead of referring to a collection item by index.

TestedCommandInterfaceGroup

This object describes the command interface group and allows you to get a collection of subordinate objects for the group as well as search for an object with specified parameters in the subordinate items hierarchy.

`TestedCommandInterfaceGroup` can consist of two types of items:

- `TestedCommandInterfaceGroup` ,
- `TestedCommandInterfaceButton` .

TestedCommandInterfaceButton

Describes a command of the command interface and can be used to simulate a button click.

TestedForm

Describes the tested application form and provides methods for form management:

- Getting a collection of subordinate objects for the form;
- Finding an object with specified parameters in the form subordinate object hierarchy;
- Getting the current active item of the form;
- Getting the default button;
- Getting the modified flag;
- Getting the command bar of the form;
- Selecting from the list or from the menu;
- Change the current form item to the next/previous item based on the form item tab order (the item list in the managed form editor).

TestedFormField

This object describes the form input field and provides methods for managing this form item:

- Getting a collection of subordinate objects of the form field.
- Finding an object with specified parameters in the subordinate object hierarchy for the form field.
- Defining the form item appearance.
- Can be used to simulate interactive actions for the following types of managed form fields:
  - Entry field;
  - Label field;
  - Picture field;
  - Switch field;
  - Check box field;
  - Indicator field;

    - Slider field;
    - Calendar field;
    - Text document field;
    - Formatted document field;
    - Field of spreadsheet document.
- No simulation of interactive actions is available for the following types of managed form fields:
    - HTML document field;
    - Geographical schema field;
    - Chart fields (of any type);
    - Graphical schema fields.
- If the object is associated with an input field, the following options are provided:
    - Get the text editable in the input field;
    - Simulate pressing the Clear button;
    - Simulate pressing the Select button;
    - Simulate pressing the Open button;
    - Simulate the opening of the selection list;
    - Simulate row selection from the selection list;
    - Simulate the creation of a new item from the selection list;
    - Cancel editing the row in the input field;
    - Simulate pressing the increment/decrement buttons;
    - For dropdown list:
        - Open the list;
        - Wait for the completion of the list;
        - Close the list;
        - Make a selection from the dropdown list.
- If the object is associated with a check box field, the following options are provided:
    - Simulate toggling.
- If the object is associated with a radio button field, the following options are provided:
    - Simulate the selection of a radio button value.
- If the object is associated with a calendar field, the following options are provided:
    - Simulate setting a selected date;
    - Simulate date selection;
    - Simulate forward and backward operations (by month and year).
- If the object is associated with a spreadsheet document field, the following options are provided:
    - Getting the address of the current region;

    - Getting the text of the current region;
    - Simulate the start of editing the current region.
    - Simulate the end of editing the current region;
    - Getting the current editing mode for a spreadsheet document.
- If the object is associated with a spreadsheet document input field, text document or formatted document, then you cave save a document to a file.

TestedFormTable

Describes a form table and provides methods for managing this form item:

- Get active table item;
- Get table cell text;
- Navigate through table rows and columns;
- Select a row;
- Add or remove a row;
- Get current table editing mode;
- Change deletion mark on a row;
- Allocate and receive selected rows;
- Change the table sort order;
- Navigate through hierarchical list levels;
- Collapse or expand node for hierarchical lists;
- Check the state of the hierarchical list node (whether it is collapsed or expanded) and expandability of the current row in hierarchical list.

TestedFormButton

Describes a form button and can be used to simulate pressing it.

TestedFormGroup

Describes a group of form items and provides methods for the group management.

- Expand group;
- Collapse group;
- Get the current group page.

TestedFormDecoration

Describes a form decoration and can be used to simulate clicking a hyperlink or image.

32.6.4. Sample Testing Scenario

Here below is a simple example of creating a new item in the Goods catalog, in some configuration. It is important that the item main form contains a field with the Name header and the Save and close button.

```
// Connect to the tested application
TestedApplication = New TestedApplication("localhost");
TestedApplication.Connect();
// Find main window
MainTestedApplicaitonWindow =
TestedApplication.FindObject(Type("TestedClientApplicationWindow"))
;
MainTestedApplicaitonWindow.Activate();
// Run a command of item creation in the Goods Catalog
MainTestedApplicaitonWindow.ExecuteCommand("e1cib/command/Catalog.G
oods.Create");
TestedApplication.WaitForObjectDisplayed(Type("TestedForm"),
"Good*");
TestedForm = TestedApplication.FindObject(Type("TestedForm"),
"Good*");
TestedForm.Activate();
// Set a name for a new item
FormElement = TestedForm.FindObject(Type("TestedFormField").
"Name");
FormElement.Activate();
FormElement.InputText("New item");
// Save item
FormElement = TestedForm.FindObject(Type("TestedFormButton"), "Save
and close");
FormElement.Click();
```

32.6.5. Service features

32.6.5.1. User Action Log Entry

You can run a client application in the mode that supports logging interactive actions. This log can later be saved to a file.

32.6.5.2. Converting the user action log into an automated testing scenario

You can convert a saved user action log into the 1C:Enterprise script code for subsequent use. To perform this conversion, you can use a special tool located on the ITS disk: Technological support - 1C:Enterprise 8 - Universal reports and processing -Convert user action log (http://its.1c.ru/db/metod81#content.5014:1).
Using this tool, you can do the following:

- Convert both a log file (e.g., a file saved earlier) and the text entered manually (in the log format).
- Specify the 1C:Enterprise script option for the resulting code written in the 1C:Enterprise script.
- Add code to implement the connection to the test client.
- Specify whether long sequences of actions should be split into procedures, each working with a single test application window.

The resulting code can be used, for example, for multiple automatic execution of the converted set of actions.

Chapter 33. Compare and merge configurations

33.1. General information

You can use the configuration compare and merge mode to compare two configurations in detail and to merge them. Selective merge based on the comparison results is also possible. Compare and merge of main configuration and configuration extensions are not supported.
You can use the merge mode, for example, when one configuration is developed by several persons at the same time. Sometimes they need to merge the results of their work (or merge all results at the end of development).
The configuration compare and merge mode can also be used to compare two configurations without merging.
Eventually, if you need to upload changes to the source configuration, you should review and evaluate them using compare and merge configurations mode.
This mode compares both common properties of configuration objects such as catalogs, documents and logs, and individual objects and tabular sections attributes. The following object forms are compared separately. modules, description texts and templates. You can view the comparison results in details as well.

33.2. Configuration comparison conditions

When the comparison mode is activated, the program analyzes the configurations and map the objects.
Objects without a match can be either new or deleted objects or they may be objects from the same category but named differently in different applications. You can configure interactive mapping for these objects.
During the automatic mapping, the program analyzes configuration objects with the same names. For example, `Catalog.Goods` in one configuration and `Catalog.Goods` in another configuration are considered as one and the same catalog; they are mapped and compared.

Objects with different names are compared by internal IDs assigned by the Designer during their creation. Internal IDs are required to simplify mapping of configuration objects and make it automatic. This situation may occur when the object name changes during configuration development. If object names are different while their internal IDs match, these objects are automatically mapped. Form mapping is performed in a special way. A form can contain several pages. First forms are analyzed by names; then pages with different names are analyzed by (non-blank) headers; and then pages with different headers are analyzed by pictures. If pages are mapped, their controls are compared and merged. If pages could not be mapped, they are considered different and the result form (after the merge) contains all these pages separately.
Pages that have none of the three properties set (Internal ID, Header and Picture) are merged by pairs in accordance with the selected sequence order.

33.3. Launch of the configuration compare and merge mode

As you may see further below, although compared configurations can be considered as basically equal (when merging, you can give priority to any of them), in a certain aspect the current configuration is considered as the master configuration, and non-damaging changes are introduced to it from the loaded configuration. Based on these considerations, it is recommended to choose which of the two configurations is to be made the main one. The selected configuration becomes the master configuration when 1C:Enterprise starts up in the Designer mode.
Before comparing and merging the loaded configuration with the master configuration, you have to save it to a file first.
To load the second configuration, use the Configuration - Compare and merge with configuration from file... command. The standard dialog box Select a configuration file will open. Find and open the file of the loaded configuration with the extension (*.cf) in this dialog box.
If the master configuration is blank (the application development has not yet begun), the Designer prompts the user to perform the configuration full boot. Upon the user confirmation, it proceeds to full boot of the configuration and the Configuration window opens. Upon cancellation of full boot, the configurations are compared.
Designer status bar displays messages during the process of loading and comparing of the loaded configuration with the master configuration. These messages show configuration objects that are being currently compared. The status bar also displays a comparison progress indicator.

Loading and comparing can take a lot of time. It depends on the size of the compared configurations.

33.4. Comparing configurations

In addition to the basic configuration compare and merge mode described in the previous section (when one of the compared configurations is always set as the master configuration), the Designer can compare two arbitrary configurations (e.g., configuration files), or repository configuration (any version) and vendor configuration, etc.
Configurations to be compared can be selected in the Configuration - Compare Configurations menu.
The following dialog box opens:

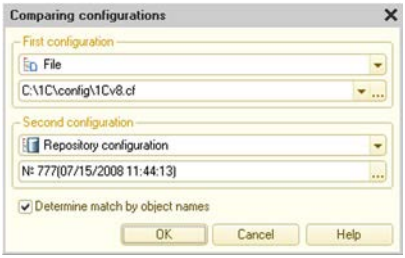


Fig. 396. Comparing configurations

The dialog box contains two sections to select the configurations to be compared. Techniques used in both sections are the same. You can select the following configurations to be compared:

- the master configuration,
- vendor configuration (if the current configuration is supported),
- database configuration,
- repository configuration version (if the master configuration is connected to the configuration repository),
- configuration extension,
- configuration extension stored in the database;
- configuration extension version from the repository (if the configuration extension is connected to the configuration repository);
- configuration or configuration extension in a file.

If a file is selected as one of the compared configurations, the field is provided in the dialog box to select a configuration file (extension .cf) or extension configuration file (extension .cfe).
If the option chosen for comparison is related to the configuration extension, and several extensions are connected to the database -, it is necessary to select a specific extension.
If a version of the master configuration or extension is selected for comparison from the repository, then you should select the version of the repository to be compared with.
If the compared configurations are not descendants of each other and therefore they cannot be matched by internal IDs, the Set matching by object names check box should be selected. Click OK to start file comparison. Designer will compare the specified configurations and display the result of the comparison in the Compare Configurations window. Techniques used in this window are the same as described in the "Compare and Merge Configurations" section, with the only difference that in the comparison mode you can only view the selected objects and get a report on the differences.

33.5. "Compare and Merge Configurations" window

After the comparison procedure is completed, the Merge configurations window is displayed. This window can be used to do the following:

- learn which objects in the two configurations are different;
- select the objects that require a detailed review of differences;
- call the specified objects for viewing and editing;
- specify the objects to be merged;
- to map objects;
- set the configuration merge modes (for the entire configuration or for each object);
- set the order of subordinate objects (for the entire configuration or for each object);
- generate a report on the configuration differences;
- initiate the configuration merge procedure;

The window contains a table box that consists of three columns if the configurations are not linked with each other. The first column contains a color mark indicating that the merge settings are different from the default settings. The objects with the merge settings different from the default settings are marked in green, and an object, slave objects of which have merge settings different from

the default settings, is marked in light green color. The second column contains information about the master configuration (the infobase that is opened in the Designer), -information about the loaded configuration is in the third column, and the last column is used to set the merge mode and the order of subordinate objects.

The background color of the lines is an indication of objects difference (matching status). The possible statuses are listed under the tabular field.

Below the status decryption is the Filter selection field that is used to set the view changes mode. When you open the Merge configurations window, the Show configuration differences filter is set. Selecting other modes allows you to explore differences in configurations or matches.

All operations that are available in this window, except for merging (the Run button), are located in the command panel of the form. The following commands are available:

- Search for ambiguous references to objects to be deleted - searches for ambiguous references to objects to be deleted. The results are displayed in a message box (with the possibility of positioning on the property containing the link).
- Load settings... - the window settings are reset to the default settings, and then the merge settings from the file are loaded. The downloaded file becomes the current one to run the save command and automatically save the settings.
- Add settings... - the file settings complement the current merge settings set in the window. In this case, the file to add the settings from will not be used to save settings (including automatically).
- Use Save Settings - to save to a file the selected objects, the merge properties and modes, and the merge settings for modules, forms and spreadsheet documents. Only those merge settings that are different from the default settings are saved. Saving is available if the current merge settings include settings that differ from the default settings.
- Save settings as... - is similar to the Save settings command, but offers to change the file to which the settings will be saved.
- Set default settings - the current window settings are reset and the default settings are set for the compare/merge window. These settings are applied at the opening of a dialog box.
- Merge Modules - performs merging of selected non-merged modules, which have been set one of the Merge with Priority ... modes.
- Update - update comparison tree after changing configuration objects.

- Set mode for all opens the dialog box for setting the merge mode and setting the order for all objects.

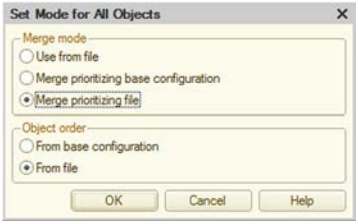


Fig. 397. Setting modes for all objects

- Use Mark by subsystems... to set the selection by subsystems of the main configuration or file configuration.
- The Comparison report is used to get a report on the comparison of two configurations with the necessary detail level.
- Settings opens the form to set up the compare/merge settings.

The Save settings automatically check box controls the automatic saving of the compare/merge window settings before the merge starts (by pressing the Run button).

Before closing the compare/merge window, you will be prompted if you want to save the settings to a file if they differ from the default settings and have not been saved.

NOTE. When the compare/merge configurations window is open, it is not recommended to perform lengthy operations on the configuration, for example, operations with repository, disabling support of the vendor configuration, etc.

33.6. Configuration Comparison Setup

33.6.1. General information

Clicking the Settings button opens the Configuration Comparison Setup dialog box.

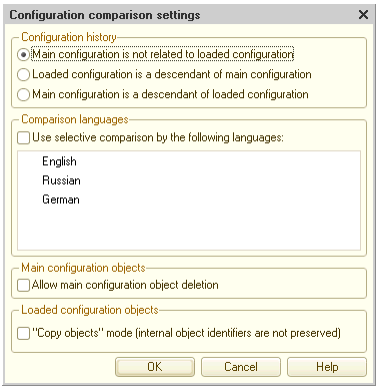


Fig. 398. Configuration Comparison Setup

In the dialog, the configuration comparison mode is selected. The default mode is comparison of various configurations (not related to each other).
If the configurations are related, you need to select the nature of the connection.
When you close the settings dialog box (clicking the OK button), a special text box is added to the Merge Configuration window, with the history of object changes shown by special icons.
A special text box is added to the Loaded configuration column, if the loaded configuration is a descendant of the master configuration; to the Master configuration column, if the master configuration is a descendant of the loaded configuration.
Also, the reference information is added to the window for description of statuses grouped in the Status group on the object history.
If the configuration uses several languages, selective comparison by languages is available. Select the Use selective comparison by languages check box and specify one or more languages to be compared.

33.6.2. Master Configuration Objects Group

When configurations are being merged, objects of the master configurations can be deleted. By default, this feature is only enabled in the vendor configuration update mode. To enable it in other modes, select the Allow master configuration objects deletion check box.

By default, deletion marks for vendor objects are set based on the following rules. If the user has modified a vendor object as compared to the previous version of vendor configuration, the object is not marked for deletion by default; if the object is identical to its previous version, it is marked for deletion. If an object has been marked for deletion (automatically or manually), clicking the Run button results in reference integrity check. If ambiguous references to a deleted object have been found, they are displayed in a separate window; however, unlike the case of ambiguous references resulting from a cancelled copying of a vendor configuration object (or any other configuration being merged), you cannot continue merging (and delete the object) in this case.

33.6.3. Loaded Configuration Objects Group

By default, upon merging of configurations, the objects unavailable in the master configuration are added to it together with IDs assigned to them in the loaded configuration.

Selecting the Object Copy Mode... check box is used to add new objects according to the rules existed in the previous platform versions: when adding objects that are not provided in the master configuration, new IDs are always generated for them.

When you press the OK button, the repeated comparisons based on the selected setting is performed.

33.7. Object Editing

When the Merge Configurations window is open in the standard mode (repository is not used and no support is available), you can view and edit objects in the master configuration.

To edit object properties, specify the required object in the column of the master configuration and correct the value in the properties palette. If the edited object is a form, template, interface or another complex object, it can be opened for

editing by selecting the Open Form (Template/Interface) command from the context menu. The master configuration objects can also be selected for editing in the Configuration window.
The loaded configuration objects are available for viewing only. The procedure for accessing the loaded configuration objects is similar to the procedure for accessing the master configuration objects.
In case any changes have been made, the Run button becomes unavailable and all objects are assigned the Undefined status (displayed in grey). Click the Refresh button to keep working in the Merge Configurations window. Designer re-compares the configurations (considering the changes made) and map objects.

33.8. Multiply development and support - editing features

In the distributed development mode you can only edit the master configuration objects locked in the repository using the Merge Configuration window.
Objects not allowed for editing can only be viewed. These objects cannot be selected for the merge (no check box is available).
A "closed lock" icon to the left of the object icon indicates that no editing is allowed. In addition, the status of the object is displayed under the table field, with a text message prohibiting editing.
The Merge Configuration window behaves similarly if the configuration is supported. If a vendor rule prohibiting change is set for a specific object, this object is available only for viewing. These objects can only be compared, and cannot be merged.

33.9. Object Mapping

When configurations are compared, the program automatically maps objects of the two configurations. Since objects are mapped based on their names, the object names can be identical in the configurations being compared. However, the objects may in fact be different. On the other hand, the objects with different names might be the same. In such cases you should map the objects manually.

To cancel mapping select an object and choose the Clear Object Mapping command from the context menu. The status of these objects changes to Undefined. Click Refresh when mapping is cleared for all such objects. Configurator will re-compare configurations.

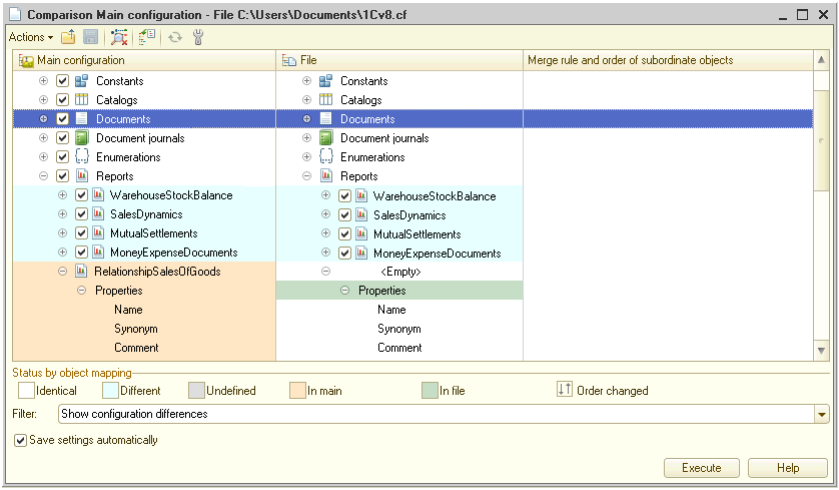


Fig. 399. Compare/Merge Configurations Main Window

For the master configuration, the objects are shown with a beige background (available only in the master configuration), and objects of the loaded configuration have the - green background (available only in the loaded configuration).
To map the object, select it and the Map Objects item from the context menu. The list of the objects that are available for mapping is displayed. The list includes objects of the same type and objects that have not yet been mapped. Select the object and press OK.

33.10. Comparison and merging features

33.10.1. General information

The merge mode can be set for each object to be merged. There are two types of modes:

- Retrieve from the loaded configuration (depending on the loaded configuration, the name of the merge mode may differ). If you select this mode,

then the configuration object is added (if it is new), or replaced (if it is modified). Moreover, the entire structure of the object, modules, descriptions, forms are migrated. Priority is given to the loaded configuration.

- Merge with priority ... - in this case, the priority of configurations is usually specified when merging. Selecting Merge with priority of master configuration means that only new items should be taken from the loaded configuration, and old ones should be kept.

Selecting the merge mode for some objects (or attributes of objects) may be unavailable, or it may have a limited choice of merge options.

33.10.2. Text documents (including source code)

Texts are merged as follows:

Text	Priority	Result
Deleted (available in master configuration, and not available in loadable configuration)	With priority master configuration	Saved in text
New (available in loadable configuration and not available in master configuration)		Added
Modified		Added as comments from the loaded configuration, while the master configuration items remain unchanged
Deleted	With priority of loaded configuration	Turns into comments
New		Added
Modified		Changes in the master configuration are turned into comments; changes in the loaded configuration - are added

Priority choice for a module with source text can be illustrated as follows.
Text in the master configuration module:

```
Procedure SpecifyStatus(Item)
    // Insert handler contents
    Message("Message from the master configuration");
EndProcedure
```

Text in the loaded configuration module:

```
Procedure SpecifyStatus(Item)
    // Insert handler contents
    Message("Message from the loaded configuration");
EndProcedure
```

Result of merging with the master configuration priority:

```
Procedure SpecifyStatus(Item)
    // Insert handler contents
//{{MRG[ < > ]
    Message("Message from the master configuration");
//{{MRG[ < > ]
//{{MRG[ < > ]
//Message("Message from the loaded configuration");
//{{MRG[ < > ]
EndProcedure
```

Result of merging with the loaded configuration priority:

```
Procedure SpecifyStatus(Item)
    // Insert handler contents
//{{MRG[ < > ]
    Message("Message from the loaded configuration");
//{{MRG[ < > ]
//{{MRG[ < > ]
//Message("Message from the master configuration");
//{{MRG[ < > ]
EndProcedure
```

If modules are merged using an external program, the result of the merge depends on the external program used.

33.10.3. Managed forms

When managed forms are merged, visualization of differences in the Merge Configurations window is not supported, but the results of the form comparison can be displayed in the text format, in a report on object comparison. When comparing forms, mapping is performed according to the following rules:

- module texts are compared by procedure names;
- form elements are compared by item type and name;
- form settings are compared by name;
- commands are compared by name;
- attributes are compared by name;

- For the command interface, comparison is performed by command type (automatic or custom) and command name.

The matching items (with the same names) in the main form are to be replaced (the properties, sizes and dimensions), if priority has been set for the loaded configuration, and to be left unchanged if priority is set for the main form. Some form element properties can be merged (e.g., a selection list, headers in different languages, etc.).
The form merge mode can be configured both for the entire form (both for the form and the form module) - for managed and ordinary forms, and for the form and its module separately - for managed forms only.

To set a single rule for a managed form and its module, you need to use the context menu of the form for which you want to set a rule. In the context menu that opens, select the required item from the submenuMerge Mode for subordinate objects.
By clicking on the gear icon in a special column, you can open the module compare (or merge) window. Depending on the Designer settings, either the tool built into the Designer, or an external program will be used for this operation. If the gear icon with a green mark is located in this column, this means that the user has configured the merge settings in this module. When you click on such a picture, the 1C:Enterprise editor is opened to refine the result of merging with an external program (the editor's capabilities in this window are equivalent to the capabilities of the built-in editor when comparing and merging modules). If you need to re-configure the merge setup in an external program, then you should use the Show differences in modules command in the context menu. If the gear icon with an exclamation mark is displayed in the column, this means that there are unresolved merge conflicts when performing a merge using an external program (only for the updating of supported configuration).
Some actions that can be performed in the compare and merge configuration window depend on the Designer settings:

- The context menu command Show differences in modules:
  - Designer Settings: Two configurations merge settings;
  - External Program Settings: Merging two files;
  - Outputs: ignored.
- The context menu command Show differences in modules irregardless the structure:
  - Designer Settings: Show differences without structure;
  - External Program Settings: Comparing two files;
  - Outputs: ignored.
- The context menu command Module merge settings:
  - Designer Settings: Two configurations merge settings;

- External Program Settings: Merging two files;
- Outputs: used as the merge result

When calling a command, the tool specified in the corresponding Designer settings will be used, and if any external program is selected as a tool - the command line specified in the corresponding external program settings will be used to perform the action. If the built-in Designer tool is used for the selected action, then editing the result of merging is unavailable when performing the comparison.

33.10.4. Comparing predefined data

When predefined data are compared, their items are mapped only by internal IDs. Upon creation of a predefined item, it is assigned a unique ID (within a metadata object). This ID remains intact during any migration of the predefined data between configurations when the configurations are merged or copied. This way the ID behavior is different from that of the corresponding ID of the metadata object. Therefore, if you create predefined data items with the same name in two configurations, these items are considered different during comparison. The only way to obtain identical predefined data in two configurations - is to merge or copy. In the event that the merge should result in the creation of several predefined elements with the same name in the resulting configuration, a numeric suffix is added to the element from the non-priority configuration (the predefined element name is changed). A diagnostic message will be displayed in the message window.

When a database configuration is updated, predefined items are mapped with existing database objects. Internal ID immutability ensures correct connection between the predefined data element and its mapped database object during migrations between configurations. Replacement of mapped objects will be made in accordance with the choice of priority configuration.

Due to the fact that mapping is performed on internal IDs, predefined data for hierarchical structures can merge, even if they are at different levels of the hierarchy. The result of the merge (the level of the hierarchy where the merge item is) depends on the configurations priority.

33.10.5. Comparing configuration extensions

The compare and merge extension operation is supported for:

- configuration extensions saved in a file (*.cfe);
- configuration extensions saved in the infobase.

Any options of comparison and merging of the master configuration and configuration extension are not supported.
There is an option to manually map objects of the compared extensions.
When comparing adopted objects, both the values of the objects and controllability property are considered. Differences in the object values and differences in controllability are included in the comparison report. At the same time, if you compare native and adopted objects -, then only the object value is checked, without controllability option.
When merging properties of objects, the following features should be considered:

- The Object Type property is displayed, but is not available for customization.
- For objects with a controllability flag set, the merge is performed as follows:
  - For the "native - adopted" pair: values are merged, controllability flag is not used.
  - For the "adopted - native" pair values are merged, controllability flag is not changed.
  - For the "adopted - adopted" pair: values are merged, controllability flag is acquired from the configuration that is being merged with.
- The saved form is considered a single object, and any actions with it depend on the merge mode:
  - The merge is performed with the priority of the configuration extension - the saved form does not change.
  - The merge is performed with the file priority - the saved form is retrieved from the file.

33.10.6. Data Types

For attribute types, including the type components in a chart of characteristic types and the components of the type being defined, you can specify both the automatic merge mode for the type components, and use the manual setting of the merge rules. When specifying a merge with priority, the merge is performed as follows:

- If at least in one configuration the attribute has a composite type specified, then the result is the merge of the composite type -the resulting type contains types from both configurations. Rules for the type qualifiers merge are described further in this section.
- If the object has a simple (not composite) type in both configurations, then the result is the type from the priority configuration.

- If in one of the type configurations consists of a type that cannot be part of a composite type, the type will be entirely taken from the priority configuration.

When merging two identical types that differ only in qualifiers, the following rules are used:

- For the `Date` type, the qualifiers from the priority configuration are used.
- For the `String` type, the following rule is used:
  - If both qualifiers have a fixed string length, then the resulting qualifier has the maximum length.
  - If both qualifiers have a variable string length, the resulting qualifier has the maximum length.
  - If one of the types is a string of unlimited length, then the resulting type is acquired from the priority configuration.
  - In other cases, the resulting qualifier is acquired from the priority configuration.
- For `Number` type:
  - If both qualifiers accuracy value coincides, and the value of the `Non-Negative` property is the same, then the resulting qualifier has the maximum length value.
  - In other cases, the resulting qualifier is acquired from the priority configuration.

When comparing and merging various types, you can control the merge manually. To do this, click the gear icon for the selected type.

Then, the types merge setup dialog opens. This dialog displays information on the types to be merged.

You can specify a filter for the displayed list (the Filter field). The result of merging for each type is displayed under the list of the merged types. To see the result of merging - you should move the cursor to the corresponding line in the dialog.

33.10.7. Exchange plans, subsystems and functional options contents

When merging configurations, the resulting contents of exchange plans, subsystems and functional options is formed by merging the contents of these merge objects.

Upon merging, the autoregistration flag of an exchange plan contents element is acquired from the priority configuration.
If you need to manually configure the merge of the contents of exchange plans, subsystems or functional options, you should click the gear button for the desired Contents property.

After clicking, the merge contents dialog opens.

You can customize the information display using the Filter field. The merge setup result for each row in the list can be seen under the list, by placing the cursor on the desired row.

33.10.8. Other objects

For objects with the property values specified in the properties palette (for example, Synonym, Comment), the dependence of the merge results on priority and availability of values is shown in the table below:

Value in default	Value in loaded	Configuration priority	Result (the value is selected from...)
Assigned	Assigned	Priority of default	Main
Assigned	Not assigned		Main
Not assigned	Assigned		Loaded
Assigned	Assigned	Priority of loaded	Loaded
Assigned	Not assigned		Main
Not assigned	Assigned		Loaded

If an object property can represent the result of a composite value of objects of two configurations (for example, the Generated based on property), the result is determined by a simple merge or selected from the loaded configuration.
If the configuration uses several languages and the setting specifies the mode of selective comparison by languages, then comparing and merging such objects is performed considering the languages specified in the language settings.
Templates are merged as follows: the resulting template consists of the master and loadable configurations templates, separated by a string with a yellow background. First, the priority configuration template is in the final template, and then - another template through the separator. The final merge should be done manually after comparison.

NOTE. The Template Type property is not shown in the Merge Configurations window. When merging, this property is merged in the same way as the Template property.

When comparing reference information, two HTML texts are compared not by character, but excluding service tags editor HTML documents.

33.11. Subordinate objects order setting

For types of objects that can have an unlimited number of subordinate objects (e.g., Attributes, Forms, Templates, Tabular sections and other subordinate objects of catalogs and documents), in some cases the order of objects in every such group is important.
If the Designer has mapped objects within a group while merging configurations, but they have different order in the configurations, you can specify the required order in the Merge Mode and Subordinate Objects Order column.
To specify the order, select any subordinate object and then in the third column select the option to set the order: Order from master configuration or Order from file.
The selected order is applied to all subordinate objects of this type.
If subordinate objects for the selected object are changed and listed in a different order, you can select only the order in the third column. You can specify the merge mode in the top-level object using the context menu. For example, there are several subordinate objects in both configurations of the `Forms` type for the `Bill` document. These forms and their order have been changed. To choose the merge mode, select a higher-level object (`Forms` or `Bill`) and choose the required merge mode in the context menu.

33.12. Viewing differences

Before merging configurations, it is useful to review the specific differences of their objects.
For this purpose, you can use the properties palette. Open the properties palette and choose the required object of the master or loaded configuration. You can open forms or templates for viewing. If they are compared with a standard configuration file which is not a distribution file, you can make corrections to the loaded configuration data in addition to viewing. If corrections have been

made, after the merge is completed, the Designer prompts to save all the changes made in the loaded configuration.
To view differences between modules, select Show Module Differences without Structure from the context menu.
Selecting this item opens the Difference Between window. If the compared texts are large, it may take several seconds before the window opens.
For selective comparison of modules, the Merge Modules mode is used (see below).
Select Show Template Differences from the template context menu to view differences between templates.
This displayes the compare window.

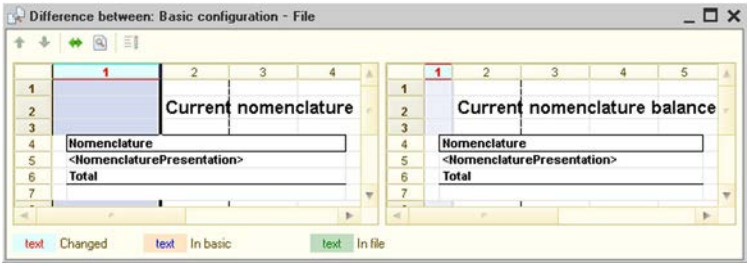


Fig. 400. Merge Templates

Spreadsheet documents are compared by cells. In the case of various cells (any properties except the Decryption parameter), then the difference is shown in color. The color legend is shown at the bottom of the compare window.
Show help topic differences context menu command shall be used to view the differences in the help topics.

The dialog opened by the command Comparison viewing parameters setting of the form command panel is intended for setting the colors of comparison results representation.

33.13. Comparing and merging modules

When merging configurations, you can configure merging of procedures and functions of any module: application module, common modules, object modules and form modules.

To do this, in the Merge configuration window, select the line corresponding to the modules being merged, and click the gear icon in the special column. If an external program to perform the module comparing is selected in the Designer settings, it is launched when you click on the gear icon. This external program defines the operations available in this case; you should familiarize yourself with them in the documentation for this external program.
The Module Merge Settings window is displayed.

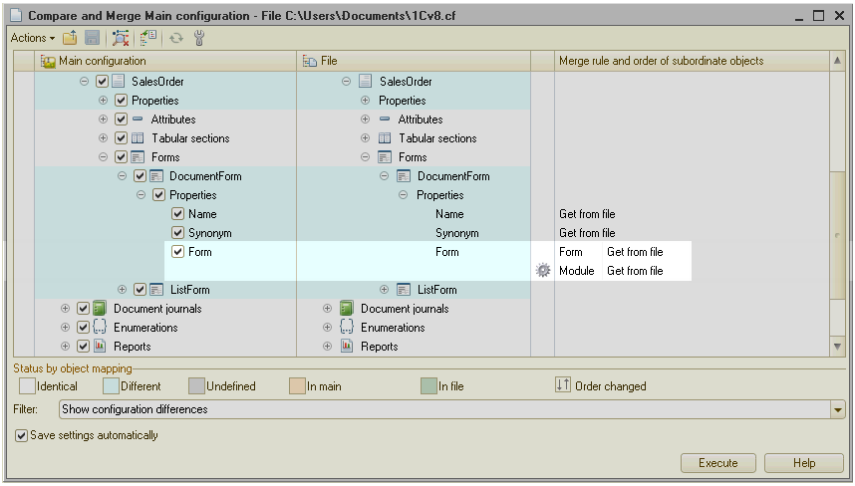


Fig. 401. Module merge settings

The top part of the window contains a table field with the content corresponding to the table field of the Merge configurations window with the only difference that the contents of the columns are generated from the names of the module parts: variable declaration section, procedures and functions section, main program section (hereinafter, the objects). Any of these sections can be missing. The result of merging under the current rules is displayed in the bottom of the form.
The table field contents corresponds to the set filter (Filter attribute)
In the first column, for objects that are not identical, mark those that you want merged. You can customize the merge settings for the selected objects.
The comparison is performed separately for each section, for mapped objects only. Mapping by name is performed automatically as the initial step.
You can clear any mapping of procedures or functions. To clear mapping, right-click the object row and select Clear object mapping in the context menu.
If in the compared modules you need to compare the objects with changed names, you should perform manual mapping of the objects. To perform manual mapping, right-click any of the objects to be mapped, and then click Map Objects.

In fig. 402, you can see that the `AddCharacteristics()` method has a different content in the main configuration and in the configuration from the file. The result of merging is displayed in the bottom of the form.

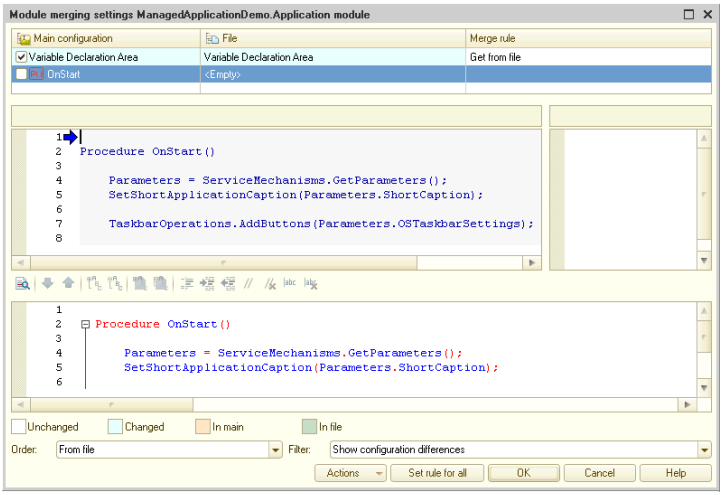


Fig. 402. Merging Modules

The text of the selected method, resulting from the merge is shown in the window bottom part. You can edit the resulting text directly while configuring the merge settings. The method text will be loaded into the main configuration, considering the changes made during the configuration process.

When working with the module merge window, to facilitate the work, some auxiliary features are available:

- The line numbers are displayed for each module (main configuration, configuration from file and the merge result).
- If the cursor points to the column with the row number, pressing the left mouse button causes the system to attempt to display the specified row in each of the three modules.
- Context help is available in the text editor (with the merge result). The context is generated by the main configuration (before merging) and the current (merged) module.
- Some commonly used commands are available in the text editor command panel with the merge result, such as working with blocks, indent change, setting and removing the comment feature, etc.

Setting the objects order in the module is determined in accordance with the value of the Order attribute.

To view the result of the module comparison without accounting structure, click the Actions - Show differences in modules without structure. The Differences Between window is displayed. Learn more about viewing differences on p. 1945.
To view the preliminary result of merging the modules, select the Actions - Display merge result.
After the mapping, setting the order and merge mode, click OK to save the module merge settings. Click Cancel to cancel-.
After completing the module merging procedure, the Custom settings appears in the Merge Mode column and the order of subordinate objects in the Merge configurations window. If you select another mode in the list of merge modes, the customization is lost.
The modules are merged according to the customization performed only for the entire configuration.

33.14. Selecting objects and starting the merge process

The list of objects to be merged is generated by checking the box to the left of the icon of a main configuration object. Selecting or clearing check box for a top level object sets or clears a mark for all subordinate objects. A check mark indicates that these objects of two configurations will be merged.
The mark of the object takes the Undefined state, if for some of the subordinate objects the mark is set, and for the others it is cleared.
When you open the Merge configuration window, the Designer includes all valid objects in the list of merged objects.
When you click the Run button, the Designer analyzes the configuration data by the set marks. If no reasons are found to prevent the merger, it proceeds to the merge. The merged configuration is saved automatically.

Upon selective merging of objects that refer to other objects not included in the list of merged objects, the Designer displays a list of ambiguous references. An example of an ambiguous reference is the object with the attribute representing a link to another object of the loaded configuration that is not included in the list of merged objects.

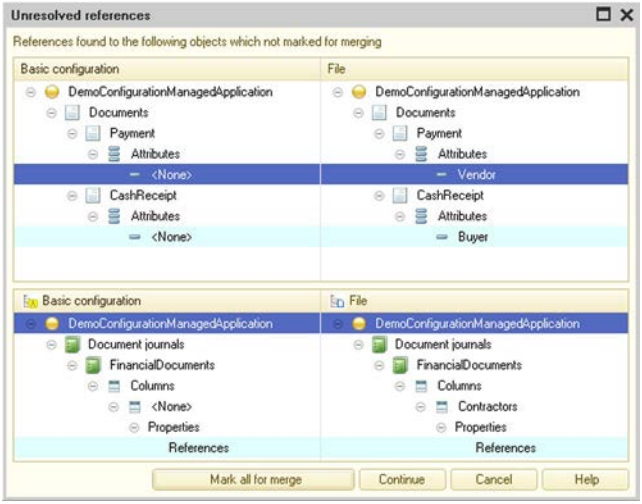


Fig. 403. Ambiguous references

The window contains two tabular fields. The upper field contains a hierarchical list of all found objects that are linked with the selected objects in the Merge Configuration window. The lower field - contains a hierarchical list of the objects that use the object specified in the top list.

Use this window to examine links between the selected objects and to decide whether it is necessary to include these objects in the list of merged objects.

Use the Mark All for Merge button to select all objects that are necessary for a full merge. When you click this button, the window closes, and all required objects in the Merge Configuration window are marked.

Pressing the Continue button results in merging of selected objects only, irregardless ambiguous references. In this case, the specialist performing the merge is responsible for the merge result.

33.15. Loading configurations from file

If the entire configuration needs to be replaced while running 1C:Enterprise, you need to load the new configuration from file.

Use the Configuration - Load Configuration from File command of the main menu to load changes.

Find and open a new configuration file (1Cv8.cf by default) in the standard file open dialog.

If the current configuration is not empty, the following message is displayed: Configuration is not empty. Loading a modified configuration will replace it. Continue? If you click Yes, it will launch the process of analyzing the differences between the old and the new configuration structures and performing necessary transformations. You will be prompted to update the database configuration.

You can also load a configuration from file to create a new configuration based on the loaded configuration.

Chapter 34. Distributed configuration development

34.1. General information

Configuration development by a team may be carried out sequentially: developers agree on modification of configuration objects, make changes and then integrate the results. This mode is associated with a risk of accidental changes to various objects; it is therefore necessary to understand all modification processes and relationships between objects. In this case, the configuration merging should be performed by a specialist who can effectively steer the direction of development effort.

This chapter explains how to reduce errors probability, increase performance and simplify the development process through distributed configuration development.

Distributed development environment is simultaneous work of a team of users (developers) on reconfiguration (modification of the configuration), where the object modification can be made only by the developer, which has previously locked the object. In this chapter, the term “configuration” means both the main infobase configuration and any configuration extension connected to the infobase which is being developed.

For the distributed development environment, a repository is created, where the Designer stores the configuration. Developers are granted access to the configuration repository: either within a local network, or using remote access through a web server. An administrator is assigned to the repository, who creates a list of users with an access to the repository. Administrative rights can also be assigned to other users.

In the distributed development environment the configuration is considered as a set of read-only objects. In order to make changes to an object the developer has to lock it first. The object can be locked only by one user at a time. The user can lock an arbitrary number of objects not locked by other users.

Techniques for working with locked configuration objects do not differ from those used in the normal mode. The user can edit the properties of the locked object as well as add and remove subordinate objects.

To add objects, it is required that the object to which they are subordinate be locked. E. g., to add a constant, you need to lock the root configuration object. To add an attribute or object form - the object itself has to be locked.
To delete an unlocked object, you should lock the object itself, as well as the object to which it is subordinate, and all its subordinate objects.
After working with locked objects, the result of their modification can be stored in the repository. On the other hand, if any of the unlocked objects have been modified, you can retrieve the updated objects immediately after adding them into the configuration repository by the author of the changes.
Results of working with the repository can be viewed in the repository history. You can open configuration versions stored in the repository to view and compare with the current configuration, database configuration, as well as various repository versions.
Configuration objects are closely interrelated. Therefore, the configuration repository maintains metadata integrity when objects are locked or stored in the repository.
In this way, the distributed development environment ensures synchronization of configuration modifications by the developer team.

34.2. Repository administration

34.2.1. Creating repositories

Configuration repository is created to share access to configuration objects. To create a repository, select Configuration - Configuration repository -Create repository. The following dialog box opens:

Fig. 404. Creating a configuration repository

Specify the repository location directory in the Repository directory field. No other configuration repository should be registered to the specified directory. Click Next.

Fig. 405. Assigning Repository Administrator

Specify the repository administrator name and access password. The repository administrator creates a list of users who have access to the configuration repository.
After pressing OK, the Designer analyzes the data, generates the objects structure and records the data in the specified directory. This process can take some time (depending on the configuration size). Upon completion of creation of the configuration repository, the Designer prompts the user to connect to the repository.

Fig. 406. Connect to repository?

If a configuration repository is registered at the specified address, the Designer displays a warning on that the repository cannot be created.
When creating a configuration repository, it should be remembered that the repository created for the configuration and the repository created to extend the configuration are incompatible with each other. You cannot connect to the configuration repository for working with the extension and vice versa.
The configuration depository is created in a format version 8.3.8 with a page size of 8,912 bytes. In order to change the repository base format or page size - use the cnvdbfl utility program. The file of the configuration repository database is named 1cv8ddb.1CD. It should be remembered that the Designer version 8.3.7

and higher cannot be connected to the configuration repository created by 1C:Enterprise version 8.3.10 or lower. In this case, it is necessary to perform configuration repository conversion.

34.2.2. Connecting to Repository

To connect to the configuration repository, select Configuration -Configuration repository- Connect to repository. Before connecting, the Designer warns that the current configuration will be replaced by a configuration from the repository and asks for confirmation of the connection.
Upon confirming, the connection dialog is displayed:

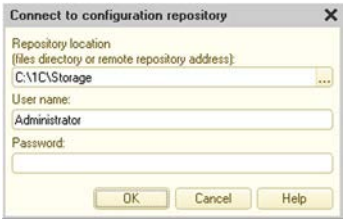


Fig. 407. Connecting to Configuration Repository

You should enter the location of the repository, as well as the username and password to access the repository, if they are set. If no one is registered with the specified user name, a warning is displayed: Authentication error! Verify whether the entered user name and password are correct. When choosing a repository location, it should be remembered that the configuration repository cannot be used for a configuration extension and vice versa.
For each user, only one configuration can be connected to this repository. Therefore, the Designer first verifies the registration of the connection for this user.
If the user name was registered, and the infobase for this user has not yet been connected to the configuration repository, then the connection is establishing. The configuration from the repository replaces the configuration of the current infobase, and information about the configuration and its location is recorded in the repository for this user.
If there is currently an open configuration associated with the specified user, the Designer displays an authentication warning.
If there is no open infobase for this user, but the location data does not match with the connection data (e.g., the configuration is in a different directory or connection is made from another workplace), then the Designer displays the following message: Configuration bound to this configuration repository already

exists for this user. Continue? Pressing Yes establishes a new connection, and configuration from the repository replaces the current configuration. New connection is established between the current configuration and the repository for this user. In this case, the previous connection will be lost (the infobase that was previously connected to the repository will no longer be able to connect to this user without reconnecting).
After completion of the connection procedure, the configuration window opens.
Not all combinations of compatibility mode of configuration repository and of 1C:Enterprise versions connecting to the repository are compatible:

	Version 8.3.2 and earlier:	**Version 8.3.3 - 8.3.5**
Version 8.3.5 or higher	Yes	Yes
Version 8.3.4	Yes	Restricted
Version 8.3.3	Yes	Restricted
Version 8.3.2 and lower:	Yes	

Compatibility modes are listed in the columns, and the lines indicate the version numbers of the Designer that attempts to connect to the configuration repository. The Yes value in the cell means that a certain version of the Designer can connect to the repository with the specified compatibility mode. Restricted value means that only the Designers versions 8.3.3 and 8.3.4 can work with the repository without restrictions. If the Designer versions 8.3.5 and 8.3.4 simultaneously connect to a repository that is in compatibility mode Version 8.3.3 - 8.3.5, the Designer version 8.3.4 cannot store modified objects in the repository.
It is also necessary to remember that if the configuration repository is created using the 1C:Enterprise platform version 8.3.10 or higher, then the Designer version 8.3.7 and lower cannot connect to it. If this is necessary - you may convert the configuration repository file to the format version 8.2.14 using the cnvdbfl utility program.

34.2.3. Connecting to repository

If the infobase is already connected to the repository, then credentials to access this repository are requested upon its opening. If after the first connection the credentials did not change, then you need to confirm the connection to the configuration repository. If the location of the configuration repository has changed (without changing the other access parameters), you should specify the new location of the repository in the connection dialog. Once connected, a configuration window opens.
If the username or password are incorrectly specified upon connecting to the repository, or you specified the username currently connected to this repository,

the following message is displayed: Authentication error! Verify whether the entered user name and password are correct.

If you specify another user when connecting to the repository, then the connection will result in the error Existing connection user is different from the current one. In this case (when changing the user, on whose behalf you are working with the repository), you should disconnect from the configuration repository and reconnect with the new user.

If an attempt is made to connect to the configuration repository for the currently open configuration, the Designer displays an error message indicating the user name.

If the location of the infobase connected to the repository has changed, a warning will be displayed Location of the infobase has changed. Continue? If the answer is positive, the new location of the infobase is saved in the repository and the connection will be established. If the answer is negative, the connection to the repository will not be established. In this case, to try again to connect to the configuration repository, use the Configure - Configuration Repository - Open Repository command.

If two infobases are connected to the same configuration repository on behalf of one repository user, then work with configuration repository will be possible only for the developer who has established the connection last. For the first developer, an error will be displayed The infobase is not associated with the configuration repository. In this case, the first developer needs to connect to the repository on behalf of another repository user.

34.2.4. Configuration window

The configuration window connected to the repository is somewhat different from the configuration window not connected to the repository:

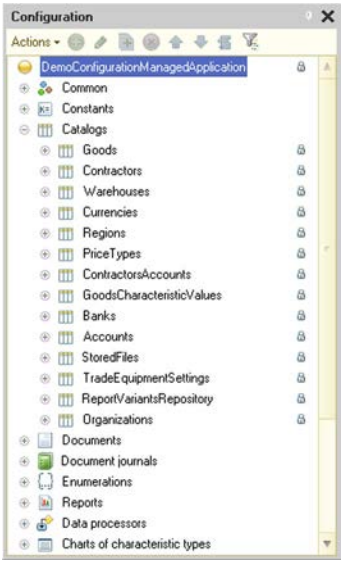


Fig. 408. Configuration connected to the repository

The status of configuration objects is displayed by special icons placed along the right border of the Configuration window. Object and subordinate object icons differ in size (they are smaller for the subordinate objects) and appearance (the subordinate object have an image of a tree branch).

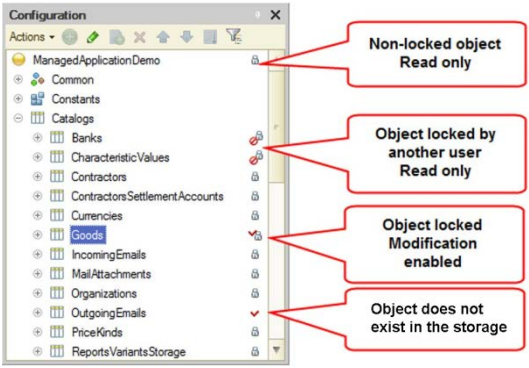


Fig. 409. Objects icons

Possible object states are depicted in fig. 409.
If an object is locked by another user (see fig. 409), then this object can only be modified by the user who locked this object. The metadata object hint that appears above the icon of the object current state in the configuration repository contains the name of the user who locked the object, the date of locking the object by the user, the name of the computer from which the user connected to the configuration repository, and the location of the information base.
Objects created by the user and not yet added to the repository, as well as objects deleted from the repository but still available in the local configuration, are indicated by a red mark.

34.2.5. Maintain a list of repository users

To create a new repository user, select the Configuration -Configuration Repository - Repository Administration menu item and on the Users tab select

Actions - Add. The window for editing user information is displayed. On the General tab, enter the user name and password to access the repository.

Fig. 410. Repository user settings

The user rights are specified in the Rights tab.

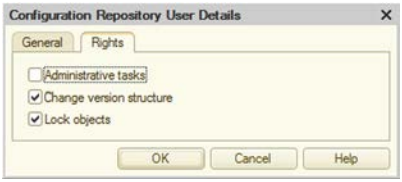


Fig. 411. Repository user rights

Administrative functions - The user has rights to create, delete and disable users, unlock, change the repository compatibility mode and perform optimization procedure.
Changing the version list - The user has the rights to perform a rollback and reduce the version history.
Lock objects - users with this right can lock and change the repository configuration. If the right is not set, the repository objects are read-only.

34.2.6. Cache versions of the repository

To speed up work with the configuration repository, two cache versions are used. One cache - is local, it is located on the side of the computer that is running the Designer connected to the repository. The other cache - is global, it is located on the computer where the repository directory is located. The cache is designed to speed up some operations on working with the repository. Versions are included in the cache as a result of the following operations:

- Connecting to the repository;
- Opening a configuration version in the repository;

- Comparing configuration versions in the repository;
- Saving the configuration version in the repository;
- When the selective comparison is performed of the following configuration objects:
  - Forms (when working through the repository server);
  - Access rights (when working with the configuration repository located on a network resource or through the repository server);
  - Command interfaces (when working with the configuration repository located on a network resource or through the repository server).

When it is needed to perform any of the above operations, the system sequentially analyzes the local and global version caches. If the search did not lead to a positive result - the required version is retrieved directly from the repository.
Location and volume of caches are managed using the repository administration dialog or configuration files.

34.2.7. Configuration repository administration

To administer the configuration repository, use a special window called up by the menu item Main menu - Configuration - Configuration Repository - Repository Administration.
A window is displayed (see fig. 412).

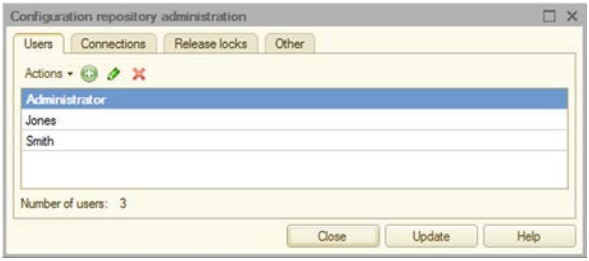


Fig. 412. Repository administration

The Users tab contains the user list - configuration developers management controls (hereinafter in this chapter - users). All users can change their own settings (username, password), except for changing rights. Bold font identifies those users who are currently connected to the repository. In fig. 412 the user Administrator

IMPORTANT! Adding, deleting and changing rights are available only to users with administrative rights.

The Connections tab displays the list of users connected to the repository.

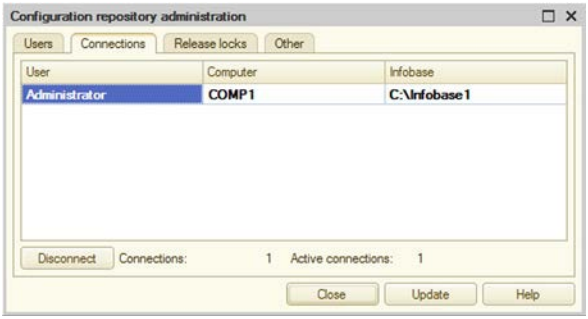


Fig. 413. Connected users

To disconnect any of the specified users that is not currently working with the repository, click Disable.

IMPORTANT! The Disable button is available to users with administrative rights.

On the Unlock tab, objects for the specified user are unlocked.

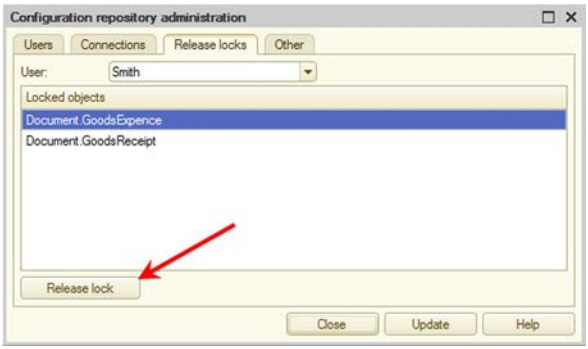


Fig. 414. Unlocking objects

At the same time, the user who locked the objects loses the ability to store changes to these objects in the repository.

IMPORTANT! This button is available to users with administrative rights.

The Other tab contains the settings that control the operation of the repository.

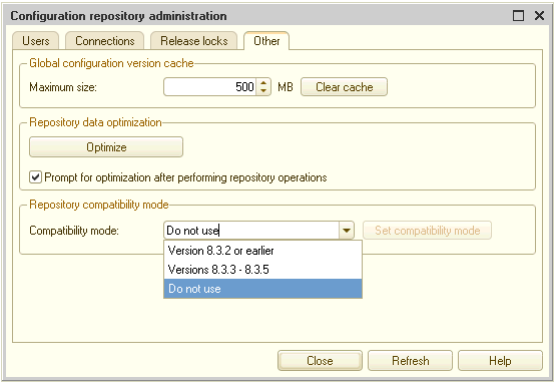


Fig. 415. Repository settings

The group of settings Global cache of configuration versions is used to set the size of the global cache of configuration versions. The cache cleaning algorithm is the same as the versions local cache cleaning algorithm. The Clear Cache button results in a full cache clear.

To control the global version cache settings, use the cfgrepo.conf file.

The group of settings Repository compatibility mode controls the repository format and the way it interacts with the repository server.

The following table shows which system versions can work with the repository with one or another compatibility mode set:

Compatibility mode	Implemented in version	Can work
Version 8.3.2 and lower:	8.3.3	Any version
Version 8.3.3 - 8.3.5	8.3.6	Version 8.3.3 and higher:
Do not use		8.3.3 and higher

If the Compatibility Mode property is set to Version 8.3.2 and lower, then any versions of 1C:Enterprise can work with the repository. If this property is set to Version 8.3.3 -8.3.5 and above (including Not to use), then after clicking the Install button, the repository will be converted to an optimized format. After optimization, working with the repository will be faster (including when using the repository server). Compatibility mode can be changed by a user with administrative rights only.

IMPORTANT! Converting repository might take some time.

If necessary, you can convert the configuration repository back to a format appropriate for use by any versions of 1C:Enterprise. To do this, set the Compatibility Mode property to Version 8.3.2 and lower and press the Install button. In addition, you may need to perform the conversion of the repository file itself. Such an operation may be required if the configuration repository was created in 1C:Enterprise version 8.3.10 and higher. For conversion, you must use the cnvdbfl utility program.

For the new configuration repository, the Compatibility Mode property is set to Not to use, and the repository database file is created in the version 8.3.8 format.

The Repository Data Optimization group allows you to optimize repository that is not in compatibility mode with version 8.3.2. As a result of optimization, the speed of work with the repository will be increased. Optimization should be performed only after the system has prompted to do this.

For a user with administrative rights, a recommendation to perform optimization may be included after an operation with the repository (adding objects, rolling back versions, merging versions). For this purpose, the Suggest optimization after the operations with the repository check box is intended on the Other tab of the repository administration dialog. In this case, after completion of the operation with the repository, the user may be shown the dialog:

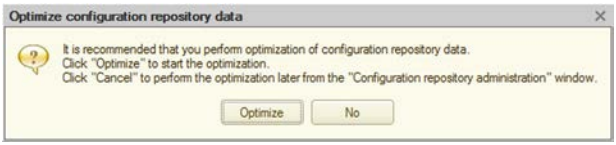


Fig. 416. Suggestion to optimize repository

If the user (repository administrator) refuses to perform optimization immediately after the recommendation appears, then optimization can be performed later. In this case, go to the repository administration dialog and click the Opti-

mize button on the Others tab. The button should be pressed if the recommendation to perform optimization is displayed to the right of it. Repository optimization might take some time.

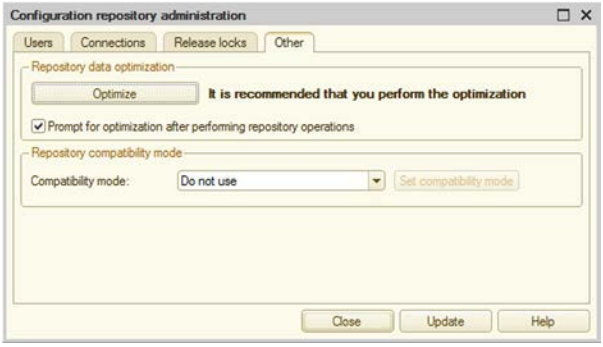


Fig. 417. Optimization is recommended

During the optimization, it is not recommended to work with repository.
Optimization can also be performed using the command line parameter of the Designer launch /ConfigurationRepositoryOptimizeData.
To configure the local repository data settings, use the dialog, which is accessible via Main menu -Configuration - Configuration repository - Local repository data usage settings.

Characteristics of the local version cache is specified in this dialog, and you can also perform optimization of the repository containing the information about development objects.
A local cache is generated when the configuration repository is located on a network drive or is operated using the repository server. The Directory property is used to specify a directory on the local file system where the cache will be located. By default, the cache is located in the directory:

- On Windows: %APPDATA%\1C\1Cv8\<Infobase UUID>\cfgrepo\cache.
- On Linux: ~/.1cv8/1C/1cv8/<Infobase UUID>/cfgrepo/cache.

The Maximum size value determines the size of the local cache in megabytes. The default value -is 500 Mb. If during operation the local cache size exceeds the set size, the oldest versions of 50% of the cache size will be automatically deleted. The Clear Cache button causes complete cleaning of the local cache.
To control the local version cache settings, use the cfgrepo.conf file.

34.2.8. Repository backup

Depending on the compatibility mode of the repository, the backup is performed differently.

IMPORTANT! During the execution of backup configuration repository, there should be no connection to it.

To perform a backup, you must copy the following files:

- The Compatibility Mode property is equal to Version 8.3.2 and lower-, copy the 1Cv8ddb.1CD file only.
- The Compatibility Mode property is not equal to Version 8.3.2 and lower - you need to copy the file 1Cv8ddb.1CD and the entire data **directory**, located in the same directory as the 1Cv8ddb.1CD file.

34.3. Configuration repository operations

It is convenient to use the Configuration repository window to perform locking, retrieving and adding objects to the repository, as well as other actions with the repository objects. To open it, select Configuration - Configuration repository - Repository

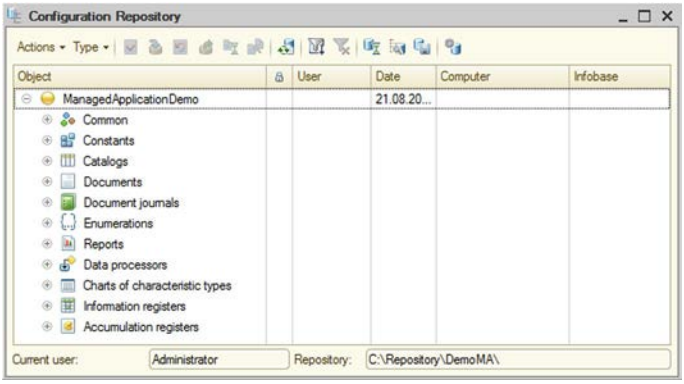


Fig. 418. Configuration repositories.

The Configuration repository window consists of a toolbar and a table box, the first column of which shows the hierarchical structure of repository objects or a list of objects, depending on the view mode (View menu).

If the display mode for deleted objects is set (View submenu), then the deleted objects are shown in a ligher font in the list.
In the second column, the state of the locked objects is indicated by icons. The icon with a red check mark indicates the objects locked by the current user. The User column specifies the user who locked the object. The Date column shows the lock date (if the object is currently locked) or the date the object was last updated. The name of the user's computer is shown in the Computer column, and the name of the local infobase directory on the user's computer - in the Infobase column.

IMPORTANT! New objects created by the user (not yet stored in the repository) are not displayed in the repository window.

34.3.1. Repository filter

You can customize the objects displaying in the Object column of the Configuration repository window. The filter is called by the Actions - Filter menu item. This opens the following window:

Fig. 419. Filtering repository objects

The names of controls are self-explaining.
It is also possible to quickly filter only those objects that are locked in the repository by the current user.

To do this, use the Actions - Locked Objects command or press a special button on the command panel of the Configuration window. The button is displayed only if the configuration is connected to the configuration repository. The command reexecution disables the filtering. The filtering works in conjunction with the filtering by subsystems and search the configuration tree.

34.3.2. Locking repository objects

Locking a repository object is intended for the editing purpose. The locked object becomes inaccessible for other users (read-only mode and retrieving the modified object for use are available).

To lock an object, select it in the Configuration repository window (multiple selection is possible) and run the Actions -Lock in repository... command

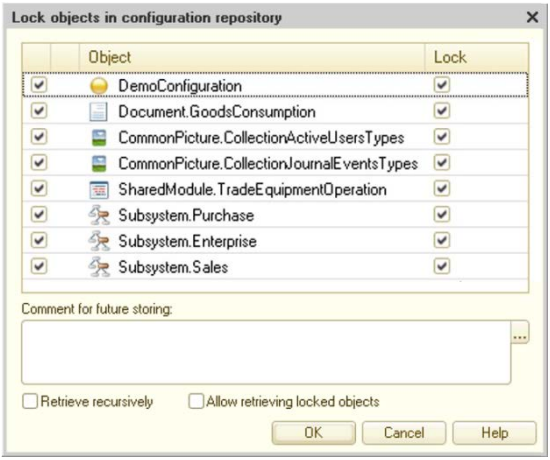


Fig. 420. Locking objects in repository

If the object is linked with other objects, these objects will also be included in the list. In this case, the selected object is shown in bold. Locking the selected object is possible only if these objects are retrieved from the repository.

The list also includes those objects that are not directly linked with the selected object, but are indirectly related.

To retrieve these objects, check the box in the first column of the list. If you also want to lock the objects, check the box in the Lock column. If not all objects required for acquisition are marked for acquisition, the Designer displays a message To perform the operation, the following objects are required and lists the required objects. In this case, the locking is not made.

If objects contain subordinate objects which you want to lock, select the Run recursively check box.

If you want to retrieve data of the locked objects, select the Allow to retrieve locked check box.

IMPORTANT! If the Allow to retrieve locked check box is selected, changes made to the locked objects and objects retrieved over again will be lost.

After locking, the specified objects will be marked with a special icon.
Upon locking, you can add a comment in advance, which can be used when adding an object to the repository. To do this, add text to the Comment for subsequent posting field or select a comment from the list of previously added comments (button ... in the right part of the field). This list contains the last 20 comments that were added when locking or placing objects.
To add new objects to the configuration, you need to lock the parent object. For example, to create a new directory, you need to lock the root configuration object; to create a new form or directory attribute - lock the directory object itself. After locking the parent object, you can create subordinate objects. Then all new and locked parent objects need to be added to the repository.

34.3.3. Storing objects in repository

The modified object can be stored in the repository, so other users are able to update this object in their configurations. To do this, run the Actions - Store in repository command. The following dialog box opens:

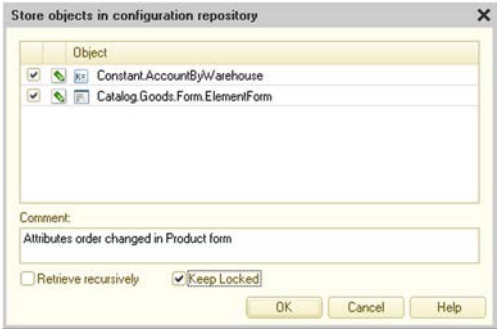


Fig. 421. Storing objects to the repository

You need to check the boxes for those objects that will be stored in the repository. To describe the changes made, add them into the Comment multiline text box. If a comment was added when locking an object (objects) in the repository or was previously used when adding an object into the repository, this comment can be selected for reuse from the list of recent comments. To open the list, click the

"..." button located to the right of the Comment: field. This list contains the last 20 comments that were added when adding or locking objects.

If objects contain subordinate objects which you want to add into the repository, select the Run recursively check box.
If you want the specified objects to remain locked, select the Keep locked check box.
If objects attached to other configuration objects are added to the repository, a full list of such objects is displayed in the dialog.
In order to add new objects into the repository, select an object in the Configuration window and select Add into repository... in the context menu. The window for adding objects into the repository is displayed.
In addition to this object, other objects linked with this object may be included in the list. In the list of objects, the selected object is indicated in bold. You can not add an object into the repository without adding (updating) these objects.

34.3.4. Retrieving objects from repository

In order to retrieve objects modified by other users, specify the required objects in the list of repository objects and run the Actions - Get from repository command. The following dialog box opens:

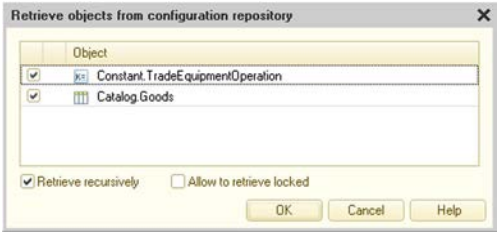


Fig. 422. Retrieving objects from repository

In the rows of the list of objects you need to check the boxes for those objects that have to be retrieved from the repository.
If the objects contain subordinate objects and you need to retrieve them, select the Retrieve recursively check box.

If the selected object is locked already, checking the Retrieve locked box results in the loss of changes made to this object. In this case, the object data is restored with data from the repository.

34.3.5. Unlocking objects

If you do not want to save the changes made to the repository object or you need to provide this object for modification to another user, you can release this object. To do this, run the Actions - Unlock in repository command.

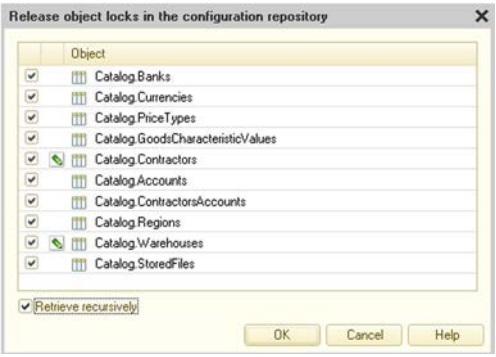


Fig. 423. Unlock objects

In the second column of the list, an icon shows the difference of the object in the configuration and repository. If there is no icon, then the objects do match.
If objects contain subordinate objects and you want to unlock them, select the Run recursively check box.
After unlocking, the objects will be retrieved from the repository.

34.3.6. Updating status of repository objects

If you need to update information about objects stored in the repository, run the Update Status command. This command updates the information on the locking or unlocking, as well as emergence of new objects throughout the repository.

34.3.7. Comparing repository with current configuration

At any time, you can compare the current configuration with the repository. To do this, run the Configuration - Configuration repository - Compare / merge configuration with the repository command. A comparison window is displayed.

34.3.8. Comparing a repository object with a current configuration object

Sometimes you need to make a comparison of a specific object in the repository and in the current configuration. To do this, in the configuration Repository window, select an object and run the Actions - Compare with configuration object command.

The query is displayed: Show comparison of configurations? If the answer is positive, the configurations comparison window is displayed, and configurations are compared; row with the selected object is activated in the comparison window, and the report settings dialog is displayed. If the answer is negative, the configuration comparison window is not displayed and no comparison is made.

If compatibility mode of the repository with version 8.3.2 is disabled, then you can perform selective comparison of two configuration objects without comparing all configurations. In this case, only those objects versions are available for selective comparison that were added in the repository using 1C:Enterprise version 8.2.9 and higher. For selective comparison, use the Perform selective comparison with repository object context menu command, Perform selective comparison with configuration object, or Perform selective comparison of the object versions, depending on where you want to make the comparison: configuration tree, repository tree, or the repository history window. When the selective comparison is chosen from the configuration tree, the comparison with the object latest version located in the configuration repository will be performed. If the selective comparison is unavailable - the platform will report this using a diagnostic message.

Selective comparison is available for the following configuration objects: module, template, picture, WS-link, schedule, register aggregate objects, exchange plan content, style, start page work area, route map, form (excluding the form module), form module, reference information, predefined elements, command interface fragment, role rights.

34.3.9. Saving repository as configuration

To save the repository as a configuration, use the command Configuration - Configuration Repository - Save repository configuration to file.

34.3.10. Disconnecting from repository

To disconnect from the repository, select the command Configuration - Configuration repository - Disconnect from repository.

34.3.11. Repository history

The distributed configuration development system maintains a history of configuration objects data changes. The results of adding objects are logged in the history.
You can view each object change and compare it with the current state, compare configuration versions, as well as retrieve a configuration of the specified version from the repository.
To open the repository history window, select Configuration - Repository - Repository History item.

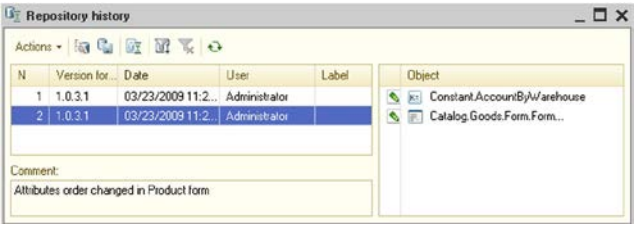


Fig. 424. Repository history

Chronological list of configuration versions is displayed In the table field located in the left part of the window.
Below the list of versions there is a field to display comments on the selected version.
The list of objects that have been changed, added or deleted in the selected version appears in the table field located in the right part of the window.

Multiple selections are allowed in the version list. In this case, comments of all selected versions are shown in the Comment field, and a list of all objects in these versions - in the list of objects.
You can open and view each object from the list in the Repository History window, view change history for this object and compare it with the current configuration object. These actions are performed using the context menu of the list of objects.

34.3.11.1. Working with the repository version

Each version of the configuration repository can be opened for viewing, loaded in place of the current version, compared with the current configuration version or saved to disk. You can also compare the version of the configuration repository with the configuration stored in the file. All the actions are performed using the Actions menu in the Repository History window.
You can use a special report to display information on the repository history. To create the report, select Actions - Repository History Report in the Repository History window. The report settings window will be displayed.

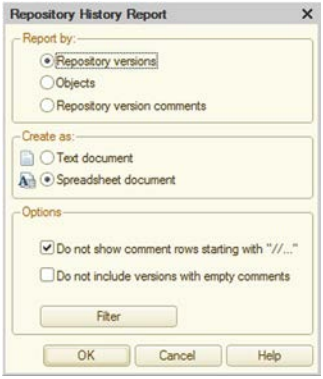


Fig. 425. Repository History Report Settings

Select the type of grouping information in the report in the Report by group. In each group, the information is sorted alphabetically.
To group information by version numbers, select By Repository versions. To group information by objects, select By Development Objects. To group information by comments in the repository history, select By Repository version comments.
Select the report view type (text or spreadsheet document) in the Show as group.

Additional report settings can be specified in the Options group.
To exclude comment lines starting with "//", select the appropriate check box.
To specify objects for which you want to create a report, and the date range, click the Filter button. In the window that opens, set the filter criteria.
Click OK to start the report generation process.

34.3.11.2. Rollback to Previous Versions

Should already published versions in the configuration repository become unnecessary, you can roll back to the repository version that you need. To do this, in the list of versions select the one to which you want to roll back and select Actions - Rollback to Version. The following warning is displayed: Information about the rolled-back versions will be permanently deleted. Continue? If the answer is positive, a rollback is performed, and all versions stored in the configuration repository after the specified one are deleted.
If a very large number of versions (several thousand) is selected for rollback, then while returning to the previous version a shortage of RAM error may occur and the operation will be interrupted. In this case, it is recommended to perform a rollback operation in smaller portions of several hundred versions.

34.3.11.3. Deleting versions

Should early configuration versions become unnecessary, they can be deleted from the repository. To do this, in the list of versions select the one, up to which you want to delete versions and select Actions - Delete previous versions. The following warning is displayed: This operation will permanently delete information about previous versions. Continue? If the answer is positive, all versions stored in the configuration repository before the specified one are deleted.
If a very large number of versions (several thousand) is selected for rollback, then while deleting versions a shortage of RAM error may occur and the operation will be interrupted. In this case, it is recommended to delete versions in smaller portions of several hundred versions.

34.3.11.4. Merging Versions

The user can merge several consecutive configuration repository versions. To do this, select the versions to be merged in the Repository History window (as you select them, a summary statement combining all comments for the selected versions appears in the Comment section, and all objects on which any actions were performed in these versions are displayed in the list of objects) and select Actions - Merge Versions.
As a result, a single version comprising summary changes of the specified versions is created in the repository history.

34.3.11.5. Tagging Versions

Each configuration version can be tagged in the Repository History window. To do this, select the required version and select Change in the context menu. This opens the following dialog box.

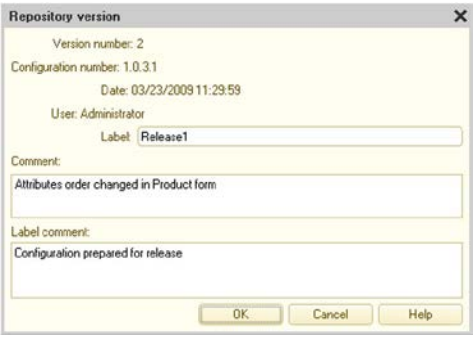


Fig. 426. Repository version

You need to add the tag value. As you enter the first character, the Tag Comment field becomes available with the text describing the tag. Tags primarily serve to mark completion of certain logical stages or actions (e.g., a new configuration version release, preparation for distribution, etc.).

34.3.11.6. Filters

The dialog box for filtering the Repository History list is different from the standard one.

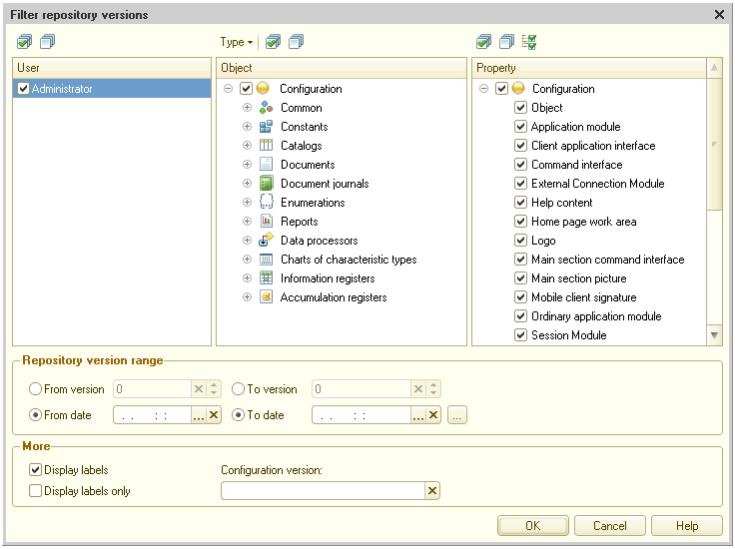


Fig. 427. Versions filtering

Version can be filtered within a time range, range of repository versions (or combination of version and date), by a specified list of repository users who have added versions, by specific configuration objects and by a specific configuration version. You can also set filtering by specific properties of configuration objects. To do this, use the property tree in the filtering settings window. Thus, in the example in fig. 427, for all selected catalogs (`Goods`, `Currencies`, `CounterpartsSettlementAccounts` in the Object tree) only those versions will be displayed where these catalogs have changed the modules of managers and the modules of objects (in the Property tree). To ease selecting properties, there is a button Mark all properties of selected objects in the command panel of the Property tree. This button selects all properties for configuration objects that are marked in the object tree. The ability to filter history is implemented on the following properties of configuration objects: module, template, picture, WS-link, job schedule, turnover accumulation register aggregates, exchange plan content, style, start page work area, route map, form, reference information, predefined elements, command interface fragment, role rights.

When you select the Object element in the property tree, those versions with the changed properties of the corresponding configuration object, for which selective comparison is not available will be logged in history.
You can also specify whether you want to include tagged versions in the filter result and whether you want to show tagged versions only.
For each selected version in the version list, the left side of the Repository History window contains a list of objects that have been added, changed, or deleted. This list is always complete for the given version (does not depend on the filtering conditions). If the specific configuration objects (not the entire configuration) are indicated in the filtering conditions, then the listed objects that satisfy the filtering condition are shown with a different background color.

34.3.11.7. Comparing repository versions

Sometimes you may need to compare configuration repository versions. To do this, select two versions to be compared in the Repository History window, (hold down the Ctrl key) and run the Actions - Compare configuration versions command or Actions - Perform selective comparison of object versions. Learn more about selective comparison on p. 1797
The specified configuration versions are compared with each other, and the configurations comparison window is displayed.

34.3.12. Object history

To view the history of an object changes in the list of repository objects or in the Configuration window, select the object and select Actions - Object history or use the menu Actions - Custom object history. Each object has its own properties set on which you can view the history.
After selecting the type of view, the history filter dialog is displayed.

Fig. 428. Filtering object history

You must specify the desired filter parameters and click OK.

The object history window is displayed.

Fig. 429. Object history

Each object change can be viewed and compared with the current state, and the versions can be compared (to do this, select two versions in the list).
You can view comments and tags for each object version similarly to configuration versions (see above), and retrieve the specified configuration version from the repository.
You can filter the object versions list. The filter is applied similarly to the repository history filter described above. In this case, in the filter versions window the current object and the corresponding properties are selected: all properties, if you have selected the object history view, or the specified properties at the custom history view.
If you head over to the repository history window, select Repository History from the Go To submenu of the object history window. If you need to open the repository history window and set a filter by the current object, select Repository History by Object.

34.3.13. Actions with Configuration Repository without Connection

Some actions can be performed without connection.
If the configuration is not connected to a repository, you should establish the connection with the repository first. To do this, select any available item from

the Configuration - Configuration Repository group. The connect-to-repository dialog is displayed.

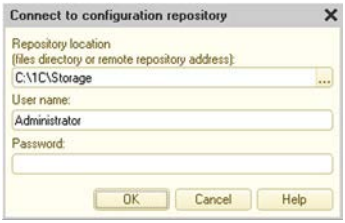


Fig. 430. Connecting to Repository

Specify the connection credentials and click OK.
When you select any of the other available menu items after the connection is established, the connection dialog is not shown again.
In connected mode, any user actions related to viewing repository data, comparing objects and configurations, as well as full repository administration (for users with administrative rights) are available. Actions related to retrieving objects from the repository are not available.
In connected mode, users with administrative rights can perform any actions related to the repository history.

34.4. Working remotely with configuration repository

34.4.1. General information

Remote work with the configuration repository expands capabilities of distributed configuration development:

- Configuration repository can be located on a computer running under Windows or Linux.
- Configuration repository can be accessed by both local network users (over TCP/IP protocol) and Internet users (over HTTP).

For application developers, working remotely with the configuration repository is practically no different from working with the repository located on a shared network resource, with the exception of some features described on p. 1818

NOTE. Remote work with repository is mainly adapted for high-bandwidth connections. It should be considered also that, when working remotely, the operations with a large data exchange volume, e. g., retrieving the full version of the repository, may take significantly longer time.

34.4.2. Architecture Overview

34.4.2.1. General information

Remote work with the configuration repository is supported by a special application - **configuration repository server**. Computer running configuration repository server application (crserver) is also called the **configuration repository server**. Depending on the access protocol (TCP or HTTP), various patterns may be used for remote communication between the client application and configuration repository. At the same time, concurrent work with configuration repository is supported, both remote and local – as with a shared file resource.

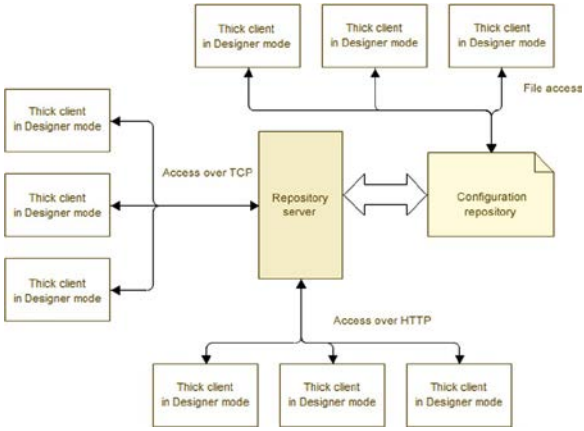


Fig. 431. Remote Repository

34.4.2.2. General operational scheme with configuration repository

In general, the “1C:Enterprise” system provides simultaneous work with the configuration repository using all access protocols to the repository: file access, TCP protocol and HTTP protocol.

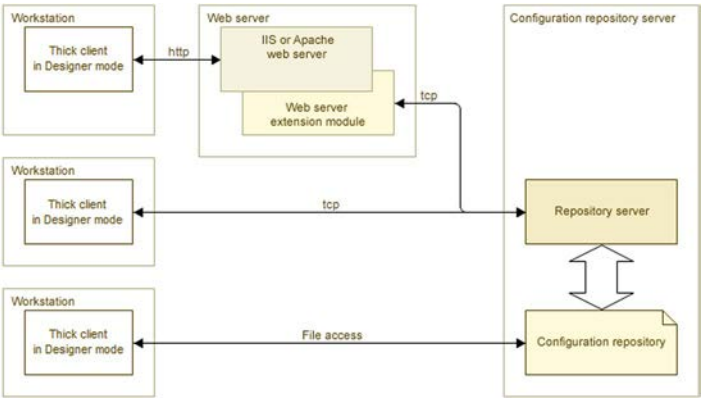


Fig. 432. Interaction with Remote Repository

At that, the clients can work with different configuration repositories using the same configuration repository server. The only restriction is that -all the repositories must be located in the same root directory.

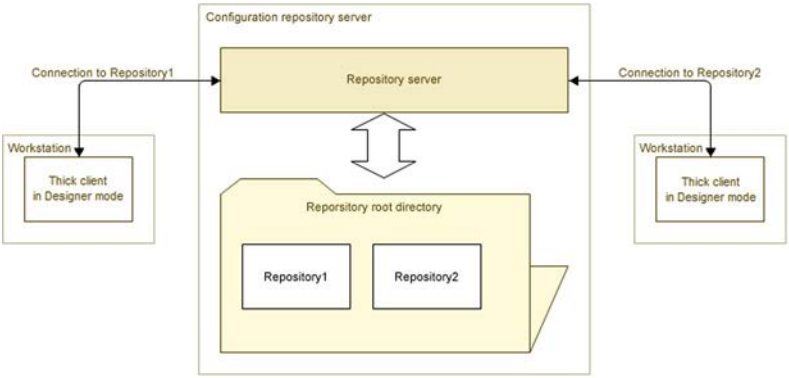


Fig. 433. Location of various repository files

34.4.2.3. Interaction using the TCP protocol

When the TCP protocol is used, the client application interacts with the configuration repository server, which in turn interacts directly with the configuration repository itself.

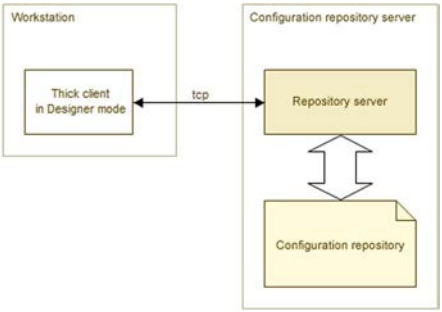


Fig. 434. Interaction using the TCP protocol

34.4.2.4. Interaction using the HTTP protocol

When the HTTP protocol is used, the interaction takes place on the following chain: client application - web server - configuration repository server - configuration repository.

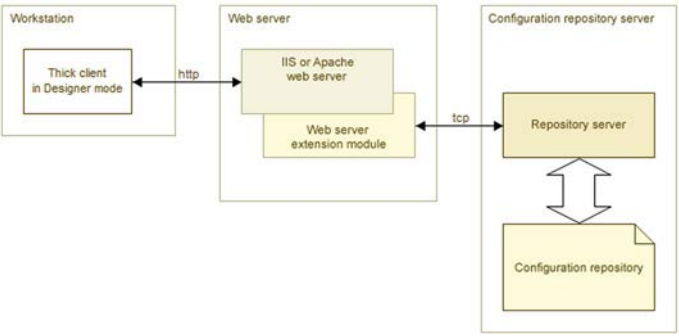


Fig. 435. Interaction using the HTTP protocol

34.4.3. Setting up the repository server

After the installation of the configuration repository server, it is necessary to provide the user, on whose behalf the configuration repository server is running with the write and read access to the directory that contains the configuration repository (1Cv8DDB.1CD file and data directory).

34.4.4. Configuring a web server to work with the repository

34.4.4.1. General information

If you plan to work remotely with the repository using the HTTP protocol, then in addition to configuring the configuration repository server, you must configure the web server to work with the 1C:Enterprise system and the configuration repository server. Configuring consists of the following actions:

- Register the web server extension module;
- Register a virtual application on the web server;
- Publish the configuration file to work with the repository configuration server.

NOTE. Support for **ISAPI extensions must** be installed on the web server.

The publication of the configuration file for working with the configuration repository server comes to placing a file with 1ccr extension in the virtual web server directory. For example, such a configuration file may have a name as repository.1ccr and the following content:

```
<?xml version="1.0" encoding="UTF 8"?>
<storage connectString="tcp://RepServ"/>
```

Administrator rights are required to configure a web server.

34.4.4.2. For Microsoft Internet Information Services web server

This section contains settings for the Microsoft Internet Information Services web server (IIS), depending on the version of the web server.

34.4.4.2.1. For IIS versions 5.1, 6.0

You must create a virtual directory. To do this, use the Virtual Directory Wizard from the context menu of Default Web Site (Internet Information Services

<Web Server Host Name> - Web Sites - Default Web Site-). In the context menu, select Create - Virtual Directory ....

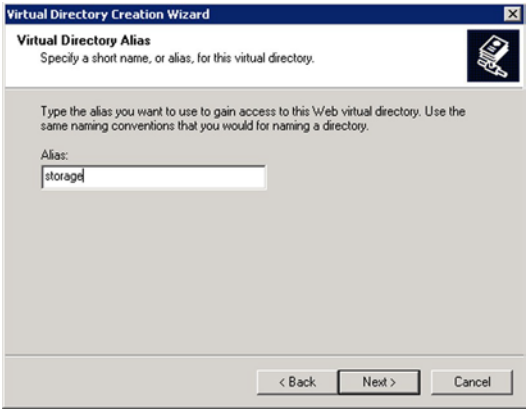


Fig. 436. Set Virtual Directory Name

The name specified in the Alias field will be used to generate a URL to access the repository. Then you need to specify the path to the folder containing the files of the directory you create.

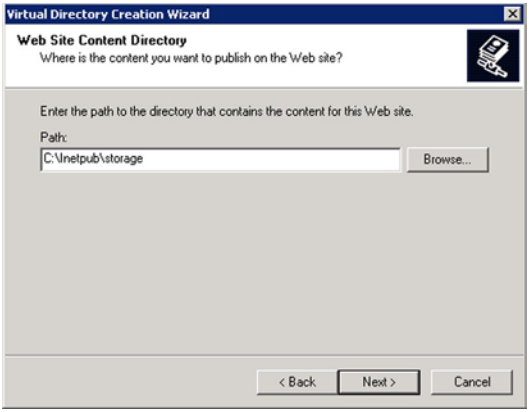


Fig. 437. Specifying Directory

As a next step, you need to specify which actions you want to allow when working with the directory. You must specify if there is a possibility to read and run

(for example, ISAPI or CGI applications). The rest of the rights should be disabled.

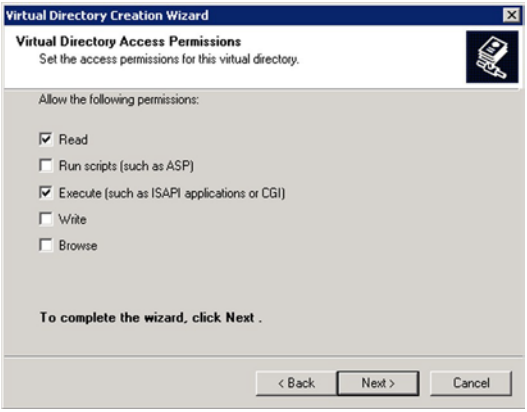


Fig. 438. Setting up permissions

яAs a next step of the wizard, you must click Finish. This completes the creation of the virtual directory. The next step is to configure the created directory and register the web server extension for it. To do this, open the properties of the created virtual directory and in the opened dialog click the Settings button.

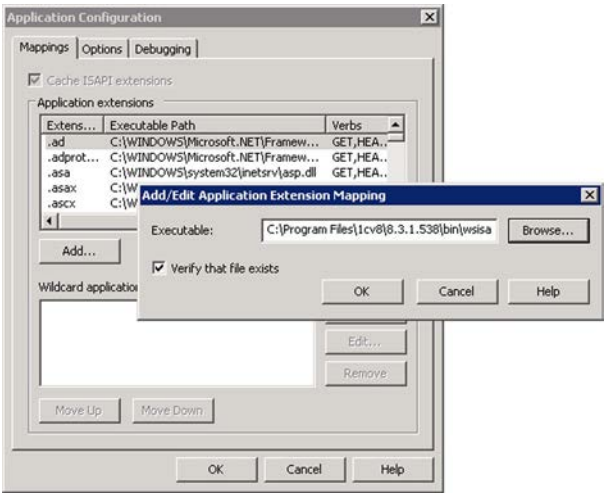


Fig. 439. Registering web server extension

The wsisapi.dll dynamic library located in the bin directory of the specific version of 1C:Enterprise is an extension of the IIS web server.

TIP. To register a web server extension, use either a short path ("8.3 format") or install "1C:Enterprise" in a directory that does not contain spaces and directory names longer than 8 characters.

As an extension (the field Extension: in the dialog forfig. 439) you can specify both a specific extension (.1ccr) and the mask of all files (*).
The final step is to restart the web server.

34.4.4.2.2. For IIS versions 7.0, 7.5

You need to create a virtual application. To do this, use the Virtual Application Wizard from the context menu of Default Web Site (<Computer name with web server>-sites - Default Web Site). In the context menu, select Add application ….

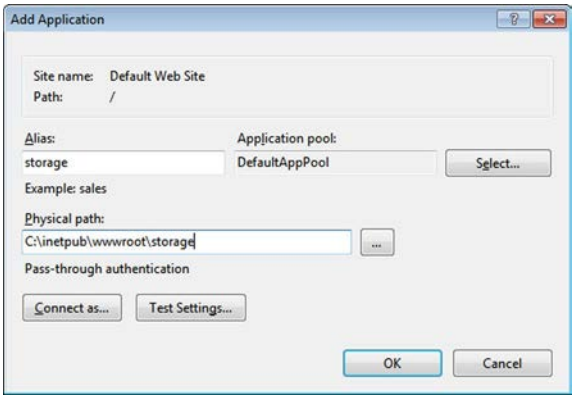


Fig. 440. Creating a virtual application

The name specified in the Alias field will be used to generate a URL to access the repository.

Then you need to add a handler for the files .1ccr. To do this, select the command <Computer name with web server> sites-Default Web Site- -storage -Handlers Mappings - Add script mapping....

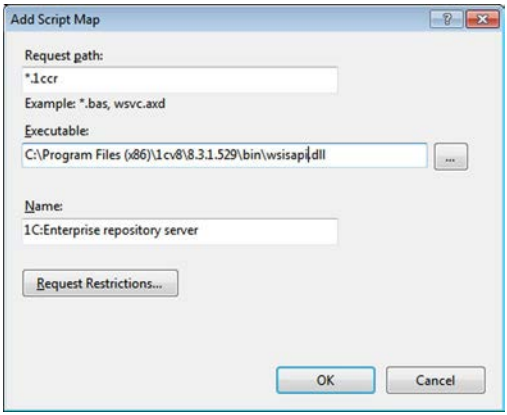


Fig. 441. Adding a handler

The wsisapi.dll dynamic library located in the bin directory of the specific version of 1C:Enterprise is an extension of the IIS web server.
As an extension (the field Query path: in the dialog forfig. 441) you can specify both a specific extension (.1ccr) and the mask of all files (*).
The final step is to restart the web server.

34.4.4.3. For the Apache web server

In order to configure the Apache web server to work with the configuration repository, you must perform the following steps:

- Enable loading of the web server extension module in the configuration file of the web server (httpd.conf). Depending on the web server version and the operating system, extension modules may have different names:
  - On Windows OS:
    - On Apache 2.0: wsapch2.dll.
    - On Apache 2.2: wsap22.dll.
  - On Linux:
    - On Apache 2.0: wsapch2.so.
    - On Apache 2.2: wsap22.so.

  The web server extension module has the identifier `_1cws_module`. To load a module, you must add the following line to the configuration file: `LoadModule _1cws_module <path to the web server extension module>`.

Example on Windows OS:

```
LoadModule _1cws_module "C:\Program
Files\1cv8\8.3.1.100\bin\wsap22.dll"
```

Example on Linux:

```
LoadModule _1cws_module /opt/1C/v8.2/i386/wsap22.so
```

- Create a virtual directory on the web server through which the configuration repository will be accessed and register a file handler for the virtual application directory. Identifier of the handler being registered: `1c-application`.

Example of a virtual directory for Windows OS:

```
Alias "/repos" "C:\www\repos\"
<Directory "C:\www\repos\">
  AllowOverride None
  Options None
  Order allow,deny
  Allow from all
  SetHandler 1c application
</Directory>
```

Example of a virtual directory for Linux OS:

```
Alias "/repos" "/var/1C/www/repos"
<Directory "/var/1C/www/repos">
  AllowOverride None
  Options None
  Order allow,deny
  Allow from all
  SetHandler 1c application
</Directory>
```

- Add the `*.1ccr` file specifying the address of the configuration repository server (in the TCP scheme) in the folder with the virtual directory files.

Example of .1ccr file:

```
<?xml version="1.0" encoding="UTF 8"?>
<repository connectString="tcp://192.168.0.12"/>
```

- Restart the web server.

34.4.5. Start the repository server

With Windows operating system, the configuration repository server can be started as an application or installed as a service.

With Linux operating system, the configuration repository server can be started as a process or as a daemon.

34.4.5.1. On Windows

The repository server is started from the command line:

```
crserver.exe  instsrvc |  rmsrvc  usr <user>  pwd <password>
   start | stop
   port <port>  d <directory>
```

In case of installation as a service, the repository server is named 1C:Enterprise 8 Configuration Repository Server.

IMPORTANT! The name and value of the parameter must be separated by a space character.

-start

Start the repository server service.

-stop

Stop the repository server service.

-instsrvc

Register the repository server as a service.

-rmsrvc

Unregistering the repository server as a service.

-usr

Name of the user on whose behalf the service is registered. This user must have the right Log on as a service (Log on as a service). In addition, he/she must have permissions to read the directory of executable files of the corresponding version of the 1C:Enterprise system and full rights to the root directory of the configuration repository (directory %APPDATA%\1C\1Cv8\ or the directory specified in -d parameter).

-pwd

Password of the user on whose behalf the service is registered.

-port

The working port of the repository server. The default port is 1542.

-d

The root directory for configuration repositories. The default directory is %APPDATA%\1C\1Cv8\.

34.4.5.2. On Linux

The repository server is started from the command line:

```
./crserver  daemon  port <port>  d <directory>
```

IMPORTANT! The name and value of the parameter must be separated by a space character.

-daemon

Start the server as a daemon.

-port

The working port of the repository server. The default port is 1542.

-d

The root directory for configuration repositories. The default directory is /home/usr1cv8/.1cv8 или ~/.1cv8.

When working with the repository server, an error <FullFileName> too many open files may occur. In this case, you need to increase the maximum number of opened files for the process using the /etc/security/limits.conf file (you need root privileges (root)). For the user, on behalf of which the repository server is started, change the value of the nofile parameter. The value is chosen empirically and depends on the size of the configuration and the number of users working with the repository. It is recommended to use 100,000 as the initial value. If the error persists, then the value of the parameter nofile should be increased.

34.4.6. Connecting to the Configuration Repository Server

Connecting a client to the configuration repository when working remotely is performed in the same way as when working with repository located on a shared network resource. The only difference is in the way of specifying the repository catalog. Depending on the protocol used, the repository directory can be specified in two ways.

For TCP protocol:

```
tcp: //<repository server name>/<relative repository path>
```

For HTTP protocol:

```
<description file address>/<relative repository path>
```

Example:

```
tcp://RepServ/MyConfRep
```

Or:

```
http://www.MyCompany.ru/VirtualFolder/storage.1ccr/MyConfRep
```

- RepServ- name of the configuration repository server;
- http://www.MyCompany.ru - address of the web server where the Web-service description file is published;
- http://www.MyCompany.ru/VirtualFolder/storage.1ccr - full address of the description file see in the HTTP scheme;
- MyConfRep - relative path to the configuration repository.

The relative path is specified from the default directory (%APPDATA%\1C\1Cv8\), if the directory was not specified explicitly, or from the directory specified in -d parameter when starting the repository server. If the relative path to the repository is not specified, a connection to the default repository will be made (maincr directory).

When working with the configuration repository server, the repository server and the Designer versions should match. In case of a mismatch between the Designer and configuration repository server versions, the work with repository is impossible.

34.4.7. Creating a configuration repository

When you first start the configuration repository server, a default directory will be created on your computer to host the configuration repository. This directory is named maincr. If the -d key was not specified when starting the server, then the configuration repository will be located in the %APPDATA%\1C\1Cv8\maincrdirectory by default.

To create a new configuration repository, you need to create the repository from the client application in the Designer mode, specifying the required new directory as the repository directory.

To ensure maximum performance, it is recommended to place the configuration repository root directory on the configuration repository server computer. The

1C:Enterprise supports the placement of repository directories on network resources available to the repository server, but this option may result in reduced performance.

34.4.8. Features of remote work with the configuration repository

When working remotely with the configuration repository, the status of objects in the repository does not update automatically. To update information about the objects placed in the repository, you should run the command Actions Update statuses-.
Active users working with the configuration repository remotely are not highlighted in bold in the list of repository users.

34.5. Recommendations for the use of repository

To provide a group of developers with information on the changes made, each developer together with the placement of modified configuration objects in the repository compiles a description of the changes made.
To describe the changes, you need to generate a text report comparing the working configuration with the repository configuration (only for configuration objects placed in the repository) and place it in an accessible location.
The administrator of the repository from time to time (period is determined by the work intensity) should backup the repository and reports on changes made by the developers.
It is not recommended to perform group operations with the repository (retrieving objects, placing objects, etc.), as well as to disable configuration support with the configuration comparison window open.

Chapter 35. Configuration distribution and support

35.1. General information

Standard configurations are changed regularly. This may be due to changes in legislation, introduction of new features or adjustments. Therefore, the support mechanism for these configurations is of great importance for users of typical configurations.
"Distribution", "Configuration support" and "Distribution package" terms are used to describe the support mechanisms.
Distribution There are a complete distribution and distribution of updates. Complete distribution represents a configuration file of *.cf format, specifically generated. Distribution of updates is an update file of *.cfu format.
Support. Configuration support makes it possible to update the configuration using Designer mode and distribution files. The purpose of the support mechanism - is to protect the logical integrity of the configuration. The description also uses the “Supported Configuration” term.
Distribution package is a distribution kit, which includes the installation program for Windows (setup.exe file), Linux (setup file) and macOS (setup.app directory), which represents an installation application for operation systems), as well as distribution files compressed in a special archive (file 1cv8.efd). The language of the distribution package corresponds to the Designer language (localized distribution resources are located in the distribution directory, in a subdirectory with the name corresponding to the language name).
Designer allows to developers of typical configurations to distribute not only new configuration versions, but also to set the support mode for these configurations considering the changes made in user configurations.

35.2. Configuration distribution

35.2.1. Distribution settings

Configuring the distribution is to specify Vendor rules for modifying configuration objects by developers supporting end-user configurations.
To set rights, select Configuration - Configuration Distribution - Distribution settings. The distribution setup dialog opens.

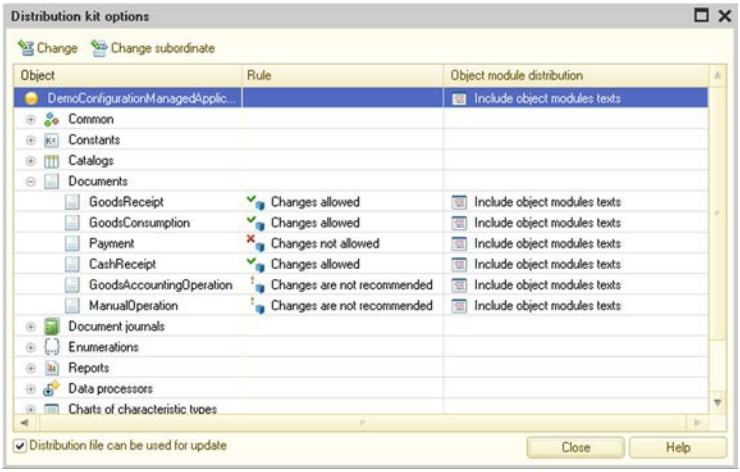


Fig. 442. Distribution settings

In the hierarchical list of configuration objects, the first-level objects are available only.

For each object you should specify the rule of change. To do this, specify the object and click the Edit button, and to change the rules for an object group - select the group and click the Edit subordinates button.

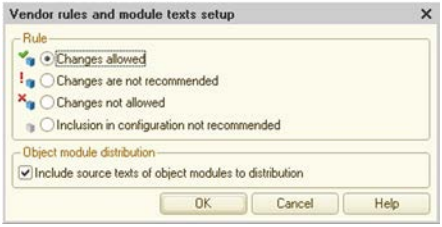


Fig. 443. Vendor rule settings

Multiple selections are allowed in the object tree. In this case, selected objects are configured.
You can specify the following rules:

- Changes allowed - any change in the user configuration is allowed.
- Changes not recommended - changes in the user configuration are not recommended.
- Changes prohibited - changes are completely prohibited. This right does not authorize to modify supported objects. Support cannot be disabled for an object with such a rule. To modify such an object, you need to disable support for the entire configuration.
- Inclusion in configuration is not recommended -when enabling support by combining with the vendor configuration or when updating a new version of the vendor configuration an object with such a default rule is not offered for inclusion in the user configuration.

By default, Designer sets the rule Changes allowed.
If only compiled module text is included in the distribution, you need to clear the Include the source code for object modules in distribution check box. In this case, the functionality is not broken, but the module text is not available for viewing. Source texts removal is not supported for the following modules:

- managed application module;
- form modules;
- command modules;
- modules including preprocessor directives;
- the client general modules operating in a managed mode (thin client, web client, and a managed mode of thick client).

NOTE. If Include the source code for object modules in distribution check box is cleared for the root element configuration, then the regular application, external connection and session modules will be distributed without source texts.

To customize the distribution file, select the Distribution file can be used for updating check box, if the distribution file is intended for standard configuration updating. You need to uncheck the box in the case where the received distribution file is to be used as an intermediate when it is required to make a series of successive updates.
For example, changing the type of attribute with conventional technology can lead to a loss of user-entered data. Therefore, an intermediate version is often used with a new attribute of the required type, while the former attribute remains there. Normally, only the update file is prepared. Special data processor performs data conversion. Then, the next configuration version is used, where the name of the auxiliary attribute of the desired type is replaced with the previous one. The purpose of the second version -is the execution of the transition function only, so it cannot be used as a separate update.
After specifying the rights, click the Close button.

35.2.2. Creating distribution files

To create distribution and update files, select Configuration -Configuration distribution - Create distribution files and configuration updates.
Preliminary, in the Development property category the Vendor and Version properties should be specified.
Vendor - the configuration vendor name.
Version - configuration version (line).

TIP. The recommended structure of the version number:<version>. <sub-version>. <release>. <assembly>, where the structure elements are decimal numbers. This structure simplifies the identification of configuration version during the sequential development and construction of the distribution files (packages).

In the list of templates and configuration updates, configurations are sorted by version numbers taking into account the number structure.
If the configuration or database are modified, Designer requests to save the configuration and update the database configuration.

This will display the distribution generating window (see fig. 444).

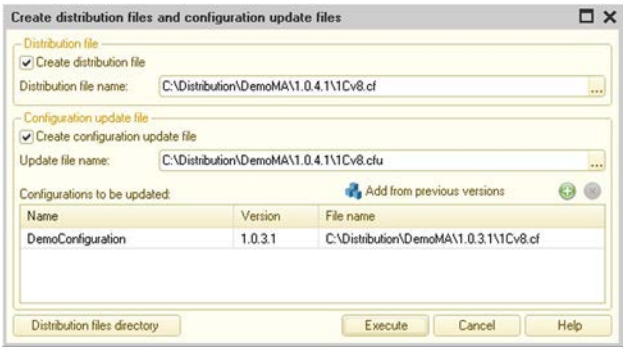


Fig. 444. Generating a distribution package

By clicking on the File Distribution Directory button, a directory is created that will be used to create new versions of distribution directories. The version directory name is the same as the version number.
To create the distribution file, select Create distribution file check box. In the name field, the default name of the distribution file will be suggested.
To create an update file, select Create configuration update file check box. In the name field, the default name of the distribution file will be suggested.
If Create configuration update file check box is selected, then you need to include previous updates in the update file. To do this, run the Add from previous versions command and select distribution files of previous versions.
To generate the specified files, click Run.
The distribution package text files must be prepared in UTF-8 encoding.
Obtained files are passed to developers supporting end-user configurations.

35.2.3. Preparing the distribution package of replicable configurations

To create a distribution package, you must first create a description of the package (stored in a file), and then create a distribution package in accordance with this description.

35.2.3.1. Distribution package description

To create the description of the distribution package, select Configuration - Configuration Distribution - Distribution Package.
This opens a dialog box (see fig. 445).

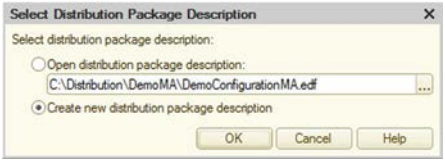


Fig. 445. Selecting a distribution package description

In the dialog box select the mode description.
Use the Radio button to Open package description, if you want to modify the existing description. Description file is specified in the input field.
To create a new description, toggle to Generate a new description.

35.2.3.1.1. Creating a new distribution package description

The window of distribution package creation assistant is displayed.

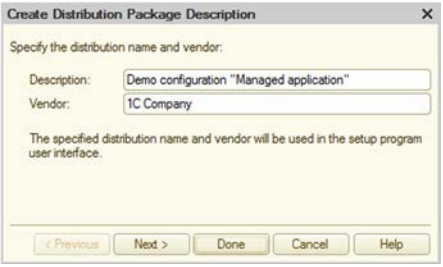


Fig. 446. Distribution package creation assistant

Specify the name of the distribution package and the vendor. Click Next.

In the next step, the template settings are to be specified.

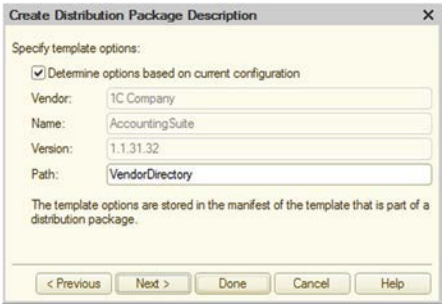


Fig. 447. Creating distribution package description

By default (the Define settings by current configuration check box is selected), settings are selected from the corresponding configuration properties. You can modify these settings, if the check box is cleared.
The Path parameter specifies the directory where the manifest file will be created.
Template parameters are written to the manifest file that is included in the distribution package.
In the next step, the template settings are to be specified.

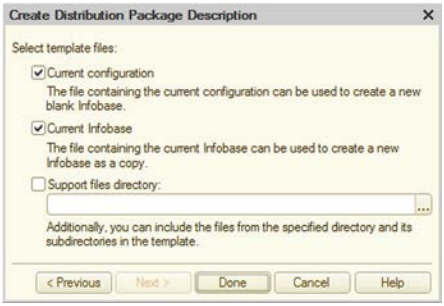


Fig. 448. Setting Templates Parameters

Specify the template files that should be included in the description of the distribution package. Specify the directory for additional files included in the package. All files in this directory, including subdirectories, will be included in the distribution package description.
To create a description, click Finish. Editing window for the distribution package description is displayed.

35.2.3.1.2. Editing distribution package description

Editing window for the distribution package description opens after creating the description (see the section above) or when choosing the editing mode.

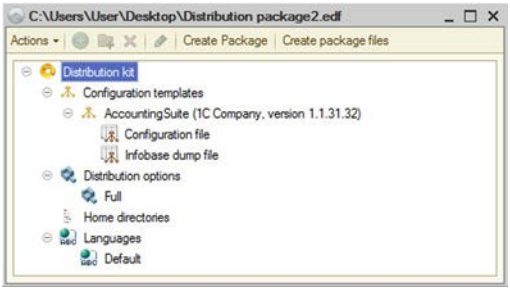


Fig. 449. Distribution package description editor

Use the package description edit window to specify distribution package creation settings.

Description is a tree with four main branches:

- Configuration templates,
- Distribution package options
- Home directories
- languages.

You can add new data, delete unnecessary and modify existing properties for any branch. Use the Action menu to manage the distribution package description. To edit properties, use the property palette.

Configurations templates

Configuration templates are the main items of the distribution package. The configuration templates include files and groups of files.

They can include files of any type (or file sets that match specific masks). They can also include configuration files or dump files created from the current infobase.

Each template is stored in the template directory. Files and file groups are stored in the template directory, their locations based on the manifest file address (file groups are stored as subdirectories).

Only configuration and dump files are included in the manifest file. Other files are located in the configuration template directory.

Templates

To set up the template properties, select a template and specify its properties.

Current configuration - If the check box is selected, use the current infobase data (Vendor, Name, and Version). If the check box is cleared, you can edit these properties:

- Vendor - The configuration vendor (it corresponds to the Vendor configuration property).
- Name - The configuration name (it corresponds to the Name configuration property).
- Version - The configuration version (corresponds to the configuration propertyVersion);
- Location of manifest -A relative path to the template manifest file in the template directory.

Files and file groups

To add a file, select a template and click Actions - Add. This opens the file selection dialog box.

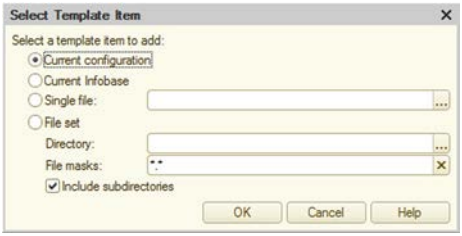


Fig. 450. Adding file to a template

Specify the location of the template element to be added in the dialog box.
The file can be located in the current configuration, the current infobase, in a separate file, as well as represent a set of files of the specified directory, selected by the file mask (files of subdirectories can be included).
To configure file properties, select a file and specify its properties:

- Description - The file name.
- Data - The file data source (a file, current configuration, or current infobase).
- Home directory -The home directory for the file reference.
- File/Directory - The path to the file, absolute or relative to the home directory.
- Place on support - Enable support for a configuration or dump file-.
- Include in manifest - Include the configuration or dump file in the manifest.

- Name in template - The name in the template directory. The name is subject to localization to the distribution package languages.
- Location - The default infobase directory.
- Name for updates - Use the file name in the template directory to name updates.

To add a group of files, select a template and click Actions New Group- In the property palette, enter the group name.

If you specify a group in the list and select Actions Add-, a file is added to this group. The number of files in the group is unlimited.

Nested groups are allowed.

Distribution package options

Distribution options are various combinations of distribution package files. Such combinations can be used for different distribution sets (for example, the full distribution package or updates only).

- Description - The name of the distribution package option.
- Distribution files - A set of distributed files. By default, the full distribution option that includes all files is created. If a number of options is specified, in the property palette, click the Edit link to generate the file set. It is necessary to check files that will be included in the selected option.
- Distribution description file - The description displayed to the user after the product installation.
- Home directory and Directory -The location of the distribution package and the distribution files. If these directories are specified, they will be filled in automatically in the query dialog box.

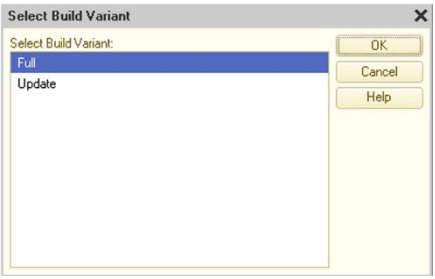


Fig. 451. Construction option

Home directories

Home directories are used to specify distribution file locations that do not depend on a computer.

An environment variable or a directory name can be specified as a home directory. Then the distribution package file name can be specified as a combination of the home directory and the relative path from this directory.

- Description - A home directory name.
- Base - This parameter can take two values:
  - Environment variable,
  - Directory.
- Environment variable/Directory - Contains an environment variable name or the absolute address of the directory depending on the Base value.

Languages

Add the languages for localizing template item names.

To have the names displayed correctly, language names must correspond to 1C:Enterprise directory names containing the localized interfaces. One of the languages is the default language (if there is no line in the current platform language available, the line in the default language is used instead):

- Description -The language name.
- Resource directory - The current language resources directory name.

35.2.3.2. Creating distribution packages

To create a distribution package, click Actions - Create package archive. This creates a zip-archive of a distribution package for the selected distribution option. The archive will contain 1cv8.efd file (containing configuration files, configuration updates, dump files), support files of the distribution package, as well as files that are used to install the application on each of the supported operating systems:

- Windows OS: file setup.exe.
- Linux: setup
- MacOS setup.app directory which contains all the necessary files to perform the installation and which is recognized by the OS as an installation application.

For Linux and macOS executable files, the correct attributes of the executable files are set. Therefore, when unpacking the generated archive on the target operating system, you can immediately run the application installation. However, if you unpack the archive on Windows OS and then copy the files to a disk drive that will be used on Linux OS or macOS, then the attributes of the executable files will be lost.

To create distribution package files, select Actions -Create distribution package files. This creates unpacked distribution files without a setup program based on the selected distribution option.

If the package description has not been saved yet, Designer prompts you to save it.
If more than one option is specified, the option selection dialog box is opened. In the dialogue box you should select the desired option.
Update package files should be placed in the resource directory specified in the Update Directory Address property.
In addition to licensing with the use of a hardware key that limits the total number of workplaces and does not control the used configurations, it is allowed to use electronic licenses that are issued for a specific workplace and control the use of certain configurations on it that are recorded in the license. Software licenses are issued by the 1C Licensing Center and stored on the user's computer.

NOTE. To create a distribution package configuration distributed under the licensing of the user workplace, the -DigiSign parameter is used for the batch run Designer / CreateDistributionFiles and / CreateDistributivePackage which allows you to specify a file with licensing parameters. The partner may create and distribute configurations supporting the licensing of the user workplace under a separate agreement with 1C company.

35.3. Configuration support

35.3.1. General information

A new version of the configuration can be obtained as a distribution file (new version of configuration distribution file) and / or as an update.
User configurations that may be updated with the received distribution files, must be supported.
The object support mode can be changed, having previously disabled full support. The capability of setting user rules (for objects modification) depends on the rules stipulated by vendor and on current locks in the user configuration.
The following features are available for a supported configuration: comparison with distribution, comparison of a new distribution with an earlier one and comparison of the current configuration status with the original one.

If the configuration is supported, objects in the Configuration window are marked with special icons located to the right of the object, along the window border.

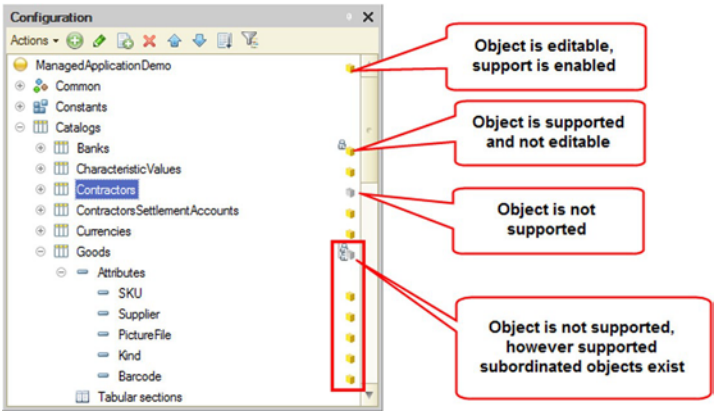


Fig. 452. Support Modes Icons

Descriptions of icons for various support modes are given on fig. 452.

35.3.2. Setting Up Configuration Support

When a standard configuration is installed, configuration support is automatically enabled.

If the configuration is not supported, use the Configuration - Compare and Merge with Configuration from File command to enable support and select the distribution file as a file for comparison.

In this case, the Designer offers to enable the configuration support. Dialog box with the Possible merge option with enabling support is detected message opens and reports the basic parameters of the vendor configuration. Query for support setup is displayed.

Clicking Yes opens the configuration compare and merge window.

After specifying the merge conditions and modes, click the Run button. Configurations are analyzed object-by-object to determine compliance of the vendor rules with the merge rules.

If the vendor rules for any objects conflict with the merge rules, a list of these objects is displayed in a separate window. Adjust the settings in the compare

window to eliminate all conflicts and repeat the merging attempt. If there are no rule conflicts, configurations are merged and support is enabled for the current configuration.
When setting up support and updating the vendor configuration with the option of changing the vendor configuration enabled, the detailed configuration of default support rules can be performed for objects with different distribution rules. If the current configuration is connected to the configuration repository, it is required that all configuration objects, which should be taken from the vendor configuration, and the root configuration object are locked in the repository when a configuration is set for support.

35.3.3. Support Setting

Configuring support settings represents the setting of user rules for each configuration object.
For support setting, select Configuration -Support -Support Setting. The support rules configuration window is displayed.

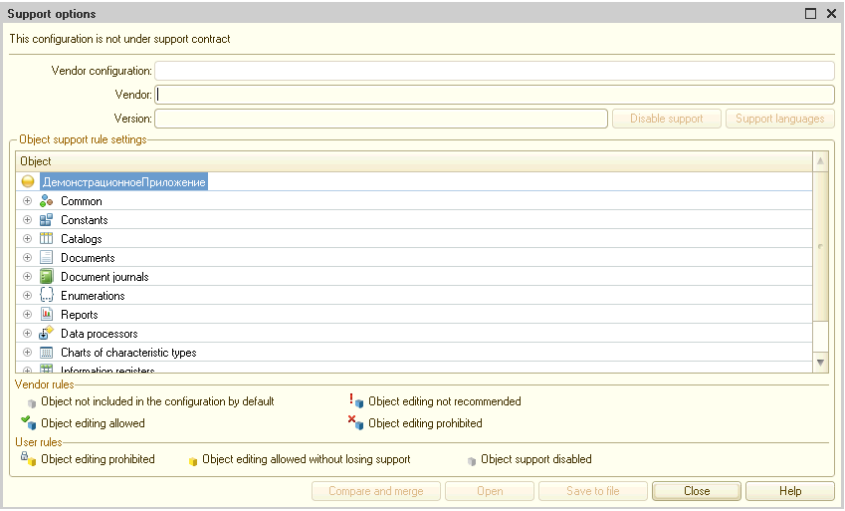


Fig. 453. Support Setting

In the top line of the configuration window, Designer reports on the current support status of this configuration.
The configuration name, vendor, and current configuration version are specified below.

The Object Support Rules Settings section contains the object tree, vendor support rules and user rules for each object.
The vendor support rules cannot be changed. Only user rules can be changed (provided that full support for the configuration has been disabled). You can make multiple selections in the configuration window to configure the same support rules for selected objects. Use the context menu in the support rule columns to change the support rules.
Distribution package may contain data in several languages. Use the support language selection feature to specify the languages that are used in the current configuration. Click the Support Language button to open the language selection window and specify the languages.

Fig. 454. Selecting Configuration Support Languages

35.3.3.1. Full (automatic) support

If the status bar of the support configuration window displays Configuration is supported, it means that all objects of this configuration are supported and cannot be changed (edited). In this case, the configuration can be updated automatically.
To update the current configuration automatically:

- select Configuration -Support -Update configuration;
- select a support file

The update is performed without displaying the Compare and Merge window. Changing user rules for any configuration object is not available in this state, regardless of the vendor support rules.

35.3.3.2. Support with Updates

It is often necessary to modify the standard configuration to meet the specific user requirements. In such cases, support of some objects has to be disabled fully (the vendor object is not supported) or partially (the vendor object is editable with the capability of support).

To gain access to the setting of user support rules, click the Enable modification button in the support configuration window.
After pressing the button, the following warning is displayed: Changing the mode results in inability to perform a configuration update automatically. Continue? If you click Yes, you are prompted to select support rules for objects with different distribution rules.

Fig. 455. Support rules settings

After specifying the support rules and clicking the OK button, the configuration status is changed, and the configuration is supported with the possibility of change. As a result of this setup, all configuration objects will be blocked from accidental changes, regardless of the distribution rules set by the vendor. Now, to make changes to an object, it is necessary to explicitly include the option to change the object or its subordinate objects.
If you do not need to change the support mode (actions were wrongly performed), close the configuration without saving.
After the full support is disabled, reinstalling support for the entire configuration is possible only by performing the following actions in sequence:

- disable support for the configuration (in the settings window, specify the root configuration object and click the Disable support button);
- select Configuration -Load configuration from file;
- select the distribution file.

35.3.3.2.1. Changing Support Rules for Objects

To change the support rules, in the object tree of the support settings window, select the desired object and select Install Support Rule in the context menu. In the dialog box, select the desired user rule from the three possible:

- Vendor object is not editable
- Vendor object is editable, support is enabled
- Support for the Vendor object is disabled

The vendor support rule determines the availability of rules.
If the vendor support rule Changes allowed is set, then the setting of any vendor rule is available.
If the provider support rule Changes are not recommended is set, then installation of any provider rule is also available, but if this rule changes, a warning message will be displayed.
If the vendor support rule Changes disabled is set, the vendor rules can not be installed.

IMPORTANT!After the user rule Support is disabled for the vendor object is set, you can not bring back the rule Vendor object is not editable.

If the selected object contains subordinate objects, then to change the support rules simultaneously for these objects, select the check box Install for subordinate objects (set by default).
Disable support completely To disable support for all configuration objects, select the Actions - Disable Support. The warning is displayed: Disabling support will make it impossible to get updates from the vendor. Continue? To disable support, click Yes. Support for the configuration is completely disabled. If these actions are wrong, close the configuration without saving.

IMPORTANT! Disabling support does not mean that this configuration cannot be updated using the compare and merge method.

35.3.3.2.2. Comparing and merging with the vendor configuration

From the support configuration window, you can make comparing and merging with the original (previous) distribution file. To do this, click the Compare, Merge button. This opens the compare and merge window.

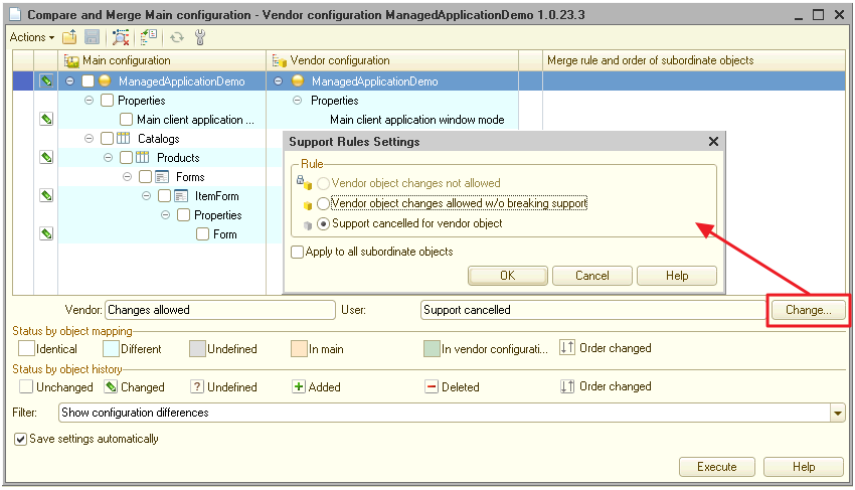


Fig. 456. Comparing and merging with the vendor configuration

Unlike the “normal” mode of comparing and merging configurations, a line showing the vendor and user rules has been added to the compare window, as well as the Edit button to change the user rule. In the dialog that opens you can specify a support rule, indicating it also hierarchically (check box Set for subordinate objects).

When compared with the distribution file, the current configuration is always a descendant of the vendor configuration; therefore, there is no button to select the configuration settings mode.

35.3.4. Updating configuration

To update supported configuration, select Configuration - Support - Update Configuration.

This opens the update assistant.

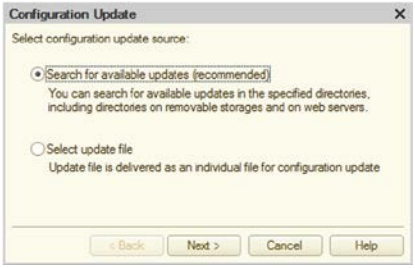


Fig. 457. Update Assistant

In the first step, the user can select a separate update file for a configuration update or start searching for possible update files locations.
If a separate update file is selected, this file is selected in the next step.

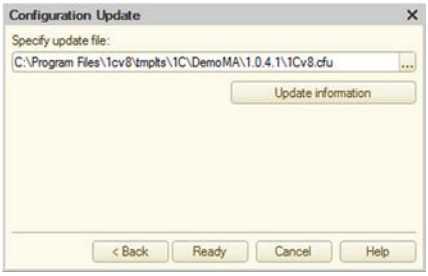


Fig. 458. Selecting an update file

Clicking the Finish button opens a dialog with the basic parameters of the current configuration and the vendor configuration (see below).

If the search for individual updates is selected, then after clicking Next>, the update wizard proceeds to selection of possible update locations.

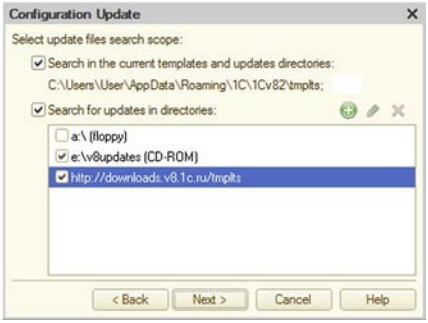


Fig. 459. Updates Locations

Possible locations are:

- current template directory;
- removable drives (the v8updates directory is added for CR-ROM drives);
- address specified in the configuration as the update directory address.

Searching for updates on the specified directories is performed as follows:

- for local directories, searching for update files and template list files is performed in specified directories and subdirectories. The template list files found should describe template directories and be in the root of these directories;
- for remote directories, the template list files are searched only in the specified directories.

If the configuration property Update catalog address specifies a configuration update address, then clicking the Next> button performs user authentication.

Fig. 460. Internet resource access authentication

Enter the user name and password to gain access to the server hosting the update kit, as well as the name and password to access via the proxy server.
In the third step, the updates are shown, for which the corresponding templates have been found. In the list of these updates, an update that is most likely suitable for the configuration update is shown in bold.

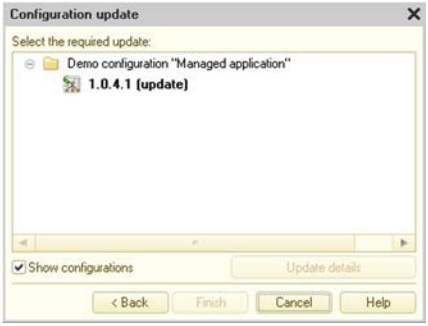


Fig. 461. Selecting an update

If an update located in a remote directory is selected, this update is copied to the current (local) template directory.
Clicking Finish completes the search for an update file.
A dialog box is displayed with the basic parameters of the current configuration and the vendor configuration.

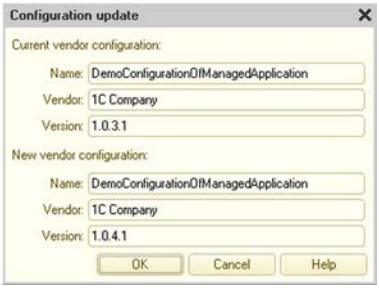


Fig. 462. Update Options

A warning should be noted in the bottom of the dialog to acknowledge that the source of the configuration update file is reliable and does contain an update of the configuration used, without any unwanted modifications.
To continue, click OK.

Designer compares configurations and opens the Configuration Update window.

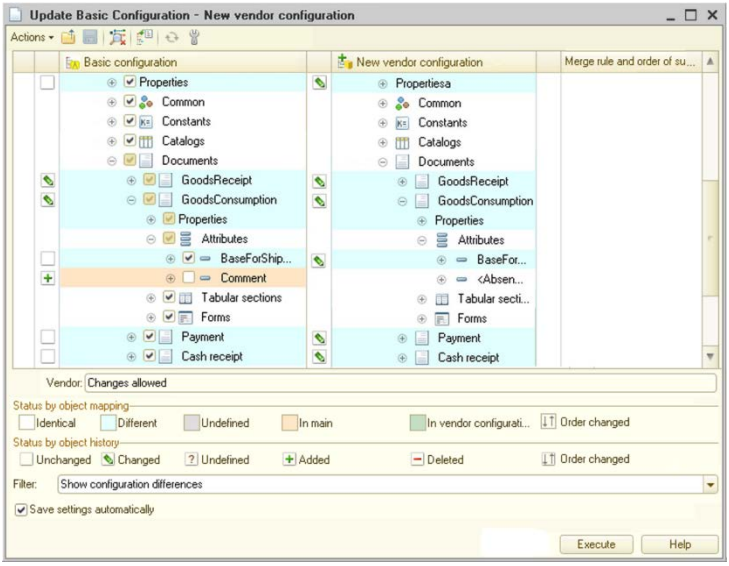


Fig. 463. Updating configuration

If the use of an external program is selected in the Designer settings, then for program modules the item Merge using external program is added to the list of merge modes. When this mode is selected, the external program capabilities are used to merge modules. If the Automatic three-way merge property is not populated for the selected external program, then interactive three-way merge is used. An interactive three-way merge is also used if the external program could not automatically resolve conflicts when merging is performed.

Some actions that can be performed in the configuration update window depend on the Designer settings:

- The context menu command Show differences in modules:
  - Designer Settings: "Retrieve from ...", "Merge with priority ..." modes;
  - External Program Settings: Merging two files;
  - Outputs: ignored.
- The context menu command Show differences in modules irregardless the structure:
  - Designer Settings: Show differences without structure;
  - External Program Settings: Comparing two files;

    - Outputs: ignored.
- Actions performed when the merge mode is specified Retrieve from ... or Merge with priority ...:
    - Designer Settings: "Retrieve from ...", "Merge with priority ..." modes;
    - External Program Settings: Tripartite merge;
    - Outputs: used as the merge result
- When specifying the merge mode Merge using external program, the following actions are performed:
    - Designer Settings: Mode "Merge using an external program";
    - External Program Settings: Tripartite merge;
    - Outputs: used as the merge result
- Actions performed after clicking the Run button:
    - Designer Settings: GroupConfiguration update is supported;
    - External Program Settings: Automatic tripartite merge or Tripartite Merge;
    - Outputs: used as the merge result

When calling a command, the tool specified in the corresponding Designer settings will be used, and if any external program is selected as a tool - the command line specified in the corresponding external program settings will be used to perform the action.

When merging configurations, the modules are merged first. For the Merge using external program mode, launching an external program that supports automatic three-way merge is performed only when conflicts are detected. If the developer did not save the merge result, the module text does not change, and the configuration merge continues. If among all the modules there was at least one unresolved conflict, then after an unsuccessful attempt to merge the modules, the configurations merging stops, and a list of modules that have not been merged appears in the message window. The merge configuration window does not close; the column in the object tree that indicates the merge setup status is updated. The developer should manually merge the non-merged modules and continue to merge configurations by clicking the Run button.

The following merge rules are used to merge the values of type Type:

- Merge mode Retrieve from file:
    - The type description is replaced with the value from the new provider configuration.
- Merge mode Merge with priority:
    - To simplify the description of merging values of type Type, the following assumption will be used: any type is considered a composite,

even if in fact it is not a composite type or the type does not allow participation in a composite type.

- When merging, the following operation is performed for each element of the types being merged:
  - If there is a type available in the old vendor configuration and not provided in the new vendor configuration or in the main configuration, it is not included in the merge results.
  - If there is a type available in the new vendor configuration or in the main configuration and not provided in the old vendor configuration, it is included in the merge results. If the type is available both in the new vendor configuration and in the main configuration, and this type has qualifiers, then the merge rules for qualifiers are used, as described on p. 1767.
  - The type is present in all three configurations:
    - And at the same time, if the type qualifiers for the object of the old vendor configuration coincide with this type qualifiers in one of the remaining configurations, and the qualifier in the other configuration is different, the merge result includes a different type value (the merge priority is not considered in this case).
    - And while qualifiers differ in all three configurations, the qualifiers from the priority configuration are included in the merge result.
    - For the same types with different qualifiers, use rules for qualifiers described below. That is, if there was a Line(10) in the database, the vendor keep it the same, and in the main configuration it is changed to Line(15), the result is Line(15), regardless of priority.
  - If there in one of the configurations in the Type property is a single type not combinable with others, then the Type property of the object is taken entirely from the priority configuration.
- If the merge results in an empty set of types, or other types are added to the non-combinable type, the entire `Type` property is taken from the priority configuration without merging.

Unlike in the usual compare and merge mode, two columns were added to the table field, showing the statuses on the objects history. Description of status icons is given in the lower part of the window. The icons is used to recognize what kind and in what configuration the changes were made.

So, in the example fig. 463 you can see that:

- The name of the `Banks` catalog (from the main configuration) has been changed;

- In the new vendor configuration, the `Activity Terminated` attribute is added;
- In the basic configuration for this catalog, several forms are implemented (`GroupForm`, `Bank NotFoundForm`, etc.), which are absent in the new vendor configuration;
- In the new vendor configuration, the `Loading Classifier` form is implemented, which is absent in the main configuration.

In order to see the vendor rules, specify an object in the table field. The rule will be shown in the attribute located just below the tabular field.

To analyze the changes made in the configurations, it is convenient to use a special filter, the settings of which are called by the Filter button. This displays the filter settings window.

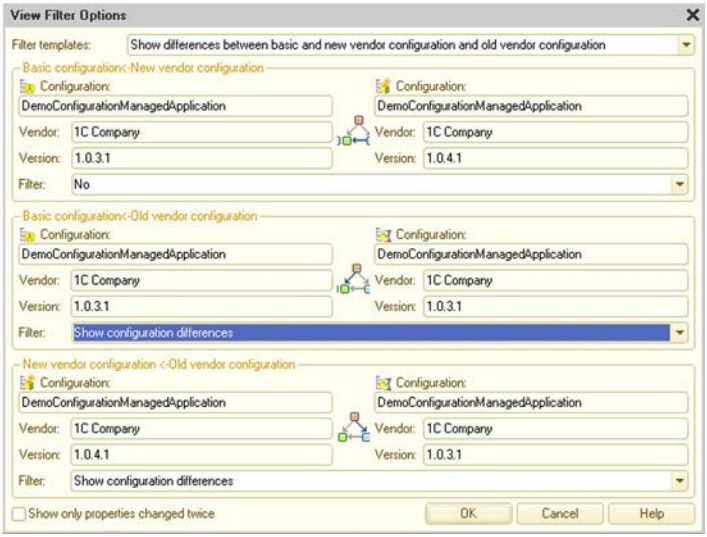


Fig. 464. Preview filters settings

The filter is used to find out the changes in configurations by selecting the objects for comparison and setting the differences display modes.

The basic (current) configuration, the new and the old vendor configurations are the objects of comparison. Therefore, the filter settings window has three sections, similar in attribute details. Each section is intended to configure the comparison mode for any pair of configurations.

Using filter settings you can set differences preview modes across all configurations at once.
For ease (convenience) of comparison, you can use a filter template. The template has the most common variants of settings combinations.
The user makes a decision to merge based on change data and support rules.

35.3.5. Template Directory Description File

In cases where configuration templates (including updates) are located on HTTP (S) or FTP resources, you need to specify a list of configurations located on a particular resource. For this purpose, the template directory description file v8cscdsc.lst is used. This file is located in the root directory of the configuration templates located on HTTP (S) or FTP resources.
This file is used to check for updates on a particular resource while executing the Configuration - Support - Update Configuration command.

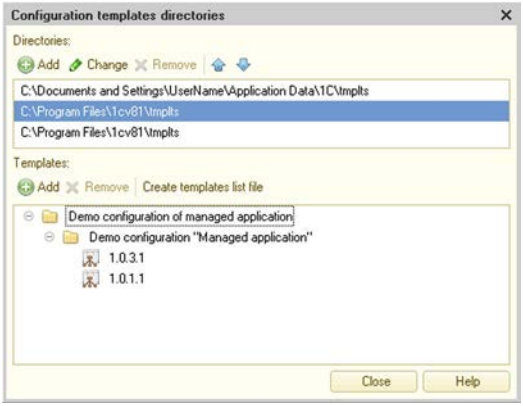


Fig. 465. Creating a template list file

To create and edit the template directory description file, select the item Configuration - Support - Configuration and update templates.
The Directories table contains a list of template directories acquired from the 1CEStart.cfg file. The Templates table lists templates that are available in the selected template directory.

Editing the configuration template directory is to remove template elements from it (or templates together with elements in its entirety) and adding configuration templates to it from other directories.

To add templates to the directory, select the Add command of the Templates table command bar. This opens an assistant to select a directory and template items.

You need to select a directory to add the configuration templates from, and then select items that are missing in the current directory.

To remove templates from the directory, select the Delete command in the Templates table command bar. This opens the deletion dialog box. Next, select the desired deletion method (only the specified items or templates including these items in its entirety) and click OK.

To create a template catalog description file, use the Create Template List File command from the Templates table command panel. This must be performed to save the changes made to the template directory description.

After that, you should move the template catalog (along with the catalog description file) to the required external resource.

The general approach to working with templates located on HTTP (S) or FTP resources is as follows:

1. A directory is created on the local computer's disk where the templates published on external resources will be located.
2. All the templates that are to be placed on external resources, are installed in this directory.
3. In the Designer (in the Configuration template directories window, see above), to the list Directories: the directory with templates created in step 1 is added.
4. In the Configuration template directories window, a template catalog description file is created (the Create Template List File button). To do this, click on the template directory added in step 3.
5. Then you need to move the template directory (along with the template directory description file) from the local computer directory to HTTP (S) or FTP resource.

Chapter 36. Configuration extension

36.1. General information

36.1.1. Terms and definitions

Configuration extension technique - is a special technique for customization of an extended configuration without changing this configuration (as well as with no support disabling).
When considering the configuration extension technique, the following terms are used:

- **Extended configuration** - the infobase main configuration for which the extension is applied or is being developed.
- **Configuration extension** - is a set of configuration objects connected to an extended configuration with a set of objects added to an extended configuration. Extension may include both the extended configuration objects, and objects that are not in the extended configuration.
- **Native object** - is a self-sufficient configuration object that can pertain both to an extended configuration and to an extension.
- **Adopted object** - is a native object added to the configuration extension.
- **Extended object** - is a native object for which some parameters (properties, forms, etc.) are changed in the adopted object.
- **Extension object** - is an adopted object that has been modified relative to an extended object. If the adopted object has only controlled properties, this does not make it an extension object.
- **Resulting object** - is a native object plus the merge of all extension objects (if there are several extensions). If there are no extension objects for a native object - it becomes the resulting object with “no changes”. I. e., all objects in the configuration the user is working with -, are resulting objects, regardless of the availability and number of installed extensions.
- **Extension property** - is the adopted object property that changes the same-name property of the extended object .
- **Controlled property** - is the adopted object property, the value of which is checked when the extension is connected to an extended configuration. If when connecting an extension (in 1C:Enterprise mode), the controlled

property value in the extension does not match the value of the same property in the extended configuration, the extension will not be connected.

- **Checked property** -is the analog of the controlled property. The difference is that the mismatch of the controlled property will result in the impossibility to connect the extension and the mismatch of the checked property will result only in the display of mismatch warning. The extension will be connected and will preserve operability in this case.
- **Mutable property** - is the adopted object property, the value of which in the resulting object will be acquired from the extension.
- **Data extension** - is extendability of the stored data structure in an extended configuration using adopted and native extension objects.

Any adopted object property cannot be both controlled and mutable.

36.1.2. Brief description of the technique

The main purpose of the configuration extension - is to customize an application during implementation (or in the "cloud") to the client needs. At the same time, support for the configuration that is being finalized should not be disabled. As a result, updating a typical supported application remains simple, with the need to make improvements. When developing an extension, some features of the configuration extension functioning should be acknowledged. Thus, the extended configuration can be changed at any time, for example, as a result of an update. In this case, the developer of the extension can not in any way affect the possibility or impossibility of updating. You should also consider the fact that more than one extension may be running in the system and the author of each extension (in general case) does not know how another extension does operate.

Objects placed in a configuration extension can change the behavior of existing objects of an extended configuration without changing the structure of the data stored in the infobase. At the same time, objects located in the configuration extension are able to supplement the storage structure of extended configuration objects (to add details) and to create their own extension objects that will store their data in the infobase of the extended configuration.

Extended configuration objects and extensions are mapped by object names. It means that a change in the object name in the extended configuration will result in the impossibility to connect the extension to such configuration. The analogous situation (impossibility to connect the extension) arises in the case of an adopted object name change in the extension even if the extended object name in the extended configuration remains unchanged. But the platform offers the tools to simplify the search for such problems which will be described farther in this chapter.

Each extension has its own purpose (extension property Configuration extension purpose). The purpose of a configuration extension describes for what purpose this extension is created. An extension can have one of the following purposes:

- Patch - the extension addresses application issues. In such extensions, the use of potentially "dangerous" extension capabilities is assumed, for example, applying method extension with the use of the annotation `Instead`. Patch is intended for a specific application version. When the next version of this solution is released, the author of the extension should analyze the applicability of the created extensions in the new version. Featuring of several extensions for this objective is possible, but it is necessary to ensure that there are no conflicts between such extensions, for example, several extensions should not extend the same method with different objectives. Such extensions may not consider the availability of extensions for other purposes.
- Adaptation - this extension is designed to adapt an application to the certain client requirements. In such extensions, it is recommended not to use potentially "dangerous" features, i.e., those features that may result in conflict of extensions when working together or which depend on the order of connecting extensions. However, careful use of "dangerous" features is allowed, provided that the author of the extension takes full responsibility for ensuring the correct operation of the resulting configuration in new applications versions, and taking into account the extensions that have the Patch purpose. It is assumed that, at each moment, there is a minimum number of such extensions in the infobase. If it is impossible to include the entire set of changes in one extension, then it is recommended that extensions with the Adaptation purpose are broken down into the largest blocks of an extended application.
- Addition - this extension is intended to implement new features of applications that are minimally bound to a specific version of an application. An example of such extension is the creation of a new set of reports, which is absent in an extended application. It is assumed that such extensions should work correctly in case of updating an extended application. At the same time extension with the Addition purpose must not consider in its operation the possible presence of extensions with another purpose. It is assumed that there may be any number of such extensions.

If an extension is applied to a distributed infobase, then it is possible to specify the scope of the extension for it: whole infobase or data area.

The extension is created in Designer, stored in the infobase and can be saved to a file. To add (apply) an extension saved to a file to the application of a specific client, there is no need to use Designer. An extension can be applied using a

special standard function (All functions - Standard - Configuration extension management). It is also possible to apply an extension using the application toolkit, which uses the software interface provided by the platform.

36.1.3. Specifics and limitations

The configuration extension does not support creation of the following native objects:

- General Attributes.
- Event subscriptions.
- Scheduled jobs.
- Definable types.
- Settings storages.
- Languages.
- Document journals.
- Business processes and tasks.
- External data sources

Extension of the following objects is not supported:

- Subordinate subsystems cannot be created.
- If an object cannot be its native extension object - its extension is also not supported.
- Adding attributes and tabular sections for the following objects is not supported:
  - Chart of characteristic types;
  - Chart of accounts;
  - Chart of calculation types.
- Changing the structure of registers of all types is not supported. Recorders composition expansion is only supported.

The extension documents (own and adopted) can move in own extension registers. In other words, the extension documents can act as the registers for own extension registers.

The use of aggregates for own accumulation registers is not supported.

The following managed properties should noted:

- Exchange plan composition;
- Predefined elements for catalogs, charts of characteristic types, charts of accounts and charts of calculation types.

When extending data, the following features and limitations should be noted:

- The extension does not support the use of the `AnyRef` type. `AnyRef` type used in the extended configuration does not include the types provided by existing extensions.

- The configuration extension does not support the use of adobted definable types.
- Whenever a configuration extension is enabled which extends data, no background restructuring is supported by the infobase.
- Testing and patching only processes the data structures of the extended configuration.

Basic application versions do not support operations with extensions. A mobile client supporting an offline mode allows it to use extensions. However, no extensions can be included in the offline configuration.

36.2. Managing extentions

36.2.1. In Designer mode

36.2.1.1. General information

The Configuration extensions dialog is used to manage extensions in Designer. This dialog is accessed through the Configuration - Configuration extensions menu. Using this dialog, it is possible to create a new extension, delete an existing one, open the extension configuration, save an extension to a file, and perform other operations.

In general, commands for operating the configuration extension are accessible in the following ways:

1. Using the command bar commands of the Configuration extensions window.
2. Using the context menu of the selected extension in the Configuration extensions window.
3. Using the Actions - Configuration menu of the window with a configuration extension.

The commands represented in these menus match the commands used for the main configuration by the meaning.

36.2.1.2. Creating an extension

When creating an extension, it is necessary to specify the name of the extension, its synonym, and the prefix of the names of the methods and objects created in the extension. This prefix will automatically be added to the new reports, data processors and subsystems that are created in the extension, as well as to event handlers that are created in the extension forms. In the future, this prefix can be viewed in the Name prefix configuration extension property.

When creating a new extension, the following properties are automatically copied from the extended configuration: Default run mode, Use purpose, Main language, Interface compatibility mode and Compatibility mode. The same properties are marked as managed.
When you create an extension, you automatically create a role whose name is generated from the extension prefix and the `DefaultRole` word. For the extension with the prefix `Ext1_`, the automatically created role name will be `Ext1_DefaultRole`. The access to the extension native objects is automatically enabled in the created role. The new role is automatically set as value of the `DefaultRoles` property of the extension.

36.2.1.3. Saving of an extension to a file

To apply an extension to any application, it is necessary to generate a file with a configuration extension. To perform this action, select the required extension in the list of extensions in Designer and use the Configuration - Save configuration to a file command of a Configuration extensions dialog box command bar or the context menu of this window.
As a result of this action, a file with the extension .cfe will be generated. This file will need to be applied to the application.

36.2.2. In 1C:Enterprise mode

To manage extensions in 1C:Enterprise mode, the Configuration extensions management standard function is designed.

36.2.3. Deleting extensions

Interactive deletion of extensions (in Designer or in a client application) distinguishes between deletion of an extension that does not extend data and an extension that extends data. In the first case, a standard confirmation question dialog will be shown, after which the extension will be deleted.
In the second case, the deletion is performed in two steps.

Removing an extension from the 1C:Enterprise language is performed without additional control on the part of 1C:Enterprise (the `ConfigurationExtension.Delete()` method). However, the developer can perform all the necessary checks independently. To check the extension activity, the `ConfigurationExtension.Active` property is intended. Checking whether the extension extends data or not is performed using the `ConfigurationExtension.ModifiesDataStructure()` method.

36.3. Operations with and extension

36.3.1. Creating an adopted object

In order to adopt an object, it is necessary to use the Add to extension command of the configuration object context menu. After executing this command in the extension, an adopted object is created. When creating an adopted object, all the parent objects are also adopted, if they have not yet been adopted in this extension. Subordinate objects are not adopted. Let's explain the behavior of the examples.
Suppose that there is a `Goods` catalog in the extended configuration.
If the catalog itself is added to the extension, no subordinate objects will be added.
If we now add the catalog form `Goods`, for example, item form, and the catalog itself is already adopted then only the form will be adopted. If a form for an object not existing in the extension is adopted -then the object itself and its form will be adopted. All items of this form but not more will be adopted in this form as well.

When adopting a composite type, the behavior depends on the characteristics of this type:

- If the composite type can include the `Arbitrary` type, then any composite type for such attribute will be adopted as a composite type consisting of a single `Arbitrary` type.
- If the composite type cannot include the `Arbitrary` type, then such composite type will be adopted according to the following rules:
  - If a composite type includes only primitive types, then all the types that form a composite type will be adopted.
  - If a composite type includes at least one reference type (`CatalogRef`, `DocumentRef`, etc.), then this type will be adopted as the simple `AnyRef` type.

But the picture will change if we add, for example, the main attribute of the form to the extension.**A**fter adding `Object` form attribute (it can be considered `CatalogRef.Goods` type), all `Goods` catalog attributes and all configuration objects specified as this catalog attributes types are added to the extension.

When a parametric command is adopted, the command parameter and, as a result, the configuration object, which determines the type of the parameter, are not adopted. If necessary, it is necessary to independently adopt an object (or objects, if the type is composite) of the configuration that determines the type of the command parameter and then specify this object as the parameter type of the adopted command.

36.3.2. Operations with adopted object

Operations with adopted object has some peculiarities.

First, the adopted object contains only those properties that can be set to be managed, checked or modifiable. The opportunity to set a property attribute is the second distinctive feature of the adopted object properties panel. Different attributes of the object property are displayed in the adopted object properties panel differently:

- The controlled property attribute. The mismatch of such properties between the extension and extended configuration will result in the inapplicability of the extension. We can see that the following controlled properties are specified for the catalog `Goods`: Hierarchy and Code allowed length.
- The checked property. The mismatch of such properties between the extension and extended configuration will result in the properties mismatch warning but will not prevent the extension application. The following checked properties for the catalog `Goods` are shown in the figure: Code length, Name length, and Code type.
- The modified property. The Main form property is checked as modifiable. The modified property will be obtained in the resulting object from the extension. If several extensions are applied to the infobase (or data area), - the modifiable property will receive the value from the extension specified last in the list of extensions.
  If the modified property value is not specified the system behavior is analogous to its behavior with the unspecified value of the usual configuration (without extension) property. Thus, for example, if the Main report form configuration property is specified as modifiable, and the form itself is not

specified, then the form automatically generated by 1C:Enterprise will be the main report form.

Do not forget that the configuration object property Name is always a controlled attribute. If at the time of the extension creation, the catalog had the name Goods in the extended configuration, and then it was renamed to Products, then the extension will terminate until the Products name is also set for the adopted catalog in the extension. The same is true for the values of other managed properties.

The extended configuration objects renaming problem can significantly complicate the extension support. To simplify this problem solution, the system offers to track the object correspondence not only by name but also by the internal ID of the configuration object type. This feature is enabled with the help of the extension property Support mapping to extended configuration objects by internal IDs and is enabled for the new extensions automatically. If the feature is enabled then Extended configuration object property, which is automatically set in the controlled status, is filled in for each adopted object of the configuration (including the subordinated ones). In case of an extended object renaming, the extension will not be applied because the name of the object being a controlled attribute has changed.

But the extension applicability can be tested in the configuration. The test will result in the specification of the problem being the absence of the object with the former name and the system will offer several solution options (which can be applied for several selected strings in the applicability window). Various combinations of the following problem-solving options are offered depending on the situation:

- To rename preserving the match -the object is given the name of the object from the extended configuration with the internal ID of the type earlier saved in the property Extended configuration object if such object exists and the meta-data types match.
- To preserve the name and change the match -the internal ID of the object of extended configuration with the same name as that of the adopted object is set for Extended configuration object property.
- Select match -the manually selected configuration object type internal ID is set in Extended configuration object property.
- Disable check -Extended configuration object property is disabled.
- Clear match -Extended configuration object property service value is set for the object. The property with such value is not updated when the extension is saved.
- Set value from configuration object -the extended object property value is set for the property.

- Delete object -the object is deleted from the extension.

If several problems are selected to be solved then the variants applicable to all selected problems will be offered.

`Comment` property of the extension object is intended to comment the extension objects. The property is not multilingual. The comment is not used in the extension application process.

36.3.3. Creating native object

It is possible to create both adopted and native objects in the extension. Creating and editing native extension objects is no different from similar actions in an extended configuration. Native objects name uniqueness check is performed:

- When applying extensions - among extended configuration names and all the extensions;
- When developing an extension - among the names of the extended configuration and the extension where the object is created.

If the extended configuration, for example, contains a `SearchInData` data processor, then no processing with the same name can exist in any extension.

When creating a native object, this object gets the prefix that is set when the extension is created.

36.3.4. Blocking extensions

Due to the fact that the list of extensions is uniform both for access from Designer and for access from 1C:Enterprise, 1C:Enterprise does not allow to perform mutually exclusive actions. In other words, if a window with an extension is open in Designer, it cannot be removed from the list of extensions in 1C:Enterprise mode. The extension, while writing to the infobase in the 1C:Enterprise mode, in Designer is inaccessible for any operations with it.

36.3.5. Shared extension development

Extensions can be developed using the configuration repository. Each extension is considered a separate configuration and a separate configuration repository is created for it. The following is not supported:

- Storing extensions in the same configuration repository as the extended configuration.
- Storing multiple extensions in a single configuration repository.

- The capability to apply an extended configuration to the extension repository and vice versa.

In general, operations with a configuration repository for extension are no different from operations with a configuration repository for an extended configuration. The developer is given the possibility to capture objects, place them in the repository, generate reports on the repository, compare objects from the main configuration of the extension and the repository, etc.

Extension operation commands are located in Configuration - Configuration repository menu of Configuration extensions window.

In the window with the expansion metadata tree, the commands for operating the repository are available in the context menu, similar to the main configuration tree.

36.4. Extension objects

36.4.1. Extension properties

The configuration extension has a set of properties similar to the properties of the extended configuration. But there are also separate properties characteristic only of the configuration extension or which behavior differs from the behavior of the extended configuration. Configuration extension purpose and Name prefix properties are described in the mechanism brief description. Support mapping to extended configuration objects by internal IDs property is intended to simplify for the extension developer the search for the problems arising when the objects are renamed in the extended configuration.

The options analogous to that of the extended configuration are provided for the extension description. For this, the extension has a set of properties:

- Brief description and Detailed description allow to specify an extension description.
- Copyright allows to specify information about the extension developer.
- Vendor information address allows to specify an URL where it is possible to retrieve more detailed information about the extension vendor.
- Configuration information address allows to specify an URL where it is possible to retrieve more detailed information about this extension.

It is also possible to replace the following images of the main configuration: main section image, logo and splash screen. The following features of this replacement must be noted:

- Images from the configuration are completely replaced by images from the extension.
- If several extensions replace the images, then one image will be used for each property. Such a image will be an image from the last applied extension.

The configuration extension provides the capability to change the design of an extended configuration. For this, it is possible to extend the Main style property. At the same time, the capability to extend both existing styles and style elements, as well as create own styles and style elements in the extension is implemented. When extending style elements, only a change of color value is supported, but not the appearance of the style element.

Default role property allows specifying the roles that will determine the accessibility of the extension objects if there are no users in the extended configuration. The property is analogous to the property Default roles of the extended configuration. The following set of restrictions are applied to the role added as the extension default role:

- The role shall not be adopted. Only a native role of the extension can act as the default role.
- The role can't grant access to the adopted objects of the extension. The extension default role manages the access only to the extension native objects.

Configuration extension compatibility mode property by its purpose is similar to the Compatibility mode property of the extended configuration. This property manages the capabilities that the 1C:Enterprise provides in the configuration extension. If an capability is used when editing an extension that is incompatible with the set value of the Configuration extension compatibility mode property, then the infobase configuration will not be saved, and when attempting to perform this operation, - a diagnostic message will be displayed explaining the reason for impossiblity to perform the operation.

36.4.2. Modules

36.4.2.1. General information

The 1C:Enterprise allows to extend almost any module related to the managed application. Extending global server modules is not possible. It is also worth noting that the extension of the privileged common module will be run in non-privileged mode (unless otherwise is allowed by the security profile). In addition, the capability it provided to create native common modules, which can not be privileged and global server ones. Operations with the native common module in the

extension are no different from operations with the common module of the standard configuration.
Operations with the extended module has significant peculiarities that will be discussed in this section.
When developing an extending module, the fact that this module will be in the same namespace with the extended module must be considered. Due to this fact, it is possible to use the extended context (variables and methods) directly from the extending module. It is also worth noting that export methods and variables created in the extension are automatically included in the public context of the extended module.
The extending module provides the following capabilities:

1. Create native methods and variables (if the extended module allows it).
2. Assign native handlers to events that are not handled in the extended configuration.
3. Trap any extended module method (including event handler methods), with the possiblity to:
   - completely replace the original method with the native one;
   - create methods that will be called before or after an extended method.

Creating native method or variable in an extending module is no different from creating a method or variable in an extended configuration, when not using extensions. However, when creating native methods or variables, it is recommended that their names are prefixed so that the method or variable attribution to this or that extension is uniquely identified, as well as to avoid a name conflict with an extended configuration.
The module body and variables area can't be extended in the extension but own variables can be defied and own module body can be written. The extension module body will be executed after the extended configuration module.
Trapping of extended configuration methods (including the assignment of handlers) is implemented using a special annotation feature. There are several different method trapping capabilities which allow implementing almost any scheme for executing extended and extending program code.
In general, an annotated method from a extension module is as follows:

```
&Before("MethodName")
Procedure Prefix_Name()
  // method text
EndProcedure
```

In this example `Before` - is one of three possible annotations, `MethodName` - is the name of extended method, `Prefix_Name` - is the name of extending method. `Prefix_` - is the extension prefix to be specified for all the extension methods. Both procedure and function can be extended. Extending

method name must follow the usual rules for generating the procedures and functions names. In particular - it must be unique.
If the extended method contains any parameters, then:

1. All extending methods must have exactly the same description as the extended method, up to the `Value` key words in the descriptions of the method parameters.
2. Parameter values are divided between all extending methods from all the extensions and the extended method itself. This means that if any extending method changes the value of a parameter, all the methods that get control **after** this method will receive the modified value of this parameter.
3. It does not make sense to specify default values for the parameters of the extended method in the extending method. Default values will be determined from the description of the extended method.

It should also be noted that if an event handler is extended, then the extending method will run before subscriptions to the “extended” event.

36.4.2.2. Extension methods

36.4.2.2.1. Execution before extended method (Before annotation)

If a method is annotated in this way, it means that the extension method will be run first, and then - the extended method.

36.4.2.2.2. Execution after extended method (After annotation)

If a method is annotated in this way, it means that the extended method will be run first, and then - the extension method.

36.4.2.2.3. Framing of extended method (Before and After annotations)

If extending methods are created in the extending module that are marked with the `Before` and `After` annotations, this means that the method that is marked with the `Before` annotation will be called first, then the extended method and then the extension method marked with the `After` annotation will be called.

36.4.2.2.4. Method replacement (Instead annotation)

Trapping a method call implies the situation when the method created in the extension completely replaces the extended method. In other words, calling the extended configuration method will execute the extension method. The name of the reaplced method is specified as an annotation parameter. It is recommended to use this method only when it is not possible to use other methods of extending the methods.
For extended functions it is possible to use only this method of extension.

In order to be able to modify the output of the extended method, the `ProceedWithCall()` global context method is provided. Actual parameters which are passed to the extending method must be specified as method parameters.

The initial version of the text will be as follows:

```
&Instead("Extended")
Procedure Ext1_Extending(Param1, Param2)
  // method text
  ProceedWithCall(Param1, Param2);
  // method text
EndProcedure
```

36.4.2.2.5. Method spot modification in extension (ChangeAndValidate annotation)

This extension method allows the spot modification of the extended module text. The whole module is copied to the extension during the adoption. Then, using preprocessor commands `#Insert … #EndInsert` and `#Delete … #EndDelete` the text fragments to add to the extended module and respectively to delete from it are selected.
Consider an example. Let's suppose we have exponentiation by simple multiplication. When writing the method we have made a mistake and written the addition operation instead of multiplication. We have got the following method:

```
Exponentiation(Basis, exponent) function export
  Result = 1;
  For Index = 1 In Powers of Cycle
    Result = Result + Basis;
  EndDo;
  Return Result;
EndFunction
```

We have detected the mistake already then and there. `ChangeAndControl` annotation can be used to correct the mistake. The corrected method will look as follows in the extension:

```
ChangeAndControl("Exponentiation")
Function Ext1_Exponentiation(Basis, Exponent)
  Result = 1;
  For Index = 1 In Powers of Cycle
#Delete
    Result = Result + Basis;
#EndDelete
#Insert
    Result = Result * Basis;
#EndInsert
  EndDo;
  Return Result;
EndFunction
```

As soon as the configuration developer corrects the error (and the extended configuration is updated) our extension stops to be executed with the diagnostics of the following type:

```
Extension1: Typical: Extension1CommonModule.CommonModule1.Module
module application error. The module text for the method
Exponentiation has changed
```

If the extension applicability is tested in the configurator then the test will result in a message about error and the opportunity to restore mapping to configuration method.

To restore, click the hyperlink in Action column. In the result, three-lateral module combining form will be opened. The following will participate in the combining:

- The method original text which was in the extended configuration. The file will be named `MethodName_RESTORED_`.
- The method text from the extension (with preprocessor directives). The file will be named `ExtensionPrefixMethodName`.
- The method text which is currently in the extended configuration. The file will be named `NameMethod`.

The combining result will be placed in the extension. If the use of an external program for three-lateral combining is not set in the configuration -it will be impossible to restore the mapping to configuration method. The external program necessary for combining is specified in the property Combining with the help of external program mode in the configurator parameters setting dialog (Main menu - Service - Parameters - Comparison/combining - group Modules - group Configuration update on support and configuration extension update). This property shall be set in the value Do not use.
Now, let's consider the operation of such method in more details.
The method is totally copied to the extension during adoption but is marked with an annotation. Then, mark the code to be deleted from the extension with the help of the preprocessor directives `#Delete ... EndDelete` (the "incorrect" code can't be simply deleted from the module in the extension). It will allow the system controlling the equivalence of the extended method and the method in the extension. The code to be inserted in the extended method must be placed inside the preprocessor directives `#Insert ... EndInsert` (it is not enough to simply insert the "correct" code to the necessary place of the method).

From the point of view of configuration, it can be stated that the method extended with the help of the annotation `ChangeAndControl` fully replaces the configuration extended method. The remaining directives will be called in respect of the corrected method. Thus, if another extension is used in the configuration and the same method (in the example above - `Exponentiation()`) is extended with the help of the annotation `Instead` (and by the method `ProceedWithCall()`) the method resulting from the modification of the extended method from the extension with the help of the annotation `CorrectAndControl` will be extended.
It results in one very important note: each configuration method can be extended with the help of the annotation `ExtensionAndControl` just once. If one more extension extended by the same method with the help of the annotation `ChangeAndControl` is connected to the configuration such extension will fail the applicability test.

36.4.2.3. Module editing

In general, editing a module in extension has little difference from editing a module in an extended configuration or a module in a configuration where there are no extensions at all. The main differences include the process of adopting a method from any module of an extended configuration.

To add a method to an extension, it is necessary to place the cursor in the required method (including the line with the name of the method) and select Add to extension command. If during the execution of this command in Designer one extension is open, then this extension will be used. In all other cases, a choice will be offered from the extensions that have been added to this infobase.

When Add to extension command is executed the following actions are performed:

- If the object from the module which the method is added from is absent in the selected extension, - this object is automatically added to the extension.
- If the method is extended from a module other than the form module:
  - The developer is offered to select an annotation for the method (`Before`, `After`, `Instead` or `ChangeAndControl`). In this case, the method extensions that already exist in the extension are highlighted in bold.
  - If an existing method is selected in the extension, the transition to this method is performed. In this case, it is possible to adjust the declaration of the method if it has become different from the extended method.
  - If a nonexistent extension method is chosen, a new method will be created in the extension, which will be preceded by a corresponding annotation.
  - The `Before` and `After` annotations are available for the functions.
  - The following is unavailable for the procedure:
    - The `Instead` and `ChangeAndControl` annotations, if there already are methods with `Before` or `After` annotations;
    - Another annotation if there already is a method with `Instead` annotation.
- If the method is extended from the form module:
  - If the extended method is a handler for one event or one command, then it is suggested to choose how to perform the extension of the selected method: as an extension for the event/command handler, or as an extension of the standard method.
  - If the extended method is a handler for several events or commands, a warning will be generated that this method cannot be extended as an event/command handler, and it will be extended only as a standard method. Then the user will be asked to choose an extension method.
  - For a standard method, which is not a handler for any event, the method will be extended using annotations.

It should be noted that if an extended method is framed by the preprocessor commands, these commands will not be transferred to the extension.

When checking the conformity of the extended method and the extension method descriptions, the following characteristics of the methods are verified:

- The number of parameters and the flag of the parameters transfer "by value" (`Value` keyword).
- A method is a procedure or a function. If the method definition changes from a procedure to a function, and before that the procedure has been extended using `Before`/`After` annotations, the annotation will be replaced with the `Instead` annotation.
- If there are several methods with the same name in the extension module, but which are framed with different preprocessor commands, then the first method from the list of such methods will be updated first in the extension.

Another feature of editing a module in an extension is that it is possible to go to the extended method directly from the extension. To do this, place the cursor on the name of the extended method in the annotation, and then use the standard editor command Go to definition.

36.4.2.4. Interaction of several extensions

When developing extensions the developer should proceed from the following assumptions:

1. An extension must be developed as a standalone product, not relying on the presence or absence of other extensions.
2. More than one extension, which extends the same object of the extended configuration, can be applied at the same time.
3. The order of modules execution in the resulting configuration is determined by the order of extensions registration in the infobase of the extended system. The extension that has been registered last is the first to be used.
4. An unhandled exception that occurs in any extension (or extended configuration) interrupts the execution of the entire extension method chain and is distributed in an extended configuration.

When considering the interaction of extensions and extended configuration, an example of two extensions (`Extension1` and `Extension2`), registered in the extended system in the specified order, will be used. In this case, the earlier the extension is registered, the fewer extensions are between this extension and the extended configuration. So, `Extension1` will extend the directly extended configuration, and `Extension2` will extend the configuration that is the combination of `Extension1` and the original extended configuration. Thus, the extended configuration is at the very bottom of this layered structure, from where the execution of 1C:Enterprise language always begins.

Next, several examples of operating the resulting configuration with different methods of extension will be considered.

Example 1

If both extensions frame the extended method in the same way (`Before` and `After` annotations), then the execution scheme of the 1C:Enterprise language will be as follows:

When attempting to call the `Extended()` method (in the main configuration), the 1C:Enterprise language will be executed in the following order:

1. A method annotated `Before("Extended")` from `Extension2` will be called.
2. Then, the method annotated `Before("Extended")` from `Extension1` will be called.
3. Then the `Extended()` method from the extended configuration will be called.
4. Then, in reverse order (relative to the list of extensions, i.e., `Extension1` and `Extension2`), the methods marked with `After("Extended")` annotations will be called.

Those. first, those extension methods that the developer has marked as called **before** the extended method is executed are called, then the extended method itself is executed, and then those methods that the developer has marked as called **after** the extended method are called.

Example 2

If the extended method in each extension is trapped completely, then the execution scheme will look as follows:

This it happened because the method, marked by `Instead` annotation, completely replaces the extended method. It should be particularly noted that if in this example the order of extensions registration is changed (the first is `Extension2`, and the last is - `Extension1`), then the only method to be executed will be the method from `Extension1`.

Due to such behavior of the “replacing” extension method, it is recommended not to allow situations when some extended configuration method extends with replacement (`Instead` annotation) in more than one simultaneously running extension. In other words, if in the `Document1` document module of the extension, the `Posting` event handler is replaced, then it is recommended not to allow situations when at least one other extension runs in the infobase, which modifies the `Posting` event handler in the `Document1` document. However, if there is an **unconditional** call of the `ProceedWithCall()` method in the “replacement” extension method, then the simultaneous existence of several extensions with such extension methods is quite acceptable.

36.4.2.5. Security profile and extension

The fact that an extension is successfully applied to an extended configuration does not mean that all extension methods that are in the extension will be executed. The operability of extension methods can be changed by the settings of the security profile, which is used by the infobase.

If the application runs in file or in the client/server mode without security profiles, then when the extension is applied:

- In normal 1C:Enterprise language execution mode - it is permissible to extend both client and server methods. Without limitations to the location of methods.
- In safe 1C:Enterprise language execution mode - only client methods and server form handlers will be extended, which are set through the property panel can be extended. The extension will not be applied to other server methods (annotated form modules server methods and server common modules).

The capability to extend server common modules in client/server mode is managed by the security profile.

36.4.3. Forms

36.4.3.1. General information

Only the form items are adopted when a form is adopted. The form attributes, commands, and parameters are displayed in the extension form editor but are not actually adopted. An adopted form looks exactly the same as this form looks in the extended configuration as of its adding to the extension. The adopted form module contains no methods. Designer “saves” the form in the extension when the form is added. Further below, this form will be called **saved**.

An item representation is determined by the extended configuration until the developer changes the adopted form items properties. No additional actions are required to change the properties of the items in the adopted form. But in order to change the properties of the form attribute, command or parameter -it is necessary to add the relative item to the extension (analogously to any other configuration item).

The form components update logics:

- If an attribute, parameter or command is not added to the extension the changes of these components in the extended configuration are automatically displayed in the extension.
- The attributes, parameters, and commands added to the extended configuration are automatically displayed in the extension.
- If the form items properties are changed in the extended configuration such changes will be displayed in the extension only after the saved form update.

In this case, the resulting form (in 1C:Enterprise mode) will be formed **taking into account** all changes in the extended configuration.

- The attributes, parameters, and commands deleted from the extended configuration will be automatically deleted from the extension without waiting for the saved form update. But if the deleted component of the form is added to the extension it will remain in the extension (until the saved form is updated) but will be absent from the resulting form (in 1C:Enterprise mode).
- If the properties of any form object added to the extension are changed the system will execute three-lateral comparison to obtain the resulting object.

36.4.3.2. Working with the form editor

The form editor displays the edited form extension. The extension name is displayed in the square brackets in the edited form title.

Update form extension command of the form editor command panel shall be used if it is necessary to transfer the changes and additions made in the extended configuration after the extended form creation to the extension.

Show only changed items button available in the form editor command bar allows it to hide form items with their properties remaining unchanged in this form (vis-a-vis a form being extended).

If this button is pressed, a list of form items displays only items with their properties remaining unchanged in this extended form. If this button is released, -all form items are displayed.
When the form items structure is changed in the extended form this fact will be defined in the form editor and the user will be offered to update the form in the extension.

Such offer will be made both on the form editor first opening and in the process of work (if the extended configuration form is changed in its window). The analogous offer will be made if the form elements structure is changed in the extension itself.

The form components display also allows determining the form components status.

Different statuses of form components are shown on **Ошибка! Источник ссылки не найден.**. The numbers in the figure mean the following:

- Number 1- the form components adopted from the extension. The form items are always adopted. The attributes, parameters, and commands must be clearly adopted if necessary.
- Number 2- the form components not adopted in the extension. These components act as the "links" to the components in the extended form. To change the properties values of such components they must be added to the extension first.

So far as a form item available in the extended form is concerned, you can select only properties which are changed in the extension. For that purpose, press Show only changed properties button available in property bar of an adopted form item.

If this button is pressed, property bar displays only form item properties, which are changed in the extension. If a form item is not covered by the extension, -this button is unavailable in the command bar.
The form attributes, commands, and parameters can both be added to the extension and "deleted" from it. The "deletion" is performed by pressing Del button on the relative form component. In this case, the component status will change to that shown by number **2** on **Ошибка! Источник ссылки не найден.**. In this case, all changes made in the extension will be lost. After the "deletion", the behavior returns to the behavior set for this component in the extended configuration. This behavior does not cover the form items. The deletion of the adopted items of the form and the form items return to the settings "as in the extended configuration" are not supported.
To remove an item from the resulting form it is recommended to set the property `Enabled` in the value `False` (to uncheck the box) for such item in the extension.

36.4.3.3. Working with the form's module

The extension of a form module is somewhat different from the extension of any other module. The extension modules features in the form module are described in this section.

The first feature is that it is necessary to create handlers in an extending form using the relative form item properties panel (form, item, command). When attempting to create a handler, the Designer will ask when the created handler must be called and how it (the handler) is to be created.

At the same time, the images will appear that display the type of extension in the property palette, next to the name of the extending handler. Also, if the extension type `Before` or `After` is selected - another input field will automatically be displayed to specify the pair extending method: if an extending method with the `Before` extension type is created another field will be created for the extending method with the `After` extension type and vice versa. There will be no such behavior for a replacement handler (with `Instead` extension type).

If it is necessary to change the type of the extending handler - delete the existing handler and recreate it.
In the event handler code with the `After` extension type it is forbidden to change the values of the `Fail` (for all the events with such formal parameter), `Execution` (for the `DragStart` event), and `CancelEdit` (for the `OnEditEnd` event) parameters. In handlers with `Before` extension type, the handlers in the following extensions and the handler in extended form will not be called in the following cases:

- setting `Fail` parameter to `True`;
- setting `Execution` parameter to `False` (for `DragStart` event);
- setting `CancelEdit` parameter to `True` (for `OnEditEnd` event handlers).

The next feature is that the managed form editor creates extending handlers without using annotations.

36.4.3.4. Resulting form building

In 1C:Enterprise mode, the resultant object is generated by combining the extended and all the extending objects. The combining process is as follows:

1. The extended configuration is combined with the first (in order of registration) extension.
2. The combining is repeated for each subsequent registered extension, where the result of the previous combining operation is used as the extended configuration. In terms of each next combining operation, the result of the previous combining operation will be considered an extended object.

In the process of combining of each component (element, attribute, parameter or command) of the managed form, the changes in the extended and extending forms are analyzed in regard to the saved one. Based on the results of the analysis, it is decided which component will be displayed in the resulting form:

1. If there are no changes - component is taken from the extended object;
2. If the change is in the extended form only - the component is taken from the extended form;
3. If the change is in the extending form only - the component is taken from the extending form;
4. If the change is in both the extended and extending forms - the component is taken from the extending form.

Matching of the components of the managed form is performed by the name of the corresponding element.

The above mode is not applied:

1. To conditional appearance. In this case, the elements of conditional appearance are combined from the extended form and all the extending forms similar to merging the configurations.
2. Command interface of the resulting form is generated by adding command interface elements from the extending form to the command interface of the extended form. In this case, it is possible that the form will contain several commands with the same representation, since these are generated by the commands from different extensions, but with the same representation.

When determining the resulting managed form elements order, the capability to arrange elements without placement conflicts is determined first. If such capability exists, then the order of the elements of the resulting form is determined as follows:

- The form elements added to the top of the list (whether in the extension or in the extended configuration) will remain at the top of the list regardless of the order of the remaining elements. The form elements from the extensions (in order of appearance) will be placed first followed by the ones from the extended configuration.
- The form elements added to the bottom of the list (whether in the extension or in the extended configuration) will remain at the bottom of the list regardless of the order of the remaining elements. The form elements from the extended configuration will be placed first followed by the ones from the extensions (in order of appearance).
- It is also possible to place other form elements next to an extended form element. To do this, it is necessary to create a group in the extension where the element of the extended form and those elements that are required to be

located next to it are placed. Then, when changing the location of the original element in the extended form, the whole group where this element was placed will be located in the resulting form in the new location.

If the order cannot be defined without conflicts, the following algorithm is used:

- The source is the list of extension elements in the order that is set in the extension.
- To determine the position of an element added to the extended configuration, the element that exists in the saved configuration and is located before the added element (in the order of the extended configuration) is searched for:
  - In accordance with the order of the elements in the extension, it is determined which element is next - existing or added by the extension:
    - If this element is existing, the main configuration element is added after the found element.
    - If an element or elements are added by the extension, then the extended configuration element is added after them.
  - If several elements were added to the extended configuration, the subsequent elements are added after the previous ones, even if the previous ones have already been added by the extension.
  - If the element after which the elements were added is not found in the extension (for example, it has been deleted), then the previous one is found in the order of the main configuration. If the search has reached the top of the element list, then a new element (or group of new elements) is added to the top of the resulting list, but after the new extension elements added to the top of the list.
- New extended configuration elements added to the top of the list are added immediately after the new elements of the extension added to the top of the list.
- New extended configuration elements, added to the bottom of the list, are added before the new elements of the extension, added to the bottom of the list.
- If when adding it turns out that the element has already been added by the extension, then such an element is skipped.
- The elements that were deleted in the main configuration compared to the saved version are deleted.

36.4.4. Form command interface

Form command interface is supplemented with command interfaces of extensions. Commands added to the form command interface manually will be displayed after the commands of the extended configuration (in 1C:Enterprise

mode). The consequence of this is that several commands with the same representation can simultaneously exist in the resulting form.
Deleting an extended configuration command is not possible.
It is necessary to consider that in the form editor in Designer, the developer will not see fragments of the command interface, which are provided by the extended configuration.

36.4.5. Object forms

For configuration objects, it is possible to change the main and auxiliary forms of adopted configuration objects. To do this, it is necessary to create a form in the extension, which will replace the existing form of the configuration object, then mark the configuration property describing the replaced form as a modified attribute and specify the new form in this attribute. In order to change configuration properties associated with forms (Main report form, Main search form, etc. properties), there is no need to adopt the root element of the configuration tree, it is enough to use the properties of the root extension element.

36.4.6. Subsystems

Subsystem can be either adopted or native extension object.
It is impossible to delete configuration objects that are added there in the extended configuration from the structure of the adopted subsystem. But it is possible to add other objects to the adopted subsystem: both native and adopted (in regard to the extension).
Native extension subsystem can include both native extension objects and adopted objects.
For adopted objects, it is possible to change the image of the subsystem by changing the modified Image attribute.

36.4.7. Access rights

36.4.7.1. General information

IMPORTANT! The behavior considered below is relevant only if the extension property Configuration extension compatibility mode is set in the value Version 8.3.14 and higher (including Do not use in version 8.3.14). The behavior in the preceding extension compatibility modes is described in the documentation for the system «1C:Enterprise version 8.3.13 and lower (https://its.1c.ru/db/v8313doc/bookmark/dev/TI000001531).

Using the configuration extension, it is possible to extend an already existing role and to create native role in the extension.
In the simplest case, when the roles are set neither in the extended configuration nor in the extension -the user can access all objects of the main configuration but not to all extension objects. The absence of roles is a kind of a degenerated situation for an extension because a role granting the rights to all native objects of extension is created when an extension is created.
Force default roles for all users attribute can be set when an extension is connected to an infosabe. This checkbox operates in the following way:

- If the box is checked and `DefaultRoles` property is set then the minimal set of rights will be defined by the roles specified in `DefaultRoles` property. In this case, the administrator can expand the content of roles of a certain user with other roles of the expansion (not included in the default roles of the expansion).
- If the box is not checked and `DefaultRoles` property is set in the expansion then the minimal set of rights will be defined by the roles specified in `DefaultRoles` property only if there are no users in the configuration to which the extension is connected.
- In all other cases, the rights to the extension objects will be defined by the system administrator who shall grant access to the expansion objects (with the help of the configurator or other tools) to each user.

It should be understood that if Force default roles for all users property is set when the extension is connected it will be impossible to deprive the users of the rights granted by the default roles. It is necessary to have the configuration extensions administration right to set Force default roles for all users property.
If there are no native roles in the extension the accessibility of the extension objects can be managed with the help of the extended configuration roles.
The accessibility of the extension native objects in the resulting configuration is determined by the status of Set rights for new objects property in the extended configuration role. It means that the resulting configuration treats all native objects of the expansion as its new objects. Thus, the expansion native roles will be accessible in the native roles in which Set rights for new objects role property is set.
Set rights for attributes and tabular sections by default role check box of the extended configuration provides the accessibility of the attributes and tabular sections in the extension native objects. The roles where the box is checked -in the extension native objects will contain all rights to the attributes and tabular sections.
Based on the above, the following general scheme of work with access rights can be outlined:

- A role shall be adopted only when it is necessary to change access rights (or to restrict access to the data) as compared to the extended configuration.
- In an extension without native roles, the access to the extension object is defined by the extended configuration roles settings.
- The extension native roles define the rights for the extension native and adopted objects. An extension native role can't change the rights to the extended configuration objects which are not added to the extension.

The work with the adopted and native roles will be described in the subsequent sections.

36.4.7.2. Adopted roles

36.4.7.2.1. General information

It is possible to change the permissions specified for the extended configuration objects for the adopted role. It is possible to change not only the access rights themselves, but also data access restrictions. It is also possible to create (and use) data access restriction templates in a role adopted object. The adopted role does not provide the capability to change access rights that are set for the root configuration object (administrative rights, capability to use a particular client application, etc.).

In an adopted role, all the rights, by default, correspond to rights from an extended configuration. This corresponds to the special value of the check box in the permissions setting. However, it is possible to set or reset any right. Thus, using the adopted role, it is possible to both extend and narrow the user rights.

The data access restrictions specified in the extension will only be used if the rights in the adopted role are explicitly specified for the access right. In this case, the access restrictions to the data from the extension completely replace the access restrictions to the data from the extended configuration (for this right and this role). If the access right is adopted from the extended configuration - data access restrictions specified in the extension will not be used.

36.4.7.2.2. Edit

When editing an adopted role, it is possible to set the “third state” for the access right: use the value from the extended role.

The rights changed in the extensions are marked with number 1 and the rights set by the extended role are marked with number 2- in this figure.

The opportunity to manage the automatic assignment of right only to native objects of the extension is provided in an adopted role. Set rights for new native objects check box is responsible for it.
Set rights for attributes and tabular sections by default property of the extended role is responsible for the rights to the attributes and tabular sections of the extension objects. The opportunity to set independent rights for the configuration subordinate objects in the extended role is defined by Independent rights for subordinate objects check box of the extended role. In other words, the rights to the attributes and tabular sections as well as the possibility to set independent rights for the subordinate attributes are defined by the extended role settings.

36.4.7.3. Native roles

Native role does not allow to set access rights that are bound to the root configuration element (administrative rights, use of client applications, use of exclusive mode, etc.). Own extension roles can be added to infobase users both interactively and using the 1C:Enterprise language. After removing an extension, all of its native roles are removed from the lists of user roles.
A user whose roles are supplemented by an extension, is displayed in the configurator with a specific icon.
A native role automatically created when an extension is created can be randomly renamed or deleted. If an extension prefix is changed the automatically created role remaining is suggested if the following conditions are met:

- The role name is formed as `<ExtensionPrefix>DefaultRole`.
- The role with such (as specified above) name is set in `DefaultRoles` extension property.
- The role `<ExtensionNewPrefix>DefaultRole is absent from the extesnion`.

Actually, it is offered to replace the role name `<ExtensionOldPrefix>DefaultRole` with the name `<ExtensionNewPrefix>DefaultRole`. The system imposes no restrictions on the modification of automatically created native role.
A separate editor is intended for the extension native role editing. Its exterior and options are analogous to that of the extended configuration role editor but it has some differences both in the structure of edited rights (which is caused by the extension specific features) and in the check box role behavior.

Set rights for new native objects check box is applied only to the extension native objects.

Set rights for attributes and tabular sections by default check box provides (if the box is checked) full rights to the attributes and tabular sections for both native objects of the extension and of the extended configuration. This check box also grants the rights to the attributes and tabular sections added to the resulting configuration by other extensions.
Independent rights of subordinate objects check box operates absolutely equivalently as the same check box in the configuration role.
In the case when the extension becomes inactive (as a result of detection of applicability problems), the own role of extension continues to be available in the `InfobaseUser.Roles` collection as a normal configuration object. Such a role can be added or removed from the list of roles of any user of the infobase. However, the `AccessRight()` function for such a role will always return the `False` value. If roles are received for user from another data area, then all roles added by extensions become invisible (they cannot be received, but they are not deleted when the user is saved).
The recommendations on the management of access rights to the extension objects when developing the extension without any adopted objects:

- The automatically created default role is enough if the extension is small and presupposes no role access to the extension sections. This role will grant access to all objects of the extension.
- It is recommended to specify the rights for the better part of users in the default role(s) if the extension presupposes the role access to the extension sections (for example, formatting extension or the dispatch of reports where all users can format the report but the responsible user shall send it). The administrator shall clearly specify the role of the responsible user (who will send the reports).
- It is recommended to pay special attention to the rights granted by the roles included in the `DefaultRoles` extension properties if the extension is connected with the checked box Use default roles for all users. It is connected with the fact that all users of the infobase will get access in accordance with the default role(s) rights.
- If it is necessary to grant the rights to certain objects of the extension selectively, not to all users, the following is recommended:
  - To create a separate role for such objects of the extension.
  - Not to specify this role in `DefaultRole` structure.
  - To exclude the access to the "private" objects of the extension from the roles specified in `DefaultRoles` property.

36.4.7.4. Extending user rights and security profiles

Change of access rights for objects of an expandable configuration depends on the type of operation of the infobase:

- The extension of access rights to the extended configuration native objects is not supported if the extension is connected in safe mode in file mode. It is possible only to restrict access rights.
- In the client-server version, change of access rights is governed by the presence of the security profile and its settings:
  - If the application solution operates without using a security profile, extension of roles is not allowed.
  - If an application solution operates with the use of a security profile, then access rights can be restricted in any case, and access rights can be extended only with the appropriate security profile setting.

    If the check box for access rights extension is selected in the properties of the security profile (in the group Full access is allowed), then you can extend the access rights for any object of the extended configuration.

    In the event that this check box is cleared, then it is impossible to extend access rights. If, at the same time, the Roles that restrict the extension of access rights security profile property is filled in, (roles are separated by the symbol «;»), then the system operates as follows: A role from the configuration extension can extend the access right only if at least one of the roles listed in the Roles that restrict the extension of access rights security profile property is set to an expandable access right. It is not necessary for the roles listed in the security profile property to be used for users of the infobase of the application. An example of this behavior will be discussed below.
- For own extension objects, access rights are granted in accordance with the settings of extension roles (own or borrowed), without collaborating with the security profile. This behavior is independent from the security profile settings.

Let's consider the extension of access rights on example. Suppose there is a document `ExpenditureBill` in the configuration. In the extension, you should extend the access right `Write`. To do this, use the role of extension `SellerExtension`. Also in the extended configuration, the roles `SellerTemplate1` and `SellerTemplate2` are presented. In the security profile property Roles that restrict the extension of access rights, both roles are indicated: `SellerTemplate1` and `SellerTemplate2`.

Example 1

In the extended configuration for the document `ExpenditureBill` the right `Write` is indicated as follows:

- `SellerTemplate1` - the right `Write` is allowed;

- SellerTemplate2 - the right Write is prohibited.

In this case, the operation is as follows: at least one of the roles specified in the Roles, restricting the extension of access rights property, the Write right is allowed (this is the role SellerTemplate1). This means that the role from the extension SellerExtension **can** provide the right Write for the documentExpenditureBill.

Example 2

In the extended configuration for the document ExpenditureBill the right Write is indicated as follows:

- SellerTemplate1 - the right Write is prohibited;
- SellerTemplate2 - the right Write is prohibited.

In this case, the operation is as follows: no role from the list of roles specified in the Roles that restrict the extension of access rights property does not provide the Write right. This means that the role from extension SellerExtension **can not** provide the right Write for the documentExpenditureBill.

In other words, the algorithm for determining whether to extend access rights from an extension can be described as follows: the state of the required right from all the roles listed in the Roles restricting access rights extension property is combined "by OR". The result of the addition "by AND" combines the right of the role of the extension.

36.4.8. Reports and data processors

Creating a new report or data processor does not differ from that in a configuration that does not contain extensions. At the same time, there is a capability to extend the existing forms of reports or data processors of the extended configuration by adopting the required object.

For an adopted object, there is a capability to create attributes of the objects, tabular sections and details of the tabular sections. Operation with these objects does not differ from that in a configuration that does not contain extensions.

For an adopted object Report, you can override the basic entity relationship diagram. To do this, add a report to the extension, then mark the Basic entity relationship diagram property as changeable and then create the necessary entity relationship diagram for the report. If you need to change (rather than recreate) the entity relationship diagram of the adopted report, it is recommended to copy the entity relationship diagram from the main configuration to the extension, specify this diagram as the main one in the adopted report and then upgrade

this diagram. Adopting objects included in the entity relationship diagram should be done manually.

36.4.9. XDTO packages

XDTO package can be either adopted or own extension object.
You can adopt not only the entire package, but also some part of this package, if the entire XDTO package is not required for the normal operation of the extension. If the adopted item already exists in the extension, the user is requested to update it.
The following is transferred when adopting:

- All established element properties, including list and aggregate;
- All the established properties of the created parent elements, except for list and aggregate.

When an extension is attached, an adopted XDTO package is monitored for a strict subset, i.e. all adopted package elements should be included in the package from the extended configuration with an exact match of the number and value of the properties.

36.4.10. Web services

Publishing of native Web service extension is possible in two ways:

1. Unconditional permission to use any Web service from any attached extension. To do this, when publishing, use the check box of the dialog box of the Publish the web services extension by default publication or similar in meaning attribute publishExtensionsByDefault of the element of the ws file default.vrd.
2. Manually specifying Web services extensions that can be used in the filedefault.vrd, in the element point.

Using Web services from extensions is available by the value of the Web service property Name and the value of the Web service property Name of the publication file. If there are several Web services in extensions with the same values of the Publication file name property, then when the Web service is accessed using the value of this property, the first found Web service will be used.

36.4.11. HTTP service

When the extension is attached to the infobase, it checks the uniqueness of the value of the Root URL property of its native HTTP extension services. Extensions in which HTTP services are existed, for which the value of the Root URL property violates uniqueness do not attached.

36.4.12. The predefined elements

In the extension there is a capability to manage the controllability of predefined elements. For this, it is necessary to set the check value flag when the extension is attached to the Predefined property.
There is also a capability to adopt a specific predefined item in an extension. In this case, the following are transferred to the extension:

- The name, code and name of the predefined element;
- The hierarchy of subordination in the event that a subordinate element is created in a hierarchical object (directory, etc.);
- The type of the predefined element (for the chart of characteristic types). Types that are missing from the chart of the characteristic types of the adopted object are removed from the element type during the transfer.
- Type and flag of outbalance (for the chart of accounts).

When attaching an extension, it is checked:

- The match of the name, code and name of the element;
- The match of the parent of the predefined item (for a hierarchical configuration object);
- Item type (for a chart of characteristic types);
- Account type (for a chart of accounts);
- Flag of outbalance of the account (for the chart of accounts);
- The components of the flags of the account (for the chart of accounts);
- Exact match of the components of settings of the account extra dimension (for the chart of accounts).

36.4.13. Spreadsheet documents

When editing an adopted spreadsheet document, some capabilities that facilitate the editing of the spreadsheet document are provided. These features are collected in the menu Main menu - Table- Extension (and the corresponding button of the spreadsheet document editing command bar).
The Extension method command allows to specify how the spreadsheet document will be expanded: combining the extended table and the table from the extension (Merge method) or the extension table will completely replace the extended spreadsheet document (Replace method).

The command Display changes allows to highlight areas of the spreadsheet document.

In the spreadsheet document shown in the figure, the cells of the expanded spreadsheet document are shown in gray background, which have not been changed in the extension. The background color of the cells that have been added to the extension is displayed in green. And for cells of a spreadsheet document, whose properties have been changed in the extension, - the current background color of the spreadsheet document is used.
So, in the document shown on **Ошибка! Источник ссылки не найден.**, you can note the following changes:

- in rows 11 and 12, cells are added in column 4 (display of the Article column in the tabular section of the document Consumption of goods);
- in column 6, cells are added in rows 1-4 and 9-10. This was due to the addition of a column with the article.
- in the cell located in the column 2 of the row 12, the value of some properties of the cell is changed, so the cell is displayed with a background color other than gray.

To see the differences in the values of the cell properties, use a button on the command bar of the cell properties palette.

The button Show only changed is available only in those cells of the adopted spreadsheet document in which the properties of the cells are changed. When you click on this button, the properties palette starts to display only the changed properties. Thus, in the example on **Ошибка! Источник ссылки не найден.**, it can be seen that in the cell, which is located in the column 2 in the row 12, the horizontal alignment is changed. The remaining properties of this cell remained unchanged.
With the help of the Show saved table command, you can see an extended spreadsheet document, in which the areas that are deleted in the spreadsheet document, which is located in the extension are marked. These areas are marked with red background. This spreadsheet document is displayed in "view only" mode.

It can be seen that in the extended spreadsheet document between the `Head` and the `GoodsHead` areas there were two rows: 9 and 10. However, one row (number 10) has been removed in extension.

The Show result command displays a table (in the "view only" mode), which is a spreadsheet document, which will result from the use of an extension that contains an extended spreadsheet document.
The Update table extension command allows to update an extended spreadsheet document if it is modified in the extended configuration.

36.4.14. Exchange plans

In the extension, there is a capability to manage the controllability of the components of the exchange plan. To do this, it is necessary to set the check value flag when the extension is attached to the Components property of the exchange plan. There is also a capability to adopt a specific element of the components of the exchange plan in the extension. In this case, the following are transferred to the extension:

- The flag of participation of the configuration object in the exchange plan;
- State of auto registration of changes.

For adopted exchange plan, there is a capability to add attributes, tabular sections and attributes of tabular sections, forms, commands and templates. The adopted exchange plan can include both your own extension objects and objects from the extended configuration (which were not previously part of this exchange plan). The capability to exclude extended configuration objects from the exchange plan is not supported. In other words, an extension cannot reduce the amount of information passed by the exchange plan.
When attaching an extension, it is checked:

- The coincidence of the flag of participation of the configuration object in the exchange plan.
- The coincidence of the auto registration of changes.

In the extension there is a capability to create your own exchange plan. Such an exchange plan will have the following features:

- Own extension exchange plan can not participate in a distributed infobase.
- The system does not provide tools to automatically synchronize the components of extensions between data exchange nodes. This synchronization should be performed separately.

36.4.15. Catalogs

Directories can be both adopted and native. In adopted directories, there is a capability to add attributes, tabular sections and attributes of tabular sections.

36.4.16. Documents

Documents can be either adopted or native. In adopted documents, there is a capability to add attributes, tabular sections and attributes of tabular sections. For native extension documents, the use of numbering units and sequences is not supported.
A document may act as a recorder only for the register that is in the same “part” of the application as the document itself. In other words, a document from an extension can act as a recorder for its native and adopted extension registers and cannot act as a recorder for its native registers of an extended configuration. As well as conversely: a native configuration document cannot be a recorder for its native extension register.
An extension document (both native and adopted) can act as a recorder for any adopted extension register, except for a turnover accumulation register with the aggregate mode enabled.

36.4.17. Enumerations

An enumeration can be adopted or native. An adopted enumeration allows the creation of new values.
In this case, remember the following specifics:

- If an infobase uses the values from an extended object then, in the case of unconnected extension, such values are displayed analogously to the links to the nonexistent objects. The representation is restored after the extension connection.
- Upon the extension deletion from the infobase, the links to the values of the enumeration from the extending object will be cleared and replaced with an empty link with extended enumeration type.
- The situation when the same links to the value of enumeration from different extensions have different sense is possible. It is possible if the enumeration native value is renamed in the extension created by copying. In this case, writing of such value in undivided data will lead to a logical problem for the application solution user. For example, an enumeration value with ID (internal) `ID1` has the representation `Aircraft` in one extension and the representation -`Steamer` in another extension. It is evident that it will be difficult for the user to understand what is meant for this enumeration value. This implies two features:
  - It is not recommended to save the native values for the adopted enumerations in the extensions created by copying. It is recommended to create all native values manually.

- It is not recommended to specify the native values of adopted enumerations of extensions which scope is set as "Divided data" in the undivided data.

- In case of an attempt to set an extension of a configuration in which the enumeration value ID equal to the ID of an enumeration value of extended configuration or another extension is used, it will be impossible to restructure the infobase.
- If the internal ID of the enumeration in the extended configuration is the same as the internal ID of the enumeration of the extension then:
  - the infobase restructuring is not locked;
  - the extension containing a conflicting enumeration value is not connected.

  To restore the infobase (or area) operability it is necessary to delete the conflicting enumeration value in the extension and to download a new version to the infobase. The existing links to the enumeration value from the extension will be allowed by the enumeration value from the extended configuration.

36.4.18. Session parameters

The session parameters can be adopted and native. The extension session native parameters can be used in the texts of restrictions of access to data.
The initiation and work with the extension session native parameters do not differ from the initiation and work with the extended configuration session parameters.

36.4.19. Functional options and functional option parameters

Functional options (and functional option parameters) can be either adopted or native. Native functional options supported by the extension can include both native and adopted extension items.
Functional option values can be stored both in native and adopted extension items. An adopted functional option can be extended by using both adopted and native extension items.

36.4.20. Filter criteria

Filter criteria can be either adopted or native. Extension native filter criteria can include attributes of native extension items only. Adopted filter criteria can be extended by using attributes of native extension items only.

36.5. Extension and distributed infobase

Configuration extension can be used in a distributed infobase. In this case, the extension will be passed between the nodes of the distributed infobase using exchange messages, thereby ensuring a uniform components of extensions within the border of the distributed system.

To indicate that an exchange plan can pass extensions, select the Include configuration extensions check box on this exchange plan. This check box can be set only for exchange plans that participate in a distributed infobase.

At the same time, each extension attached to the infobase has a property Used in a distributed infobase. Setting this property means that this extension will migrate to subordinate infobases. The capability of changing this property is not related to the actual presence in the infobase of exchange plans included in the distributed infobase.

The `ConfigurationExtension` object has the `MasterNode` property, which allows to determine that the extension was added to the current node (the property has `Undefined` value) or arrived using the data exchange mechanism (contains a link to the master node of the distributed infobase).

Creating the initial image and exchange message

When creating an initial image of a peripheral database that supports the passage of configuration extensions, all extensions that are attached with the Used in a distributed infobase check box set are passed to it. Creating the initial image will not be possible if in the session from which the initial image is being created:

- There are unconnected extensions with check box Used in a distributed infobase.
- There are extensions that change the data for which the check box Used in a distributed infobase is set and which are not active.
- There are extensions changing the data for which Used in a distributed infobase box is not checked.
- There are extensions with the check box Used in a distributed infobase, for which new versions of extensions are loaded.

At the time of generation of the message exchange to the session from which the generation is performed, the same requirements are imposed as in the generation of the initial image.

Features of operation with extensions

In the peripheral node of the distributed infobase, it is forbidden to edit the properties of extensions and delete extensions that are added in the master node.

These actions are prohibited both interactively and using the methods of the 1C:Enterprise language. A flag that the extension is received using the data exchange mechanism is displayed in the interface of the standard function Extension configuration management.

When starting a session, extensions are applied in the following order:

- extensions added by the central node in the same order in which they are applied in the central node;
- extensions added to the current node in their order.

If the message comes to the node of the distributed infobase that the Used in a distributed infobase check box for the extension is cleared, then such extension is deleted.

If infobase has extensions which was received using the messages of exchange of the distributed infobase, then in such infobase, using the method `MasterNode()` of the manager of the exchange plans, there is a capability to set only that node from which the extensions were received or disable the infobase from the distributed system (set `Undefined` value). In particular, this means that without removing extensions, it is impossible to switch the infobase from one distributed infobase to another by changing the master node.

36.6. Security profiles

When connecting an extension to an application, there is a capability to specify whether the extension will run in safe mode or not. This can be done:

1. Explicitly indicating the Safe mode check box. In this case, the extension will be performed in safe mode or unsafe mode (depending on the state of the check box).
2. Indicating the name of the security profile, within the border of which, the extension will be performed. This feature is available only if the application is used in the client/server option.

Specifying a security profile when attaching an extension indicates that the mode of operation of the extension is determined by the settings of the specified security profile. In the security profile, access to the extension is specified in the settings of the External Modules element.

If the security profile prohibits the unsafe use of external modules, and the extension requires such use, then its (extension) can be added to the exclusion list. To get the checksum value of the extension to add to the list of exclusions, you should use the capabilities of the dialog boxes with the extensions or the program interface for operation with the extensions. The received value of the checksum should be entered in the appropriate field in the cluster administration console.

When an extension is connected, the capability of enabling unsafe mode (if required) is checked and if the current extension does not meet the requirements of the specified security profile, - the extension attachment does not occur and an error is issued.
To change the settings for safe mode when attaching an extension, it is necessary that the user who performs these settings has the `Administration` right (in addition to the right `ConfigurationExtensionsAdministration`, which allows to perform other actions with extensions).

36.7. Attachment of extensions and time of their operation

36.7.1. General information

An extension is attached to a user session before calling the event handler `SessionParametersSetting` and remains attached until the session is terminated. Updating an extension (using administration tools) does not update the extension in existing sessions. Existing sessions will continue to operate with the "version" of the extension that was available at the time of the start of the session. In order for the user session to start managing the new "version" of the extension, - it is necessary to restart the session.
If the configuration extension uses data extension, then the creation of new tables (for storing the extension data) in the infobase is performed:

- In the configurator - when performing the restructuring of the infobase.
- In 1C Enterprise mode - while adding or updating an extension. In fact, a change in the database structure is performed during the execution of the `Write()` method of the `ConfigurationExtension` object.

When deleting an extension, the procedure is the same, but instead of creating tables, they are deleted.
The order of attaching extensions is determined by the purpose of the extension and its scope. Extensions are attached in the following order:

1. Patch:
2. Customizations;
3. Addition.

In addition, within the border of each purpose, extensions with a scope Infobase are first attached, and then - Data area. If there are several extensions with the same purpose and scope, - the order of attaching such extensions is generally not defined.

When attempting to attach extensions, each extension is sequentially checked for compliance with the controlled attributes between the extension and the configuration of the infobase. If at least one controlled attribute fails validation, an error message is displayed to the user and this extension is not attached. When checking properties with multiple values (component type, components of the overall attributes), the value from the extension is controlled for the occurrence - all values that are in the extension should be in an extended configuration, however, some of the components of the extended configuration may not be part of the property being monitored. As a result, all extension objects cease to participate in generation of the resulting configuration.
If problems are detected when connecting extensions, the user is presented with a dialog box with information on which extensions have problems, and a detailed list of problems is available by clicking on the hyperlink Information for technical support available in this dialog box.
In addition to the interactive applicability test, the 1C:Enterprise platform provides the capability to perform applicability testing from code in the 1C:Enterprise language. There is a capability to perform a check for a specific extension using the `ConfigurationExtension.CheckCanApply()`. There is also a capability to check all extensions that are currently attached. To do this, use the `ConfigurationExtensions.CheckCanApplyAll()` method.

36.7.2. Operation at discrepancy of data structures in a configuration and infobase

In the event that not all extensions that extend data are attached at the start of the session, the application can read the extended configuration data of successfully attached extensions without errors. At the same time, an attempt to write an object whose data is supplemented with a disabled extension will be impossible. In order for the entry of such an object to become possible, - it is necessary to restore the functionality of the disabled extension. Obviously, for extensions that do not extend the data, the above features is inoperative.

36.8. Operation with extensions from the 1C:Enterprise language

If you need to implement your native tools for operating with extensions, you should use a specialized software interface. Its methods are available if the current user has the right to `ConfigurationExtensionsAdministration`. The current session should run in an unsafe mode if the `Write()` method of the `ConfigurationExtension` object will be used.

The program interface is provided through the configuration extension manager, which is represented by the global context property `ConfigurationExtensions`. A detailed description of the methods is given in the Syntax Assistant.

There is a capability from the 1C:Enterprise language to verify that the configuration object is expanded in this operating session. The method of configuration object `<Metadata object>.ThereareChangedByConfigurationExtensions()` is intended for this. This method will return the `True` value if in the current session the selected configuration object is modified by extensions in one of the following ways:

- property of the configuration object has been modified;
- added a new child object of the configuration object (form, template, command, etc.);
- properties of one of the child objects were modified.

Using the method `<Metadata object>.ConfigurationExtension()`, there is a capability to determine which configuration extension (or extended configuration) belongs to the configuration object.

Chapter 37. Service features

1C:Enterprise includes many different service features and auxiliary modes. Their description is given in this Chapter.

37.1. Switching between application windows

37.1.1. Window states (window display modes)

37.1.1.1. Setting Window Display Mode

Windows can be arranged on the screen in different modes, such as:

- MDI. A window is displayed inside the work area.
- Floating mode. Windows can move freely across the screen (not only within the work area).
- Docked mode. Windows are "pinned" to the borders of the work area.
- Auto-hide mode.

To change the window display mode, select the desired mode from the window header context menu.

Floating windows are always displayed over windows in all other modes. If several windows are in the Floating mode, the active window is displayed on top of the other windows.

Windows in the Normal and Floating mode are able to be attached to each other. If two windows in the same mode have the property Dockable set (in the context menu of the window header), these windows can be docked to each other. To dock windows in the MDI state, hold down Shift while dragging the windows. You can also dock windows that are already docked to other windows.

If the Dockable property is not set, you cannot select the Docked or Autohide modes.

For the Docked and Autohide modes, the Dockable property is always set and cannot be disabled.

The Docked state requires special discussion. In this mode, the window can:

- Docked to the Designer window border.
- be docked to a border of another window with the Docked mode set;
- Displayed on top of another docked window (stacked windows).

If the Docked mode is set for a window, this window is docked to a border of the work area or to another window in the Docked mode.

To resize a docked window, move the pointer to the free (facing the work area) border and drag it.

If a window is in Floating mode, and its border is dragged close to the work area border or another Docked window, the window borders may change abruptly. If you release the mouse button at this moment, the window docks and its state changes to Docked.

You can change the state of docked windows by dragging them. To move a window left click and hold the window header and drag the window to another location. This changes the window state to Floating.

There are several ways to dock multiple windows to one side of the Designer window or other window. Windows can be arranged in layers where each window occupies the entire side of the work area of the Designer window or other window. You can arrange windows in a sequence, so that each window is located in a single layer along the border of another window. If there are more than two docked windows, you can arrange some of them in layers and the others - in a sequence.

To move a docked window on top of another window (create stacked windows), do the following:

- Drag the window
- move the window so that its header appears on top of the header of another docked window, while the contour of the window being dragged should have the tab contour at the bottom;
- release the mouse button.

As a result, the tabs are displayed at the bottom of the docked window. The tab text matches the window header text. Each tab contains a docked window.

Click the tabs to switch between the windows.

Clicking Close in this case closes only the active tab.

To detach stacked windows from each other, drag one of the window tabs away (when a window is ready to be detached, its border changes).

You can change the order of tabs by dragging them within the tab area.

You can stack a window to other stacked windows by dragging it to the tabs area instead of the header. When doing so, you can choose a position for the new window tab among the existing tabs.

You can also arrange windows in the Autohide mode. If the Autohide mode is selected, an additional bar appears on the side of the work area. This bar contains window tabs, including a tab for the current window, while the window

itself is docked to this side of the work area. The bar location is determined by the position of the Docked window. In this state a window is displayed on the screen while it is active. Once you switch to another window, the previous window is hidden. To display this window, hover the mouse over the window tab (do not click it). Once you move the pointer away, the window is hidden.
For docked windows, in addition to the stacked mode another mode can be set as well. For example, it can be convenient to set the Autohide mode. In this case, an additional line will contain only tab icons. When hovering the mouse pointer over any tab icon, the window is displayed, and its name is shown in the additional line.

37.1.1.2. Restoring default window positions

You can change position, size and status of windows as described above. At the same time, when closing, the latest display parameters of the window are saved and applied automatically when the window opens again.
To restore the original position, size and status of the window, select the Restore Window Position command in the context menu of the window header (or window pane header). This reverts the parameters to the default values. The window is closed and re-opened with the new display options.

37.1.1.3. Selection of Window Mode for Object Forms

You can configure the window state mode for applied object forms at the first opening in the 1C:Enterprise mode. For this purpose you can use the Window properties category for each object form.
Window State - indicates the state of the window at the first opening. If the Normal state is selected, the window state cannot be modified.
Dockable Window - indicates the ability for the window to be attached to other windows.
Window Location - if Auto is specified, the window position is determined based on the state and dimensions of the work area. If you specify the Center option, the window is placed in the center of the work area when it is first opened. If Don't Overlap Owner is selected, location of a subordinate window is determined so that it does not overlap the owner window or control (if the work area has free space). If there is no place to display it, the subordinate window attempts to expose the left part of the owner window provided that 50% of the owner width can be unclosed.
The position of the docked window - is determined by the position of the window of the form in the docked state relative to the work area.

Change size - permits or forbids changing the size. The DontChange option does not affect the ability to change the size of a window in the Docked and Autohide modes.

37.1.2. Managing windows

Use the Windows dialog box to manage the open windows. This includes switching to a selected window, saving changes, arranging windows, and closing one, several, or all windows.
To view the list of all open windows, click Windows - All windows... Only the windows with MDI status are included in the list.
If the content of an open window is modified, the window is marked with an asterisk (*) in the list, to the left of the window name.
All the operations available in the dialog box are applied to one or several selected windows. Use the standard item selection methods.
Operations that can be performed with a single window or a group of windows are described in the following table.

Button	**Action**	**Number of selected windows (condition)**
Activate	Activate the selected window.	One
Save	Saves all changes for certain documents types that support this operation (e.g., text and spreadsheet documents).	Any
Close windows	Close the selected windows.	Any number
Arrange	Arranges the on-screen windows placement Horizontally - arranges selected windows from left to right. Vertically - arranges selected windows from top to bottom. Cascade - arranges selected windows in a cascade	More than one

Button	Action	Number of selected windows (condition)
Join	Join two windows	Joining is allowed
Detach	Detach joined windows from each other.	
OK	Closes the Windows dialog box	Any

37.1.3. Use of New Window mode

Use Windows- New window to make a copy of any form editor, text editor or spreadsheet editor window. This will open a new window instance, an exact copy of the current window, in the work area of the program. Its contents will be identical to the original. The window number will be displayed in the header.
This mode is convenient for viewing, comparing and editing information in a window. Any changes in any window instance are displayed in all window instances.
We recommend using this mode, if the window is a configuration object form with items distributed across several pages and you need to view these pages at the same time or view the module and form dialog box simultaneously. Similarly, this mode allows editing of long texts or templates. Also, One window instance can be used to view one part of a text or a template and the other window instance can be used to view the -other part.

37.1.4. Service windows

While using the Designer, service windows will provide various support information. These windows include:

- message window,
- a window with search results from all texts,
- text templates,
- syntax assistant.

The first two types of windows are automatically displayed if needed (text templates and Syntax Assistant are displayed upon user request).
The purpose of each window is described in the corresponding sections of this manual; this section will outline the general principles of operations with such windows.

To manage windows you can use context menus that are opened from the window title bar and within the window. The First menu option is used to manage the window itself, while the second menu option operates - window's contents. The contents of the message window are not automatically cleared. To clear it, use the clear command from the context menu.

37.1.5. Closing windows

Any window can be closed by clicking the close window button or by using the Shift + Esc or Ctrl + F4 combinations on the keyboard.

37.1.6. Windows navigation history

Windows navigation history is stored during the session. For example, you can use this tool to return to the initial window when navigating to the place where variables, procedures and functions are defined.
While using windows history, you can move back within the history using Windows- Move Back item. To move forward within the history, -use Windows-Move Forward item.

37.2. Setting Designer parameters

To set-up various Designer parameters, use Tools- Options property.
The Options window is organized as a card type file. All available parameters are combined in groups. To access controls of a specific group, click on the corresponding window tab.

37.2.1. Overall

This tab contains the only parameter that defines the operation mode and interface for Designer- Edit configuration for startup modes.
This parameter can take two values:

- Managed application- the platform hides interface items used for editing properties and objects in the ordinary run mode;
- Managed application and ordinary application- no interface items are hidden and the user can edit the configuration in both ordinary and managed mode at the same tame.

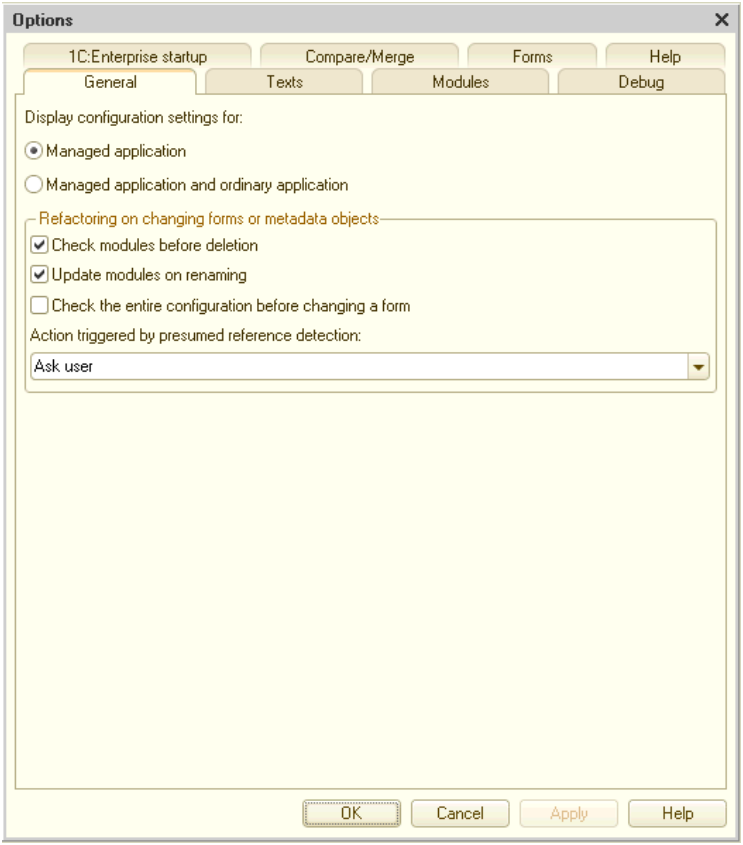


Fig. 466. General parameters

This parameter is stored across the infobase, - this allows to set it up for each particular infobase. By default, the parameter is set to editing mode for managed mode, however if you load-up the default run mode configuration with this property set to Ordinary (or this property changes interactively), the parameter is set to editing of managed and ordinary run modes.

Below is a list of Designer interface items that are hidden when the parameter is set to the managed run mode:

- Styles metadata tree branch;
- Interfaces metadata tree branch;
- Main interface configuration property;
- selection of main interface in the user's editing dialog box;
- Form tab in Designer parameters;

- drop-down list Default run mode in Modules tab- Validation;
- configuration property Use managed forms in ordinary application;
- configuration property Use ordinary forms in managed application;
- Interfaces tab in metadata editing windows;
- Open ordinary application module configuration context menu item;
- flag of managed form creation in the form wizard;
- commands Output form wizard;
- Client (ordinary application) common module property;
- Additional tab on the Forms tab of the configuration object editing window;
- interactive Activation Processing and Before Execute Interactively event handlers at a business process route point of the Activity type.

Refactoring if forms or metadata objects are changed group of parameters allows you to set-up how the platform will react when making these changes:

- Check modules before deletion-if the check box is selected, the platform will search if an object to be deleted is used somewhere. If the object is used (or might be used), the platform will ask for additional confirmation for deleting and will display the search results.
- Apply changes to modules if renaming-If the check box is selected, the platform will search for the old name of the object, that was used before renaming the configuration object and will apply changes to the code.
- Check the entire configuration when changing forms-If the check box is selected, when renaming or deleting the platform will check if the old name of an attribute, element or managed form parameters is used somewhere. The module form check is done in any case, regardless of the check box selection

Actions triggered by presumed reference usage parameter allows you to set-up refactoring if it's impossible to unambiguously define whether the object to be deleted or renamed is used somewhere:

- Ask a question- For each case of a presumed reference use, a question will be asked.
- Continue operation- the required action and replacement for actual usage cases will be performed. Presumed usage cases will be displayed.
- Cancel operation-inthis case, the action won't be performed and the search results of the object reference usage will be displayed.

37.2.2. Texts

Controls in texts tab are used to configure the text document editor.

Drag and drop text. To move or copy text blocks in text editor, you can use the drag-and-drop mode.

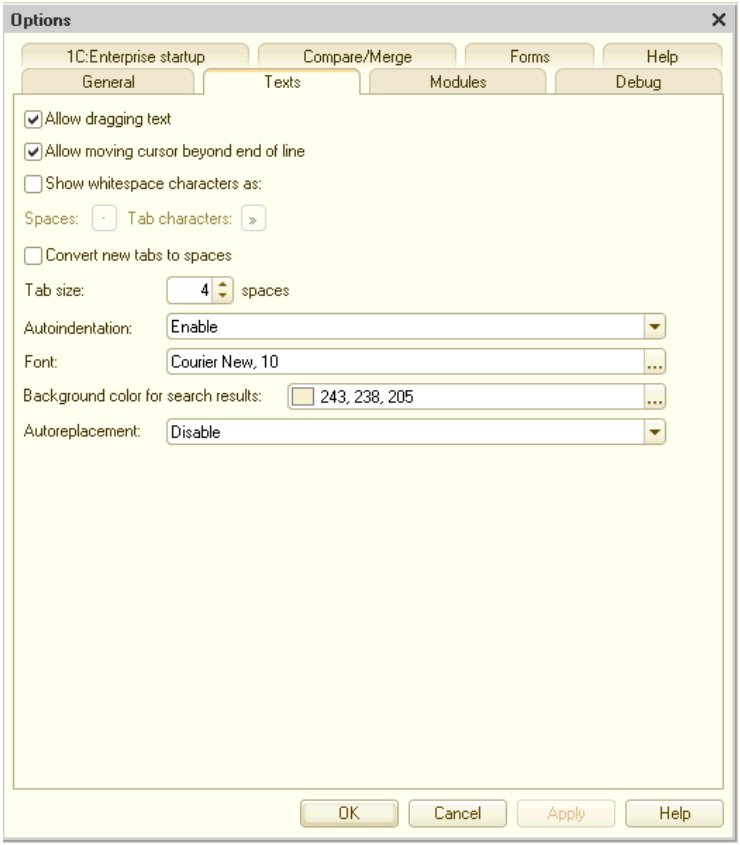


Fig. 467. Text editing parameters

Allow moving cursor beyond the end of lines. If this check box is selected, the cursor can be positioned beyond the end of the line. If this check box is cleared, the cursor position is limited by the Line Break character.

Show spaces and tabs. If this check box is selected, the spaces and tab characters will be displayed in the text. Use this option to verify the text formatting. If the Show spaces and tabs property is selected, the Space and Tab properties will allow you to enter characters that will be used to display spaces and tabs:

- Space - specifies the symbol used to display spaces;
- Tabs - specifies the symbol used to display tabs.

Replace tabs when typing. If this check box is selected, the typed tabulation symbols will be replaced with the number of spaces specified in Tab Width property.
Tab Width. Defines the number of characters that correspond to a tab symbol when typing text.
Auto-indentation. Sets or removes auto-indentation when entering the line break symbol (pressing the Enter key).
Font. Selecting a font for text input. We recommend using a fixed-width font for module text input (for example, Courier New).
Background color for search results allows to set a background color for search results in text editor.
Auto-replacement. If this property is enabled, the entered text, that corresponds to the text in the Automatically replace string attribute of the template, will be automatically replaced with a template text. If the Enable with tooltip option is selected, the template text is displayed after a short pause while the text to be replaced is being entered. If the Only tooltip option is selected, template text will be displayed after a brief pause, but there will be no text replacement.
To replace input text with a template text, press Space or Enter after typing a string.

37.2.3. Modules

The Modules texts tab controls are used to configure the color highlighting of module syntax, auto-indentation, tabulation width for modules and to set up grouping. Settings are combined into two sections, each one located on a separate tab.

37.2.3.1. Setting editing parameters

The Edit tab is used to set color for highlighting syntax in modules, auto-indentation and tabulation width for modules.

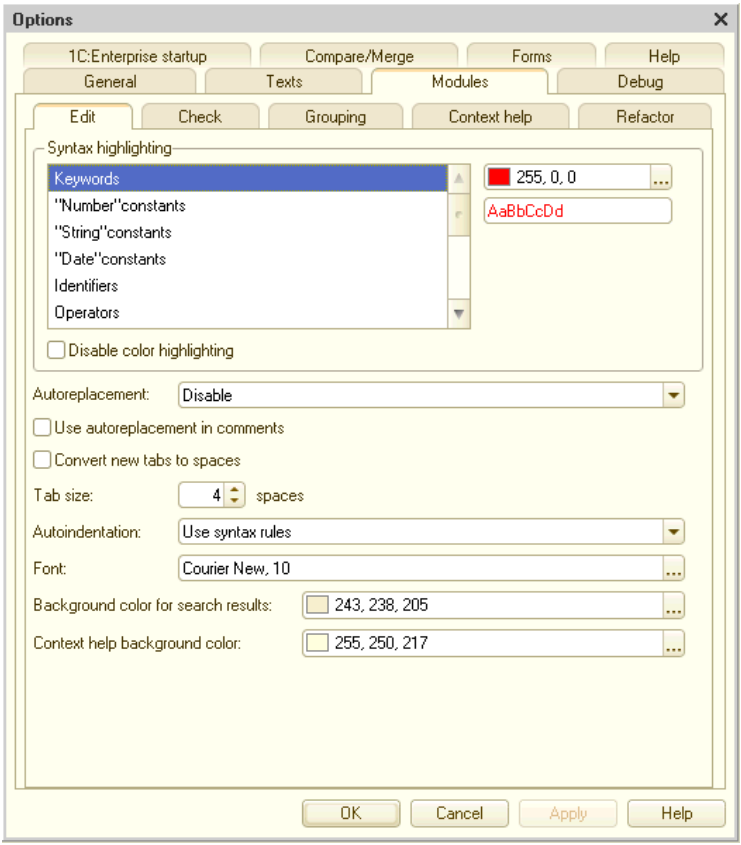


Fig. 468. Module texts editing settings

Syntax highlighting. It's easier to perceive module text if certain syntactic constructs (constants, names, operators, comments etc) are highlighted. Color is also used to check whether module text has been entered correctly. If you enter the operator's name incorrectly, it won't be highlighted with a required color. You can choose a different color for every construct (constant, name, operator, comment, etc).

Among syntactic constructs that can be color highlighted, let's individually discuss Current ID, Selected ID and Block borders.

Color, selected for the Current ID construction is used to highlight all uses of any ID, with cursor for text editing "inside". No additional actions are needed for highlighting. You just need to point the cursor on the ID.

Color selected for the Selected ID construction is used for selecting all uses of an ID, that is fully selected (by double clicking or otherwise).

TIP. If you don't need to automatically select all the IDs with a cursor "inside" of them, then you can select a construction's Current ID colour, that matches to the construction IDs. After that, it will be possible to see the uses of any ID by selecting it.

When highlighting IDs, string constants, that follow the rules of IDs forming, are going to be highlighted as well.

NOTE. Default colors, selected for the constructions Current ID and Selected ID, match the background color and need additional set-up.

Color, selected for Block borders construction is used for highlighting paired brackets (), (,], [) and keywords, that are used together, for example `Procedure ... EndProcedure`, `If ... Otherwise ... EndIf` etc. Highlighting is done only if the cursor is "inside" a keyword or a bracket.

Disable highlighting. If set, module text is displayed as ordinary text and the color highlighting setting is unavailable.

Auto-replacement. If this property is enabled, the entered text that corresponds to the text in the Automatically replace string attribute of the template is automatically replaced with template text. If the Enable and show tooltips property is enabled, the template text is displayed after a brief pause while the text to be replaced is being entered. If the property is set to Only tooltip, tooltip text is shown after a brief pause when you type the replaceable text.

To replace input text with a template text, press Space or Enter after typing a string. If the Auto-replacement property is set to Disable or Only tooltip, these actions are not performed.

Use auto-replacement in comments. If set, template text is displayed when you enter a text to comments, that matches the template text.

Replace tabs when typing. If this check box is selected, the typed tabulation symbols will be replaced with the number of spaces specified in Tab Width property.

Tab Width. Specifies the number of Space characters equivalent to one tab character.
Auto-indentation. Sets or removes auto-indentation when entering the line break symbol (pressing the Enter key). If Syntax value is selected, module text input is performed according to the current syntactic construction.
Font. Selecting a font for text input. We recommend using a fixed-width font for module text input (for example, Courier New).
Background color for search results allows to set a background colour for search results in text editor on the 1C:Enterprise language.

37.2.3.2. Setting Module Check Parameters

The Check tab is used to set module check modes.

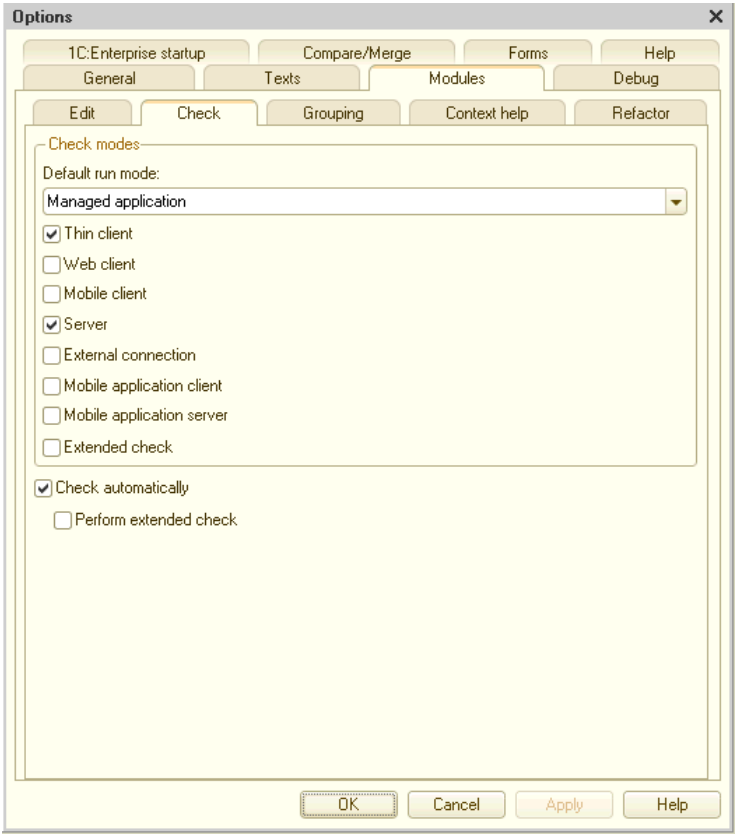


Fig. 469. Module Syntax Check Settings

Module check settings are used to select a suitable environment for compilation. Execution environment includes common modules with a corresponding flag and modules that are used in particular executive modes (managed application module, session module etc.). If executive environment compilation is successful, the module that initiated syntax check is verified. Moreover, the number of performed check passes (including executive environment compilation) is equal to the number of check marks set in dialog box. Modules that don't exist in the selected check mode, for example, application module, will not be checked if you selected the Thin client check mode.

Availability of the Default run mode drop-down list in the given form depends on the Edit configuration for startup modes parameter. If the parameter value is set to Managed application, the drop-down list is missing from the form, moreover, in this case the Thick client (ordinary application) parameter is hidden from the dialog box. When opening the dialog box, the value of this parameter always matches the Default run mode configuration property value.

If the Default run mode drop-down list is available in the form, you can set-up check modes for each run mode. By default, when choosing Ordinary, only the Thick client (ordinary application) check box is selected; when choosing Managed application mode, - Thin client and Server check boxes are selected. Also, you can set-up a new set of check boxes for each run mode and press Apply or OK. Moreover, after Edit Configuration for startup modes parameter changes value, Designer will restore the required set of check modes.

Preprocessor commands are defined depending on the selected check mode:

Check mode	**Preprocessor command**
Thick client (ordinary application)	`Client`, `AtClient`, `ThickClientOrdinaryApplication`
Thin client	`ThinClient`, `Client`, `AtClient`
Mobile client	`MobileClient`, `Client`, `AtClient`
Web client	`WebClient`, `Client`, `AtClient`
Server	`Server`, `AtServer`
External connection	`ExternalConnection`

If the Check automatically check box is selected, module text will be automatically checked for syntax errors (when recording or closing). If the Including extended check check box is selected, calls to object properties and methods using a dot are verified for a limited number of types. Also, a check is conducted to determine whether the settings specified by a string are correct for a limited number of methods.

37.2.3.3. Setting group parameters

The Grouping tab is used to set-up grouping and folding modes for various syntactic constructs. The table lists syntax types of constructions by lines.

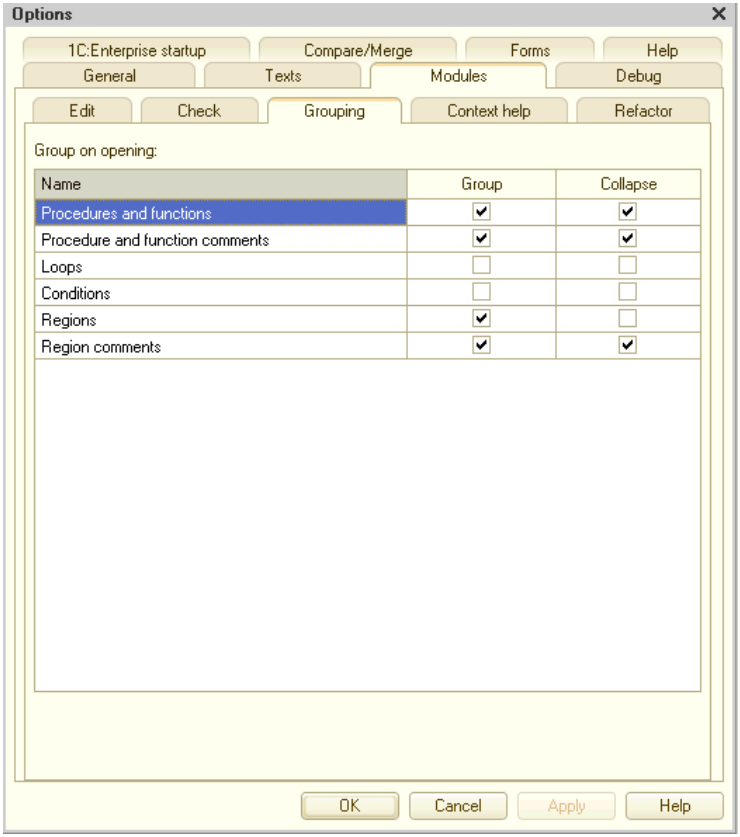


Fig. 470. Group settings

Selecting a check box in the Group column, will cause the syntactic construct to automatically group. Note that only syntax structures of the second nesting level can be grouped.

If the Collapse check box is selected, the syntax structure will automatically collapse when a module is opened.

37.2.3.4. Context tooltip settings

The Context tooltip tab is used to specify the settings for opening the context tooltip when module text is being entered.

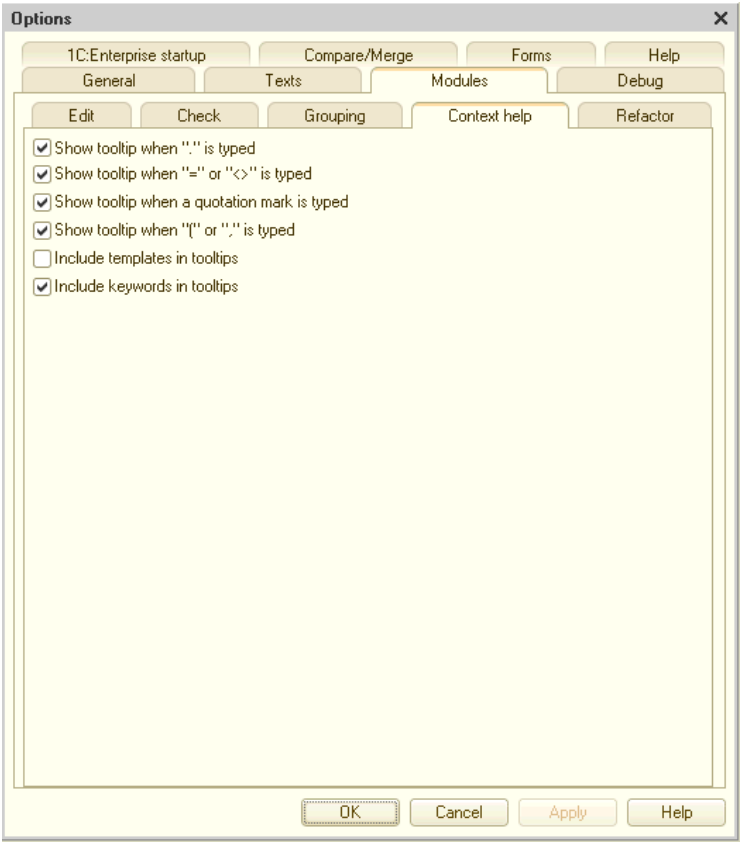


Fig. 471. Context tooltip settings

If the Show tooltip when "." is typed check box is selected, a context tooltip is automatically displayed when you type a dot ".".
If the Show tooltip when "=" or "<>" is typed check box is selected, a context tooltip is automatically displayed for system enumeration value selection when you type the characters "=" and operator "<>".
Selecting the Show tooltip when a quotation mark is typed check box enables the context tooltips for parameters of some methods, if the parameter is set by a string.

Selecting the Show tooltip when "(" or "," is typed check box enables the context tooltips for parameters of platform's procedures and functions, including application procedures and functions, with comments designed in a specific way.
If the Include templates in tooltips list check box is selected, templates with auto-replace strings defined are listed.
If the Include keywords in tooltips list check box is selected, all keywords (such as `If`, `Procedure`, `Loop`, `Return` etc) are included in the list.

37.2.3.5. Refactoring settings

On the Refactoring tab, you can set-up platform behavior when opening legacy synchronous calls search.

The Convert modal calls check box selection defines whether the refactoring feature will process the modal method calls with dialog boxes or not.
The Convert synchronous calls of extensions and add-ins check box selection defines whether the refactoring feature will process synchronous operation methods for files, extensions and add-ins.

37.2.4. Debug

On this tab you can set-up which protocol will be used for application debugging. If you use the HTTP protocol on this tab, the parameters will be set for this protocol.

In this dialog box, the Designer prompts some default values, depending on the attached infobase. However, the developer can set-up these parameters individually, if required.
Debug server group allows you to specify where the debug server to be used is located.

- Use local debug server. Mainly used to debug using the file mode of the infobase. If several network cards are installed on the computer, you should specify which network card will be used for debugging. This is done by filling in the IP address of the required network card in the Local NIC address field. In the Port ranges field, you can fill in the range of the network ports that will be used for debugging. A port that to be used by the debug server will be selected from the specified port range (ranges). The port range is filled in using the " :" symbol, the ranges are separated using the "," symbol. Example: 1560:1591, 2560:2591, 7700-8000.
  In this case, when starting the application to be debugged, a debug server will be started and it will manage the debug in this infobase.

This mode is automatically selected for file mode infobase operations.

- Use remote debug server. In this case, the Designer will try to connect to a remote debug server, it's address and port shall be filled in the Remote debug server address property.
- Use cluster debug server. In this case, Designer will use the connected debug server deployed in a server cluster, to perform debugging. To enable debug in server cluster, you need to start this cluster in debug mode, using the /debug keyword. In this case, when starting the application to be debugged, a separate debug server won't be started.
  This mode is automatically selected when working with client/server mode infobase

Infobase name group is used to specify the name of the infobase, under which the debugger will register on the debug server, specified in Debug server group:

- Default. Will be selected in case Designer uses the file mode of the infobase. In this case, the DefAlias name will be always used.
  Current infobase name ("name"). Will be selected in case Designer uses the client/server mode of the infobase. The infobase's name used in server cluster will be used as the infobase name.
- Use specified infobase name. If required, you can fill in a custom name in the infobase name field. This set-up might be required when you need to debug the configuration according to data of a different client/server infobase with a similar configuration.

Access group is used in case the debug server is password-protected. Password to access the debug server to be used, needs to be filled in the Password field.
After changing the settings on this page, you will need to restart Designer.

37.2.5. 1C:Enterprise startup

37.2.5.1. General parameters

On this tab, you can select the application to be started, and user selection when starting 1C:Enterprise.

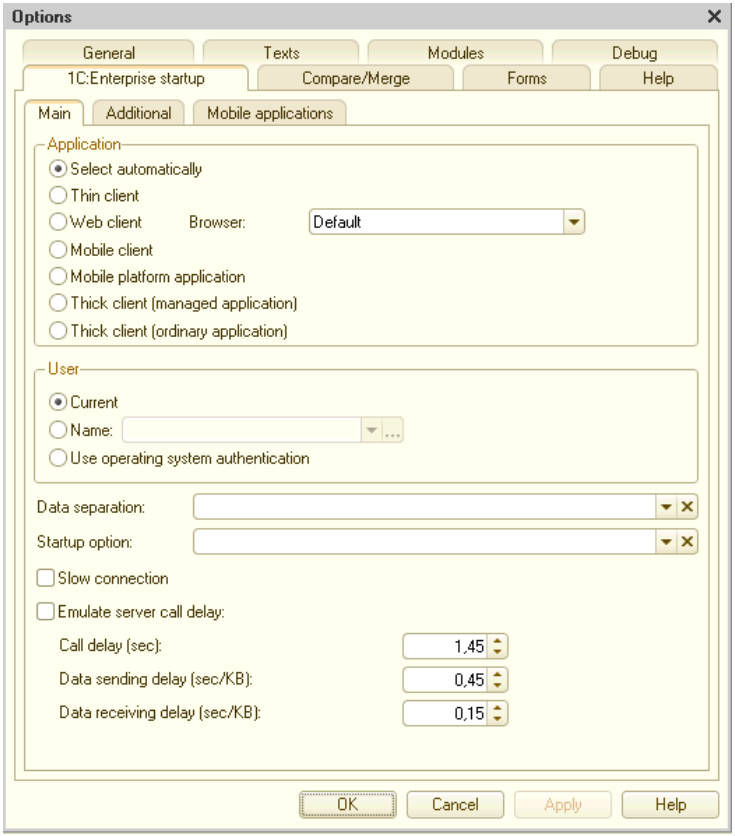


Fig. 472. 1C:Enterprise startup parameters

When starting 1C:Enterprise from Designer, the client to be loaded is specified in Application group.

If the mode is set to Select automatically, the 1C:Enterprise selects the client type depending on the configuration Default run mode and Run mode property of the user that starts the 1C:Enterprise. If configuration properties and user properties are not specifically indicating something, then:

- if the Default run mode property is set to Ordinary, thick client will be started in ordinary mode;
- if the Default run mode property is set to Managed application, thin client will be started.

If a specific client type startup is required at 1C:Enterprise startup from Designer, you need to set it manually (items under Select automatically). When specifying the start of the web, you can specify the web browser that will be started:

- default web browser,
- Microsoft Internet Explorer,
- Mozilla Firefox,
- Google Chrome,
- Safari.

When starting the web client from Designer, the platform version, with an uploaded earlier infobase, is checked. If the versions are different, the user receives a message suggesting to perform the publication again. If the answer is positive, an infobase publication dialog box is displayed.

If a mobile client is selected as a client application, you can specify the mode in which a mobile client is operated: in the mobile client (Standalone Mode is set to Do not use) or offline mobile client mode (Standalone mode is set to Use). If Standalone mode is set to Auto, standalone mode will be used, if an application has standalone configuration components duly assigned.

The user that starts the 1C:Enterprise from Designer is defined in User group:

- Current -from the user list.
- Name - a specific user is selected (the /N command-line parameter analogue). The required user can also be selected from the list. In order to do so, use the "…" button, located on the right side of the field.
- If the Use operating system authentication - check box is selected, the user will be selected using the operating system authentication (the/WA command-line option analogue).

Data separation - allows to specify the separator's values in a way that is used to specify them for the /Z parameter. The string specified in this field must match the specifying separators values rule, however, it must not have quotation marks framing the values, when specifying these values in the /Z parameter. For example, when specifying the following parameter in command-line: /Z "-1, +4" in the Data separation field should be specified with-1, +4. 1C:Enterprise stores different options for separator's values input and allows switching between them.

Startup parameter - startup parameter /C command-line parameter analogue) that is available for processing through the global context property LaunchParameter.

Low connection speed - allows to enable or disable the low connection speed mode at thin client startup. Check box selection is saved for the specified infobase. The check box is cleared for an infobase with slow connection speed mode startup not set at the base list. If the check box is selected, 1C:Enterprise starts using slow connection speed.

For an infobase with a slow connection speed mode startup set in base list, the check box is displayed in gray. If the check box is cleared, 1C:Enterprise starts using normal speed. The check box is the same as /O command-line parameter.

Emulate delay on server calls - the check box enables emulation of the thin client operation under the conditions of significant time delays, that are caused by interactions with the server. This feature is turned off by default.

Call delay, sec - defines the delay duration for each server call by a client application.

Data sending delay (sec/kB) - defines the delay duration when transferring data to the server. Time is indicated by 100 bytes of transfered data.

Data receiving delay (sec/kB) -defines the delay duration when receiving data on the server. Time is indicated by 100 bytes of received data.

NOTE. Default values allow GPRS connection emulation.

Server calls delay emulation mode can be enabled using the command line: /EmulateServerCallDelay [-CallXXXX] [-SendYYYY] [-ReceiveZZZZ].

37.2.5.2. Additional parameters

Additional parameters on this tab can be helpful when developing applications (in particular, for debugging).

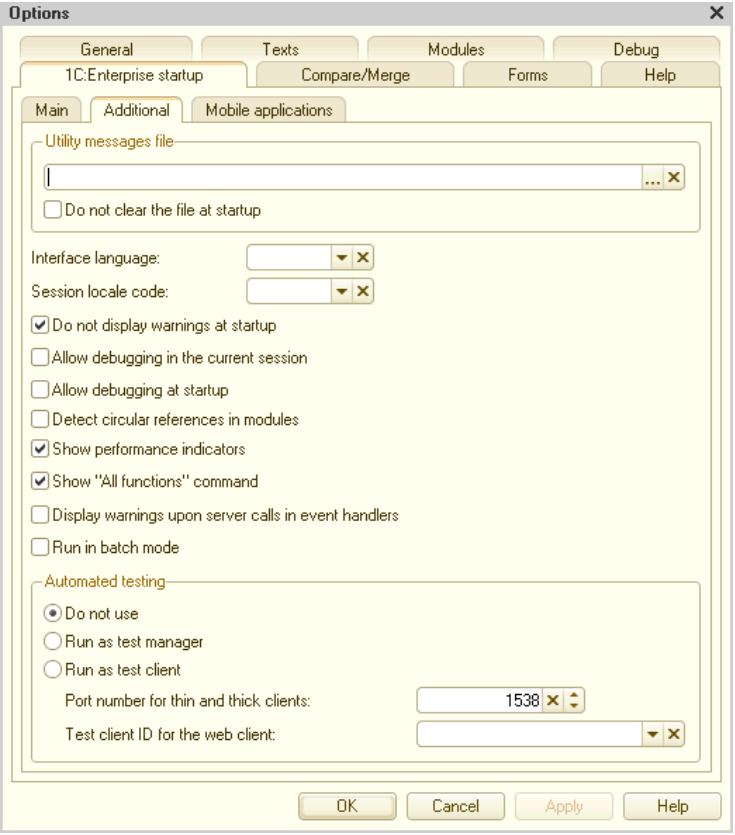


Fig. 473. Additional 1C:Enterprise 8 startup parameters

Service messages file -to save service messages to a file automatically enter its name in the parameter (same as /OUT command-line parameter).

Do not clear file before startup -If this check box is selected, during the next startup, messages will be added to the specified file (/OUT command-line parameter with NoTruncate key analogue).

Interface language - is used to specify the national language of the interface (/L command-line parameter analogue).

Session localization code -is used to specify the session locale code (/VL command-line parameter analogue).

Do not display warnings on startup -if this check box is selected, the following startup messages are suppressed:

- Database configuration does not match stored configuration. Continue?
- Your computer capabilities are inadequate to edit configuration help. To edit help, install Microsoft Internet Explorer version 6.0 or higher.
- Your computer capabilities are inadequate to edit HTML documents, including help sections. To edit HTML documents, you must install Microsoft Internet Explorer version 6.0 or higher. HTML document editing will be unavailable during this session (parameter passed through the /DisableStartupMessages command-line key analogue).

Allow debugging mode -If the check box is selected, debugging is available when the application is started in 1C:Enterprise mode (analogue of the parameter specified in /debug -tcp or /debug -http command-line parameter).

Start debugging on startup- if the check box is selected, when starting 1C:Enterprise connection is performed automatically. When using HTTP protocol to debug, this setting is the same as when using the /debug - http -attach command line key.

NOTE. If 1C:Enterprise starts from Designer using the Debug - – Start Debugging menu item, the debugging mode is enabled automatically (regardless of Enable debugging mode and Start debugging on startup check boxes).

Check 1C:Enterprise language circular references-if this check box is selected, the client application (thin or thick client) will check the circular references when operating with 1C:Enterprise language, when started from the Designer. This check box is an analogue of the command line /EnableCheckScriptCircularRefs key.

Display performance indicators- if this check box is selected, a button is added to Favorites and History panel when starting 1C:Enterprise. When clicking the button, a window with current statistics on interaction with the 1C:Enterprise

server is displayed (analogue of the /DisplayPerformance command line parameter).

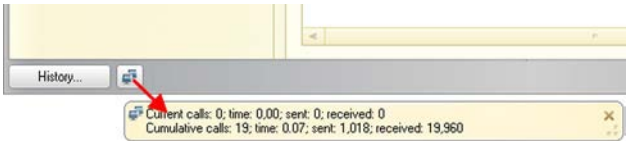


Fig. 474. Display server calls

Display All functions command - displays Service and Settings -All functions (see fig. 475) and is an analogue of /DisplayAllFunctions command line parameter. This command call displays a form with a list of all applied objects you can use (used in managed mode of thick client as well as thin and web client). The menu command is turned off by default.

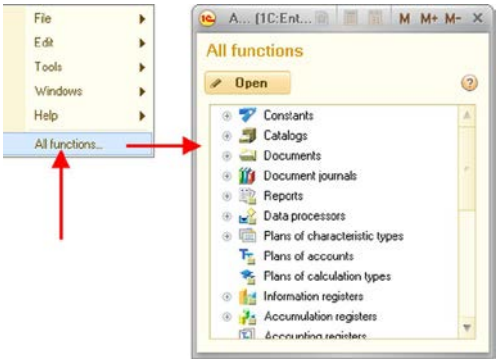


Fig. 475. "All functions" form use

If All functions command is enabled then the work with this window looks as follows:

1. All functions form is opened with the help of Service and Settings- - All functions command.
2. The necessary configuration object is selected from the configuration objects tree.
3. The selection of the necessary objects with the help of All functions form search string is an alternative variant of step 2.
4. The double-click on the selected object of configuration or pressing Open button is the last step. The selected form will be opened after this step is performed.

The search string is activated by Ctrl+F key combination or as of text typing start in the configuration object tree. When the search string is used the configuration object tree is switched to the flat list (without hierarchy) mode and - returns to the tree mode when the search is aborted. The search with the help of this mechanism selects the configuration objects and forms which match the search string for display. No other objects will be displayed in the list. The search is aborted with the help of Ctrl+Q key combination.
The search is performed by representation and uses the following algorithm:

- Search is carried out by the beginning of the word
- If a word contains letters and digits:
  - sequence of letters is a separate word;
  - a sequence which contains digits only is deemed a separate word.
- In a word consisting of digits, leading zeros can be both significant and insignificant simultaneously.
- An item is deemed to be found (including if this item contains subordination) if one of the following conditions is met:
  - the representation contains all words searched for;
  - the item representation contains at least one searched word and its parents contain the remaining words.

Thus, if all registers which representation contains the word "sales" is to be found then the query "sales register" can be specified in the search string with (for Commerce version 11.4.3) the following result:

A usual search window (with the help of Alt+F key combination) can be opened if a search in the displayed information of the tree (group names and open tree branches) is necessary. In this case, the cursor will be on the first occurrence of the searched string and F3 (go to the next search result) and Shift+F3 (go to the previous search result) can be used to move in the search results.

NOTE. Forms opened from the All functions window, always appear in the auxiliary window of the application.

Also, in the All functions window, Standard functions used for execution of some standard procedures are available: viewing the active users list, deleting marked objects etc.
Check server calls in event handlers- if this check box is selected, the command line startup parameter `/EnableCheckServerCalls` is added (from Designer) to the command line of 1C:Enterprise startup.
Selecting this check box in the startup settings is recommended if the application can potentially have handlers of form events, where context server calls are performed - using them in this kind of handlers is not allowed. Breaking this rule can lead to errors in some cases of server calls usage. Mainly problems occur, if

the form items or its attributes change in such handler while in server call. As a result, unexpected behavior or non-obvious execution time errors may occur.
Run in a batch mode - allows launching the client application in the batch mode from the Configurator.
The Automated testing group is used to define the role (in terms of automated testing) of started client application when starting from the Designer:

- Do not use -started client application is not used in automated testing;
- Start as testing manager -the client application to be started will be used as the testing manager (the `TestManager` command line key will be used);
- Start as testing client -the client application to be started will be used as testing client (the `TestClient` command line key will be used). For interaction with testing manager, the port specified by the Port number for thin and thick client (`TPort` parameter for `TestClient` command line key) parameter will be used. For the web client, an ID specified in ID for web client parameter is used (`TestClientID` command line key) will be used.

37.2.5.3. Mobile application

This tab is used to set-up the parameters required for 1C:Enterprise, in order to operate with Android SDK.

Use "Android debug bridge" to start a mobile application check box defines the method to be used for sending the configuration to the mobile device:

- Check box selected: Android Debug Bridge (ADB) is used to send configuration to the mobile device.
- Check box cleared: Send configuration using a web server schema is used.

Android SDK directory- defines the path to the installed Android SDK. This parameter should be specified **before** performing any operations using Android Debug Bridge.
Mobile platform- contais a link to the mobile platform archive. This link will be used on the mobile device (or emulator) The archive must have an Android directory, with the following files in it:1cem-arm.apk(mobile platform for ARM architecture developer) and 1cem-x86.apk (mobile platform for ARM architecture developer).
The No devices available parameter group defines Designer's behavior when no mobile devices are connected to the computer where Designer is operating. Or when an emulator is not started on this computer.

- Wait for connection- in this case, the Designer will wait for a physical device connection or an emulator strart. Waiting is performed with no timeout.
- Start device emulator-in this case, the Designer will try to start an emulator specified in the nearby field. Emulator is created using the Android virtual device manager (included in Android SDK).
- Refuse starting-this will interrupt the performance of the action.

HyperlinkVirtual device manager is used for starting the AVD Manager application from the Android SDK directory.

37.2.6. Comparison / merging

This tab is used to set-up the parameters of add-in, that can be used to compare/merge modules.

The programs that can be used for comparison/merging of files are listed in the Add-ins list.

The parameters for the following Designer add-ins are set by default:

- Araxis Merge
  - OS Windows
  - Website: http://www.araxis.com/merge/;
  - Executed file: compare.exe;
  - Supported modes:
    - Comparison of two files;
    - Merging two files;
    - Trilateral merging.
- KDiff3:
  - OS Windows, Linux;
  - Website: http://kdiff3.sourceforge.net/;
  - Executed file: kdiff3.exe (OC Windows), kdiff3 (OC Linux);
  - Supported modes:
    - Comparison of two files;
    - Merging two files;
    - Trilateral merging;
    - Automatic trilateral merging;
- TortoiseMerge:
  - OS Windows
  - Website: http://tortoisesvn.net/TortoiseMerge.html;
  - Executed file: tortoisemerge.exe;
  - Supported modes:

    - Comparison of two files;
    - Trilateral merging.
- Perforce P4Merge:
  - OS Windows, Linux;
  - Website: http://www.perforce.com/product/components/per-force-visual-merge-and-diff-tools;
  - Executed file: p4merge.exe (OC Windows), p4merge (OC Linux);
  - Supported modes:
    - Comparison of two files;
    - Trilateral merging.
- DiffMerge:
  - OS Windows
  - Website: https://sourcegear.com/diffmerge/;
  - Executed file: sgdm.exe;
  - Supported modes:
    - Comparison of two files;
    - Trilateral merging.

However, to use add-ins an additional set-up is required- you need to specify the path to the executable file.

Modules group is used to specify which mechanism will be used for executing a specific operation:

- Show differences without structure- is used to select the mechanism, that will be used when using a corresponding command in the configuration comparison/merging window. Selecting the Ordinary method item means, that the Designer's built-in mechanism will be used for comparison of two files' contents. Also, add-ins with specified Path and Compare two files properties, will be available for selection.
- Merging settings when merging two configurations- is used to select a mechanism that will be used for merging two configurations. Selecting the Ordinary method item means, that the Designer's built-in mechanism will be used for merging two files' contents. Also, add-ins with specified Path and Merge two files properties, will be available for selection.

Supported configuration update group describes 1C:Enterprise behavior, when updating a supported configuration.

- "Take from", "Merge with priority" modes- defines what will be used for interactive merging. Selecting the Ordinary method item means, that the Designer's built-in mechanism will be used for merging (two-way). Also, add-ins with filled-in Path and Trilateral merging properties, will be available for selection.

- Merge using add-in mode- is used to specify which add-in will be used for trilateral merging and allows to manage the availability of the Merge using add-in mode in the in the configuration update settings. Add-ins with Path and Trilateral merging properties filled are available for selection.
  If this property is set to Not used, then the Merge using add-in feature disappears from the merging mode settings. When saving Designer settings all merging modes, where the use add-in property is set, are replaced with take from new vendor configurationmode (if the update main configuration window is open).
  If an add-in is specified in this property, and the use by default check box is selected, when saving settings the Merge using add-in merging mode is selected.

It is possible to add a native file content comparison utility. To do so, press Add button and specify all the parameters in the opened dialog box.

The Name field is used to specify the presentation of the comparison program. The path to the utility is specified in Executable file field. If the path to an executable file is not specified, this add-in is displayed in gray color in the Add-ins list and these programs cannot be selected in other comparison and merging settings. On the**Ошибка! Источник ссылки не найден.** DiifMerge, TortoiseMerge and Perforce P4Merge are displayed in gray color. For the Araxis Merge and KDiff3 programs, a path to the executable file is specified.
In the following fields (Command line group) the command line parameters are defined for the add-in that is used for any operation from Designer:

- Comparison of two files- two files contents are interactively compared;
- Merge two files- two files contents are interactively merged (with a feature that allows modification of the acquired result);
- Trilateral merging- three files contents are interactively merged (with a feature that allows modification of the acquired result);
- Automatic trilateral merging- automatic merging of three files contents. Is not performed interactively if there are no conflicts when merging.

The command line parameters that will be passed to the add-in for executing operations are set for each field with an identical name. If an operation is not supported by the utility -the corresponding field is left blank.
When forming the command line you can use the following predefined parameters:

- `%basecfg`- path to file from the main configuration;

- `%secondcfg`- path to file from the second configuration. The second configuration can be:
  - Configuration from file;
  - Vendor configuration (new vendor configuration if updated);
  - Configuration from configuration repository;
  - Database configuration.
- `%oldVendorCfg` - path to file from the previous vendor configuration;
- `%merged` - path to the resulting file (merging result);
- `%baseCfgTitle` - window title for the file from the main configuration;
- `%secondCfgTitle`- window title for the file from the second configuration;
- `%oldVendorCfgTitle` - window title for the file from the previous vendor configuration.

37.2.7. Forms

This tab is used to configure the display parameters of the initial form editor preview window.

The Use assignment and Interface parameters determine in which option the preview will open when the form editor is opened. If the Use assignment parameter is set to Application for platform, then the Interface parameter will allow you to select which application interface will be displayed in the form editor's preview window. If the Use assignment parameter is set to Application for mobile platform, then the interface cannot be selected and the preview window will always display a preview of the form for the mobile platform.

Personal computer group contains parameters that are relevant for previewing a form intended for use on a personal computer:

- Scale- defines the default scale value.
- Resolution - defines the default resolution of the preview panel.
- Rotation - defines which screen orientation will be used by default:
  - Auto - the horizontal orientation is used.
  - Horizontally, Vertically- indicates the corresponding orientation of form.

Mobile device group contains parameters that are relevant for previewing a form intended for use on a mobile device:

- Mobile device - allows to specify the mobile device that will be used if the interface of the mobile device is selected on the preview panel.

- Scale option - allows to specify the scale option that will be used by default.
- Rotation - defines which screen orientation will be used by default:
  - Auto - uses the orientation that matches the settings of the selected mobile device.
  - Horizontally, Vertically- indicates the corresponding orientation of form.

If a device is missing from the list of existing mobile devices, you can add it for use in the preview. To do so, press the Devices list button (located right from the drop-down list of mobile devices). This list will display characteristics of mobile devices added by the user themselves. In the selection list (in setting dialog box or in form editor) user devices will be located in the begging of the list in order specified in Additional device description list.

To add a new device, press the Add button.

In the Device Description form, the parameters of the device being added are set:

- Name -how the device will be displayed in the list.
- Operating System- allows to specify, which mobile OS is installed on the device: iOS, Android, Windows Runtime.
- Screen height, Screen width - indicates the screen resolution of the mobile device for default device orientation.
- Screen DPI - indicates the pixel density on the device.
- Use on-screen buttons - is used in the preview panel for a more accurate device display.

37.2.8. Help

The tab is used to set-up the display type of various help information.

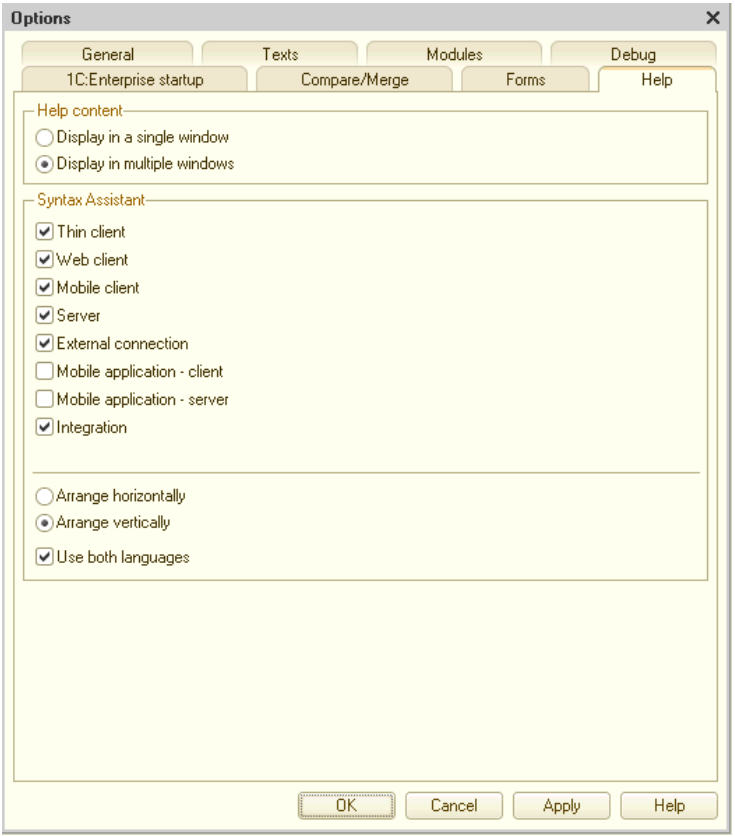


Fig. 476. Help information display settings

Help info can be displayed in one window or split into several independent windows (Display in one window/Display in different windows parameters).
The Syntax assistant can be arranged horizontally or vertically (Arrange horizontally/Arrange vertically parameters) and use two languages for terms (Use both languages parameter).
The filter of displaying the contents tree and Syntax Assistant index can be set up by object availability for a specific 1C:Enterprise start mode. For example,

Syntax Assistant sections can be displayed only for objects available in thin client.
Display deprecated objects check box is applied when it is necessary to display the deprecated objects in the Syntax Assistant objects tree. In this case, "object" means any configuration object property, function, procedure, object, etc. "Deprecated object" means that the object can't yet be applied or is already not recommended to be used in this compatibility mode. Let's explain the said by examples:

- `StrTemplate()` method was added to the platform in version 8.3.6 but this method is not accessible in the mode of compatibility with version 8.3.5. It means that if the Dispaly deprecated objects box is not checked and the configuration compatibility mode is set in the value Version 8.3.5 (and any preceding version) then `StrTemplate()` method will not be displayed in Syntax Assist tree.
- `BackgroundJob.WaitForExecutionCompletion()` method is announced outdated in the platform of version 8.3.13. It means that if the configuration compatibility mode is set in the value Version 8.3.14 (and subsequent) this method will not be displayed in the Syntax Assist objects tree.

The Syntax Assist tree is rebuilt when the configuration compatibility mode is changed.
This tab can be accessed by using the command bar button in the Syntax Assistant.

37.3. Using calculator

Operations with calculator are described in Calculator section in 1C:Enterprise 8.3 documentation. User Manual.

37.4. Calendar

For description of calendar features, see the Calendar section in 1C:Enterprise 8.3 documentation. User Manual.

37.5. Text templates

The Designer is used to save frequently used text fragments and perform the fast insert operation of cashed fragments in an edited text document or a module.

Text fragment is assigned a conditional combination of characters - short fragment name, called template. If auto-substitution mode is enabled in Designer parameters, required text fragment is inserted automatically in editor when inputting a template.

IMPORTANT! Auto-substitution mode is enabled separately for modules and text on different tabs of Designer parameters settings.

37.5.1. Template list maintenance

Templates are stored in template files. You can select **standard templates** shipped with 1C:Enterprise and **user** templates generated by the user. Standard templates are stored in the bin directory of the installation directory for a particular 1C:Enterprise version. User templates can be stored at any location on the hard drive. The list of attached templates is saved for a 1C:Enterprise language option- you can generate different attached templates sets for different 1C:Enterprise language versions. Standard templates are provided for both 1C:Enterprise language options.
Templates are configured in the Text templates window in a mode that can be called from Service -Text templates.

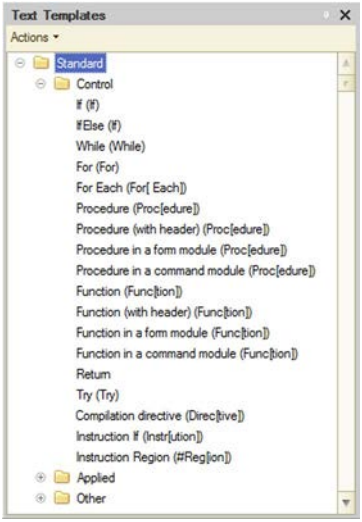


Fig. 477. Text templates

To manage templates select Actions -Template settings in the Text templates window. The Templates settings is displayed.

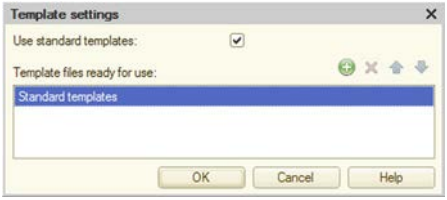


Fig. 478. Template settings

To add standard templates select the Use standard templates check box. De-selecting this check box disables using of standard templates. Each 1C:Enterprise version uses its own set of standard templates. In addition to standard files you can attach user templates. To do this, click the Add button on the tools panel of Template settings and select the required template files.

TIP. It is recommended to store user templates in a directory different from the 1C:Enterprise installation directory. For example, you can store templates in a specific subdirectory of the My documents directory.

To rearrange files within the list, use the Move up and Move down buttons.

To delete an unwanted template, select it in the file list and click the Delete button.

37.5.2. Editing a template

To edit an existing template file use the Actions -Change item. The template editing window containing templates located in the file will be displayed.

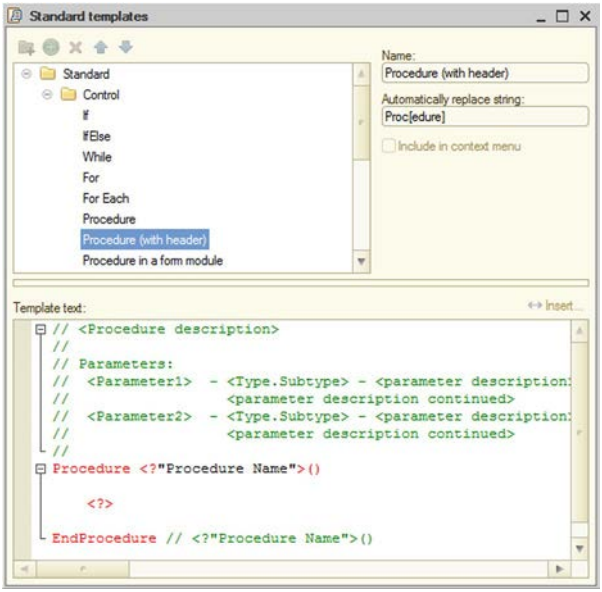


Fig. 479. Template editor

IMPORTANT! Standard templates editing is unavailable. They can only be viewed in the template editor.

In the Name attribute, the template name is displayed as it is shown in the list of templates. In the Automatically replace string attribute, the string sequence is specified, after typing it, Designer allows you to replace it with a template. Replacement is initiated by pressing the Spacebar or Enter key.

Also, when a string is typed, a tooltip containing template text is displayed on the screen as shown on the figure below.

Fig. 480. Use of templates

The template editing window contains a tools panel that can be used to create template groups and separate templates, delete unwanted templates and arrange the list. The tree of templates is located in the left part of the window. In the right part, the template group name and the template itself are edited.

For convenience, templates can be grouped into folders. To create a group, use the Add folder command from the context menu or press the appropriate button on the tools panel. Enter a group name in the Name attribute.

To create a template, select a group where it will be located and use the Add command from the context menu or press the appropriate button on the tools panel. Specify the template name in the Name attribute (this name is used to select a template for inserting into edited text). In the Automatically replace string field, you need to specify the initial part of the string and when it is typed, it will be replaced with template text in a text document window (if automatic substitution is on). To use automatic substitution, press Enter or Spacebar key, after entering a sequence of characters. Use the Template text multiline field to enter text that will replace the text specified in the Automatically replace string field or to be saved in document text when a template is selected.

Select the Include in context menu check box to make a template available for interactive selection.

A template for insertion in text (module) can be selected from the list of templates in the context menu. Templates are placed in the list, if the Include in context menu check box is selected.

37.5.2.1. Template controls

Custom query

Description:

Is used to enter any text. When using this template, a dialog box for inputting context-dependent template part is displayed.

Syntax:

```
<?"Tooltip">
```

Parameters:

<Tooltip>

Explanatory label text.

Cursor positioning

Description:

The construct is used to position the cursor to the indicated location after inserting template text.

Syntax:

```
<?>
```

Specific Query

Description:

The construct is used to enter configuration objects, predefined items and other data.

Syntax:

```
<?"Tooltip",<Keyword>[,<Parameter1>[,<Parameter2>...]
]
```

Parameters:

<Tooltip>

Explanatory text in the entry query.

<Keyword>

Type of query.

<Parameter1>...

Query parameters.

Descriptions of specific queries are listed in the following table:

Russian option	**English option**	**Description**
BusinessProcess	BusinessProcess	Selecting a business process type
OptionChoice <Tooltip 1>, <String to insert 1>, … , <Tooltip N>, <String to insert N>	OptionChoice <Tooltip 1>, <String to insert 1>, … , <Tooltip N>, <String to insert N>	When using this template, a list of strings for selection is displayed. Each string in the list consists of a tooltip and insertion text. Example: "Tooltip 1" - parameter describing the text of the first string in the list; "String to insert 1" - parameter or text that will be inserted with selecting the first string
TypeChoice	TypeChoice	Type selection
Document	Document	Document type selection
DocumentJournal	DocumentJournal	Document journal type selection
Task	Task	Task type selection
EnumerationValue	EnumValue	Enumeration value selection
UserName	UserName	Username entry
TypeDescriptionConstructor	TypeDescriptionConstructor	Type description wizard
UserFullName	UserFullName	Full username entry

Russian option	**English option**	**Description**
Configuration-StorageUserName	Configuration-StorageUserName	Entry of full configuration repository username
Constant	Constant	Constant selection
FilterCriterion	FilterCriterion	Filter criterion selection
Data Processor	DataProcessor	Data processor selection
Report	Report	Report selection
Recalculation	Recalculation	Recalculation selection
Enum	Enum	Enumeration selection
ChartOfCalculationTypes	ChartOfCalculationTypes	Chart of calculation types selection
ChartOfCharacteristic-Types	ChartOfCharacteristic-Types	Chart of characteristic types selection
ExchangePlan	ExchangePlan	Exchange plan selection
ChartOfAccounts	ChartOfAccounts	Chart of accounts selection
ChartOfCalculation-TypesPredefinedData	ChartOfCalculation-TypesPredefinedData	Selection of predefined data for a chart of calculation types. First, a chart of calculation types is selected and then predefined data value

Russian option	**English option**	**Description**
ChartOfCharacteristic-TypesPredefinedData	ChartOfCharacteristic-TypesPredefinedData	Selection of predefined data for a chart of characteristic types. First, a chart of characteristic types is selected and then predefined data value
ChartOfAccountsPredefinedData	ChartOfAccountsPredefinedData	Selection of predefined data for a chart of accounts. First, a chart of accounts is selected and then predefined data value
MetadataObject	MDObjectsSubset	Metadata objects selection. When creating a template, a dialog box opens where you need to select types of metadata objects. When selecting a template, a selection dialog box is displayed, containing all metadata objects of the specified types
Sequence	Seq	Sequence selection
AccountingRegister	AccountingRegister	Accounting register selection
AccumulationRegister	AccumulationRegister	Accumulation register selection

Russian option	English option	Description
CalculationRegister	CalculationRegister	Calculation register selection
InformationRegister	InformationRegister	Information register selection
Catalog	Catalog	Catalog selection
CatalogPredefinedData	CatalogPredefinedData	Selection of predefined data from the catalog. First, a catalog is selected and then the predefined data value
QueryText	QueryText	The query text is entered using the query wizard
DateTime, <Format string>	DateTime, <Format string>	Entry of the current date in the format specified in the <Format string> parameter
FormatString	FormatString	The format string text is entered using the format string wizard

37.5.2.2. Template text editor

Structurally, a template can consist of a static (unchangeable) part and a dynamic part which content in context-dependent and cannot be specified in advance. For example, the `If` operator of the 1C:Enterprise language has the following structure:

```
If<?"Condition 1"> Then
ElseIf <?"Condition 2"> Then
EndIf;
<?"Condition...">
```

These are context-dependent template parts.

For versatility of template use, the control directive insertion mechanism is applied. Control directives are inserted into the template text - character sequences that initiate execution of some actions when inserting the replacing template text.

For example, control directives allow to request information from the user and to add this information to the text to be inserted.
Control directives can be inserted into the template manually, or interactively, using a specific query.

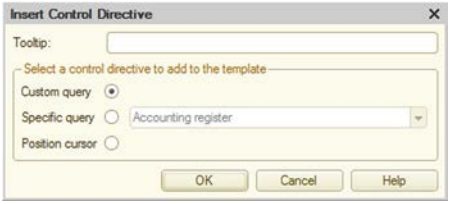


Fig. 481. Control directive insertion

You need to position the cursor in the template area with a variable part and click <->Insert... The Insert Control Directive dialog box is displayed.
In the Tooltip attribute, user specifies a text line displayed as an explanation to an action performed when a context-dependent part of template is requested. Since a template can contain several control directives that can be entered in a sequence, this text explains the performed actions and the location of the variable part.
Radio button group can be used to select the type of a control directive for insertion into the template.
A custom query is used to enter any text. A control directive of <?"Tooltip text"> type will appear in the template text.
The Custom query control directive is used as follows. If the text to be replaced contains a <?"Tooltip text"> directive, a request with tooltip text and text input field is displayed before the replacement text is inserted. In the input field, you can specify the text to be inserted in the replacement text instead of <?"Tooltip text"> directive. There could be several Custom query type directives in the replacement text and some directives can have the same tooltip text. When inserting a replacement text, a specific text request is provided for each directive. There will be one common query for all identical directives and the text entered in the query will replace all identical directives.
Specific query is used to select configuration objects. To place a type selection of a configuration object into the control directive template using the Insert control directive window, you need to enter the query name in the Tooltip field and

select a directive from the list (selection of configuration object, predefined configuration object item or an appropriate directive), for example, Catalog.

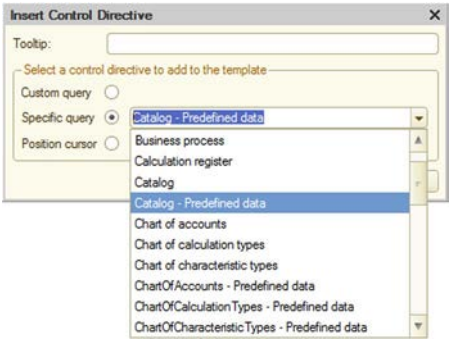


Fig. 482. Specific Query

When using this template, a list of possible configuration objects is displayed. The name of a selected object is placed instead of the entered text. Position of the selected string is stored and the list will be positioned on this string when selecting this template again.

The possible values of a specific query include the switch, we will additionally describe it. When a template with this type of a query is formed and OK is clicked, the window of possible values formation list is displayed.

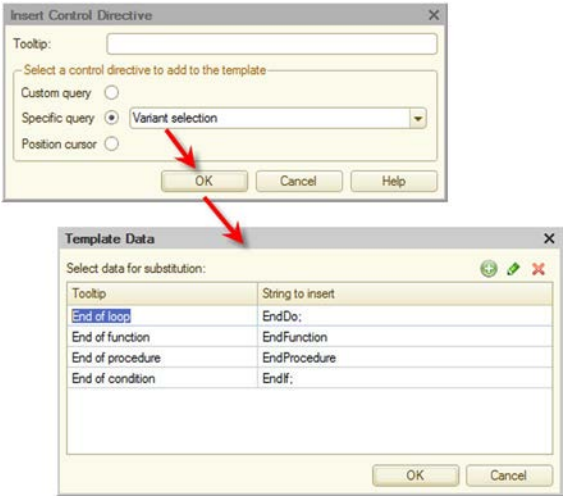


Fig. 483. Data for Selection

The Tooltip column is used to enter text, that will be displayed in the selection list when this template is used. The Insertion string column is used to enter text, that will be placed into the text document.
The list can only be edited in the template text. When re-calling the insert directive, a new directive will be added.

After pressing OK, a control directive is inserted into the template text.

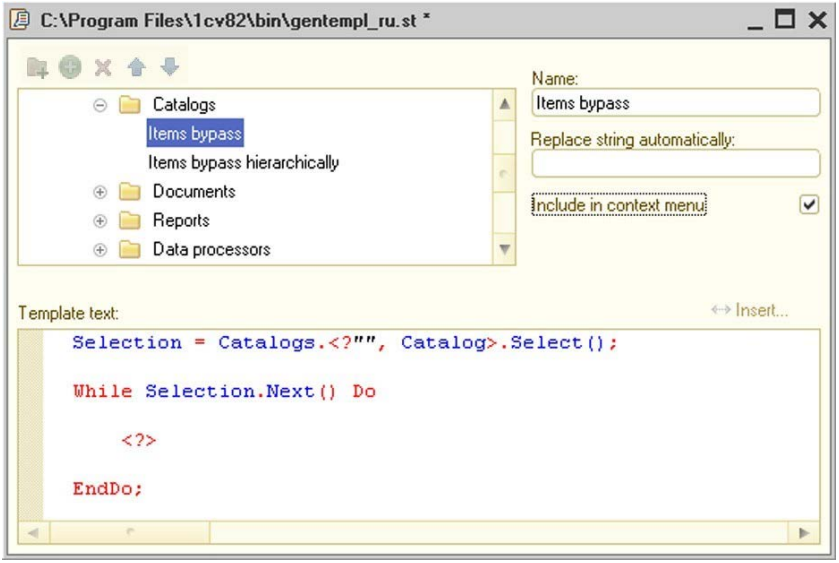


Fig. 484. Template with a control directive

In the template tree (or during automatic substitution) select this template.

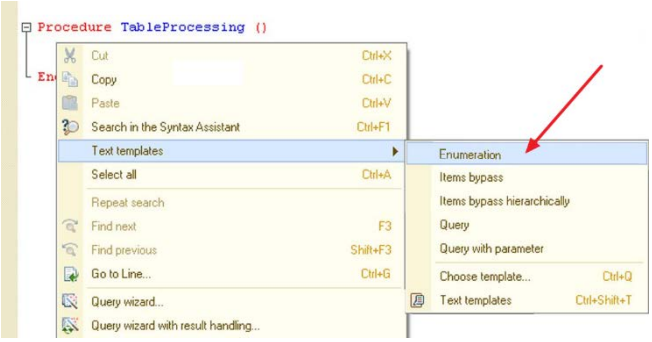


Fig. 485. Context menu

In this case, a catalog type selection query is displayed.

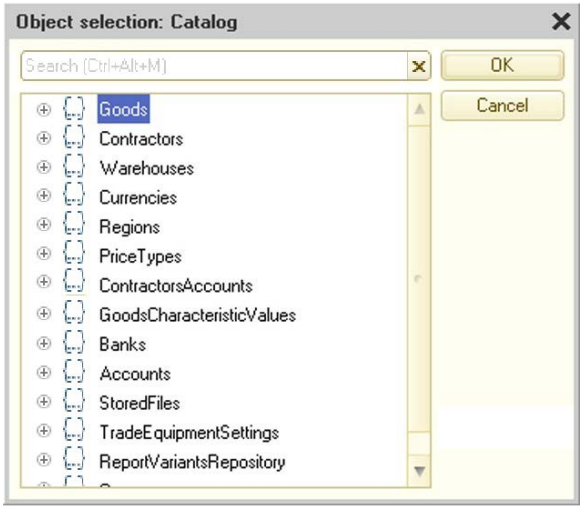


Fig. 486. Enumeration selection

As a result of a selection, the following text is generated:

```
PredefinedValue("Enumeration.ProductTypes.Product")
```

Position cursor - is used to position the cursor in the indicated location after inserting template text. The <?> type directive is inserted in the replacement template text. This construct defines where the cursor should be positioned when entering the replacement text. If there are several directives of this type in the replacement text, the cursor is positioned in the first in order <?> directive place.
When writing a template text, the use of multiple control directives as well as the use of existing templates is allowed.

37.6. Syntax Assistant

37.6.1. General information

Syntax Assistant- a tool that facilitates module development. The main task of the Syntax Assistant - provide swift tooltips on 1C:Enterprise language to the specialist configuring 1C:Enterprise.
To call the Syntax Assistant, use the Help -Syntax Assistant item.

The Syntax Assistant window is divided into two parts. Window division can be performed vertically or horizontally.
The selected 1C:Enterprise section details are displayed in the bottom (or right) part. The contents of the upper (or left) part are defined by the selected tab.
The window contains three tabs: Content, Index and Search. A hierarchical list of 1C:Enterprise language items is placed on the first tab: operators, control directives, procedures and functions, system constants, etc. And the second and third tabs are used for searches using the 1C:Enterprise language item names and for custom searching using description text correspondingly.
If horizontal arrangement is selected for the Syntax Assistant, tabs on the command bar are replaced with buttons that have the same names.

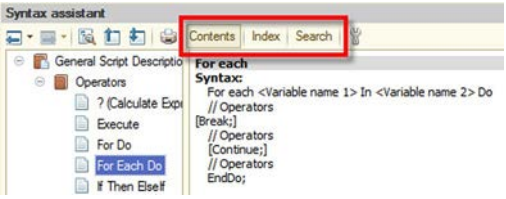


Fig. 487. Horizontal arrangement

The filter of displaying the contents tree and Syntax Assistant index can be set up by object availability for a specific 1C:Enterprise start mode.

37.6.2. Syntax Assistant settings

Command bar buttons of the Syntax Assistant are described on fig. 488.

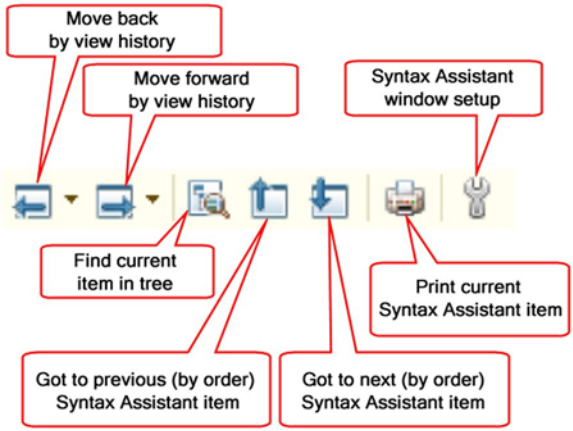


Fig. 488. Description of command bar buttons

37.6.3. "Content" Tab of Syntax Assistant

For convenience, all 1C:Enterprise language items are grouped into thematic sections, presented in the form of tree branches, that is displayed in the upper part of the Syntax Assistant window.

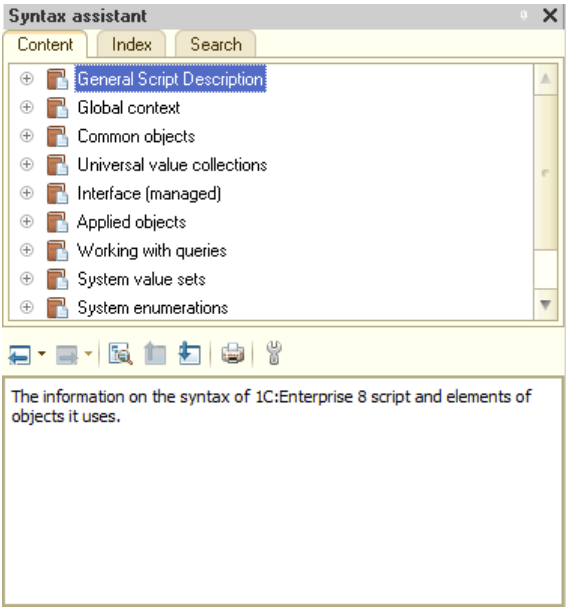


Fig. 489. Syntax Assistant Window

Operating methods in the Syntax Assistant window are standard for hierarchical data that is presented as a tree. To quickly expand a branch, use the "+", "-" and "*" keys of the numeric keypad: The "+" key expands one level of the branch, the "-" -collapses a level and the "*" - expands all sub-branches of the current branch.

To get description, select the name of a 1C:Enterprise language item in the upper part of the Syntax Assistant window. Item description will be displayed in the lower part of the window. Description can contain links to descriptions of the referenced 1C:Enterprise language items.

The window contains a toolbar that can be used to view the history of previously selected descriptions.

To display the location of an item, description of which is currently displayed in the lower part of the window, in 1C:Enterprise language items structure, click the Find current element in tree button in the toolbar.

37.6.4. Copying 1C:Enterprise language Items

The Syntax Assistant has a function that allows copying of the selected 1C:Enterprise language item to the text editor.
To make a copy, select the name of the desired language item in the tree-type list and use the clipboard. 1C:Enterprise language item instance will be moved to the cursor position in the text editor window.
Beside using the clipboard for copying, you can also drag and drop instances of 1C:Enterprise items: you can just drag the selected item from the Syntax Assistant window to the text editor window.

37.6.5. Searching in Syntax Assistant

Searching in Syntax Assistant can be performed by 1C:Enterprise language item name and by arbitrary description text.
A search by 1C:Enterprise language item name is performed in the Index tab of the Syntax Assistant window. The upper part contains a field to input the required 1C:Enterprise item name and a full list of all item names.
During the name input, the program performs a context search. When searching, with each typed character a transition is made to the first name on the list that begins with the entered character.

To call the context search mode, select the Search item in Syntax Assistant from the context menu of text editor. This will display the Syntax Assistant window, that is opened on the Index tab.

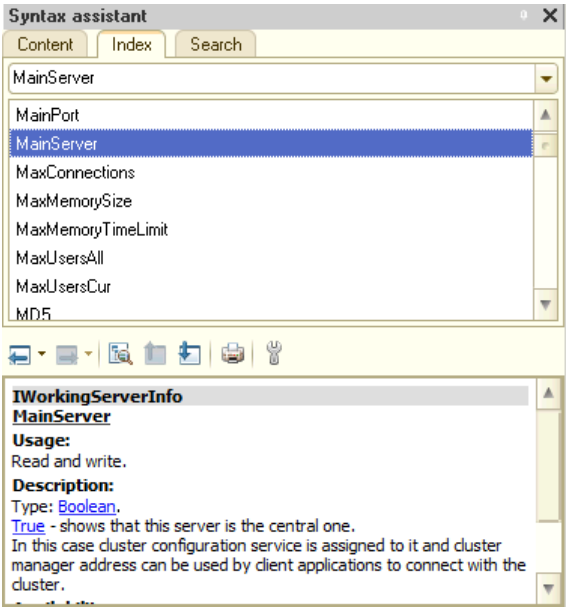


Fig. 490. Syntax Assistant index

If the found (specified) item is used in several objects, the Select section window, that contains a list of entries for the selected item, will open.

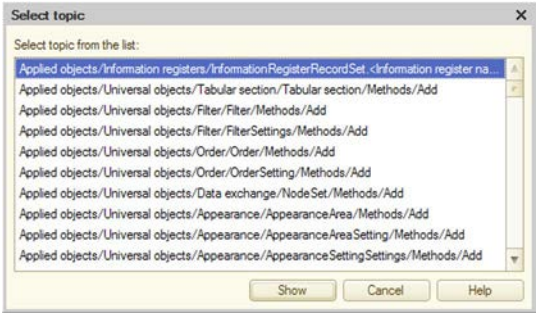


Fig. 491. Select section

Now, if you press the Show button, the description of the found 1C:Enterprise item, selected in the window of entries will be displayed in the Syntax Assistant window.

TIP. If the Select section window contains a lot of lines (see figfig. 491), you can press the Ctrl+F key combination and enter the name of the searched object in the search dialog box. This will speed up the search for the required section.

You can search for 1C:Enterprise language item description in the Syntax Assistant while editing a module. To perform a search, position the cursor over the 1C:Enterprise language item for which you want to get a description or select a text block and press Ctrl + F1 key combination. The result of the search with the help of Ctrl+F1 key combination does not depend on Display deprecated objects check box status. It is assumed that the help info is obtained conditionally with the help of the method which is already in the configuration source text.
If the word, on which the cursor is positioned or the selected text block is a 1C:Enterprise language item described in the Syntax Assistant, the Syntax Assistant window containing the description of this item will be immediately displayed.
If a 1C:Enterprise language item is described in several thematic sections of the Syntax Assistant, the Select section window will open first to select the necessary thematic section.
If the selected word or text block is not a 1C:Enterprise language item, the context search window will open in the Syntax Assistant when you press Ctrl + F1. key combination.
The performed search is stored in a list that can be called in the name input field. To view a section, select it from the list and press the Enter key. Description of the selected section will be shown in the lower field.
The Search tab of the Syntax Assistant window is used to search using an arbitrary description text. The upper part contains a search string input field and a field with a list of founded 1C:Enterprise language items descriptions.
To begin the search start typing text. While typing, the program searches for sections where the entered text is found. You can use any input register, text words are considered wholly (unless the "*" operator is used), with morphology also considered. Use of search operators is allowed.

While typing, the program swiftly displays a list of these sections.

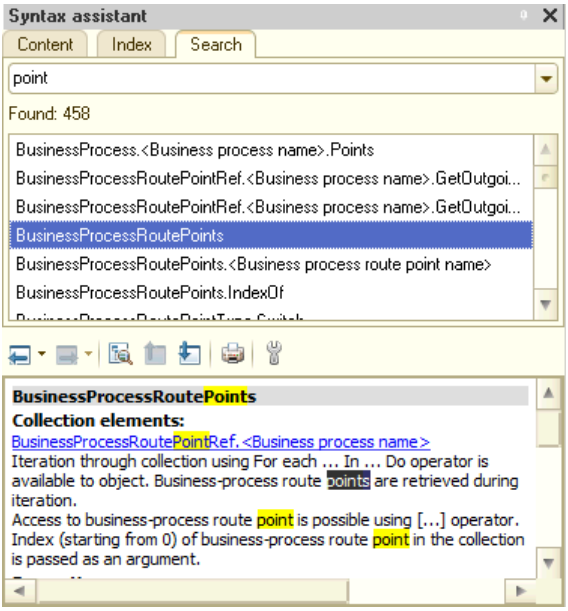


Fig. 492. Searching in Syntax Assistant

When you open a topic, the help system highlights the first occurrence of the search string in that topic.

To view a section, select it from the list and press Enter. Description of the selected section will be shown in the lower field.

When the section containing description of the 1C:Enterprise language item is found and the Syntax Assistant window is open, use the Find current element in the tree button of the command bar to search for the section in the description tree.

If you selected several pages while viewing, you can use the Go forward and Go back commands to return to previously viewed pages.

37.6.6. Using methodological materials

When viewing a Syntax Assistant article, there is a feature to go to various methodological materials available on the 1C company web resources. For this purpose, there is a Methodical materials hyperlink at the bottom part of the Syntax Assistant article.

Clicking this link will open a web browser, which will display an appropriate page with links to additional materials on the studied Syntax Assistant article. To navigate through the links an authorization on the relevant site might be required. In Windows OS, the link will be opened in Microsoft Internet Explorer and in Linux OS, the web browser used will depend on the system settings.

37.6.7. Versions details

The information on the versions associated with the displayed object is displayed in the bottom section of the Syntax Assist window which displays the information on the property, function, procedure, etc. ("object" term can be used).

NOTE. The information on the recommended replacements is displayed for the Syntax Assist objects which were outdated in version 8.3.14 and subsequent versions. No such information is provided for the objects announced outdated in the preceding versions.

The following set of information is displayed:

1. The platform version as of which release the object was implemented. In this case, if the object implementation version precedes version 8.3.1 then only the first two numbers are displayed as the version details, for example, Accessible from version 8.1. If an object was implemented in version 8.3.1 (and subsequent versions) then more complete details are displayed, for example, Accessible from version 8.3.6. If an object was implemented taking into account the compatibility mode this information will be displayed in Syntax Assist as well: Accessible from version 8.3.6 (in the mode of compatibility with version 8.3.6 and subsequent versions).
2. The platform version as of which release the object was announced outdated. If there are recommended replacements for the outdated object then

the link to the recommended objects will be displayed in the Syntax Assist window as well.

3. The platform version as of which release the object description was considerably amended. The term "considerable amendment" means any change in the information describing the object functioning. For example, if a list of accessibility is changed for the object -it is a considerable amendment and if any orthographic mistakes -are corrected it is an inconsiderable amendment. The information on such change will be represented by the text Description is edited in version 8.3.13.

The configuration compatibility current mode determines which platform objects can and -can't be used in this configuration. Syntax Assistant simplifies the decision-making: can this object be used in this compatibility mode or not. If the compatibility mode prevents the object use then the reason is displayed in red in the top section of the object details, before the syntax description. There can be two reasons:

1. Object is not accessible **yet**: it means that the compatibility mode is set in the value for which the desired platform object if not yet implemented.
2. Object is not accessible **already**. It means that the used object is announced outdated and its use is not recommended. Despite the fact that the deprecated objects remain functional to keep the compatibility (as a rule), the use of replacements is recommended.

37.7. Comparing files

You can compare any two files. To do this, select File - Compare files item. This opens the file selection dialog box.

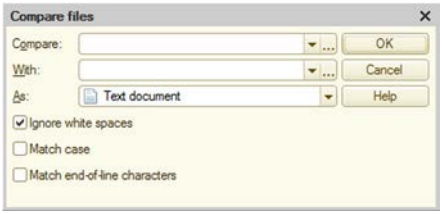


Fig. 493. Comparing files

In the Compare and With fields, specify the names of the files to be compared. File names can be entered manually, selected from the drop-down list or by clicking the (...) selection button, call the standard file opening dialog box and select the file in this dialog box. You can specify any type of source file in the File Type field of the dialog box.

In the As field, specify the file comparison method. You can compare text and spreadsheet documents, external data processor. You can also use the binary comparison, where the files are compared on a binary level. In this case, the comparison result will be a message about the match or difference of files.
If files of different types are specified in the Compare and To fields, the selected files will be interpreted by comparison to the type specified in the As field. Then a comparison is performed and comparison results are displayed.
Click OK to start comparison.
If you selected the Text document comparison type, the selected documents will be interpreted for comparison as text documents, despite the fact that non-text files can be selected (by extension) in the Compare and With fields. If one or both of the specified files cannot be interpreted as a text file, a binary comparison is conducted.

37.7.1. Comparison of external data processors

When comparing external data processors, a standard procedure of external data processor comparison is conducted.
Selecting the Match by object names check box, will allow matching objects firstly by names and then by internal IDs. If the check box is cleared, matching is done by internal IDs.

37.8. Built-in help application

Help content can be used to get information on the 1C:Enterprise operations and on specific objects and modes. You can call it at any time by selecting Help item or by pressing the F1 key. This will open the help information section that corresponds to the mode in which the user is currently operating.
The Contents menu item and the Shift + F1 key combination allows you to access the overall contents of help information where you can get information on operations in Designer mode.

Help content contains a description of configuration sections, 1C:Enterprise language and additional help information.
The toolbar used for navigation and information searching is located in the upper part of the help content window. The help content is displayed below the toolbar.

Chapter 38. Add-ins

38.1. General information

1C:Enterprise is an extensible application. It's functionality can be extended by using add-ins.
The application uses two technologies of add-ins creation:

- using Native API,
- using COM technology.

This allows to create add-ins for:

1. Windows OS and Linux OS.
2. Web client running in Microsoft Internet Explorer, Mozilla Firefox, Google Chrome and Safari web browsers.
3. Mobile application, working on iOS, Android and Windows OS.

Add-ins designed for operations in web client or mobile application environment, must be packed into a ZIP archive with an appropriate structure. This file can also be used on the 1C:Enterprise server and in other clients.
When operating with add-ins that have been created using Native API, the following procedure should be followed:

- Interface implements a command, that installs the add-in on user's computer (global context method `InstallAddIn()`). This operation is required in thin and web clients.
- The add-in is attached using the `AttachAddIn()` method. You should also keep in mind, that an attempt to attach a non-installed add-in causes an error.
- An attached add-in is used in compliance with the provided program interface.

TIP. Don't combine codes for add-in installation and attachment. Installation is considered as a one-time event and re-installation causes an interactive exclusion.

When an add-in that is packed in a ZIP archive is used, to specify the add-ins parameters (name of the file to be installed, type, architecture and web browser to be used) a specific manifest file is used, generated by the add-in developer.

Add-ins obtained from a configuration or infobase are saved when received and are used upon subsequent connection without re-acquisition (add-in used on the server is only saved for the server working process lifetime).

NOTE. Attachment of external add-ins that are designed using the COM technology using the `LoadAddIn()`, `AttachAddIn ()` methods and component object registration are performed "for user". If "for user" registration fails, an attempt to perform "for computer" registration will be made.

If an add-in is attached using the `AttachAddIn()` method , the parameter generation in the `New()` operator (of the created type name) is performed from the following parts:

- Text `Addin.`;
- Name specified as the `Name` parameter of the `AttachAddIn()` method ;
- Name of the object implemented in the add-in.

If a `Timer` object is implemented in the add-in and the add-in is attached as follows: `AttachAddIn(URL, "MyName");` then, to create an object from the add-in, you need to specify the following ID: `New("Addin.MyName.Timer")` .

Detailed information on the development of add-ins is available on the 1C: ITS Portal (http://its.1c.ru/db/metod8dev#content:3221:hdoc).

38.2. Features of operation in thin and thick clients

When operating in thick and thin clients, you can use add-ins developed with COM technologies as well as add-ins developed with Native API technologies (as separated files stored on hard drive or appropriate ZIP archives).

Add-ins can be updated by add-ins re-installation using the `InstallAddIn()` method.

Add-ins are installed in the %APPDATA%\1C\1Cv8\ExtCompT directory. Add-in installation directory is not treated as cache and is not cleared when 1C:Enterprise is called using the ClearCache command line key. Using the `InstallAddIn()` method is required in thin client.

38.3. Operation features in web client

When operating in a web client, you can use add-ins developed with Native API or COM technologies and add-ins packed in appropriate ZIP archives types. Add-ins developed with COM technologies can be used only if the web client operates in Windows OS. When installing the add-ins file, user permission for extension installation is asked.

Each add-in is installed as a separate web browser extension. This extension is a distribution package designed for a particular kind of browser. Add-ins are installed:

- for Microsoft Internet Explorer web browser - in the %ALLUSERSPROFILE%\Application Data\1C\1CEWebExt (%ALLUSERSPROFILE%\1C\1CEWebExt directory for Windows Vista OS and higher);
- for Mozilla Firefox web browser - in current user profile directory (created by web browser);
- for Google Chrome web browser - in current user profile directory (created by web browser);
- for Safari web browser - defined by add-in developer.

Add-ins can be removed:

- for Microsoft Internet Explorer web browser - via add/remove programs feature of the control panel;
- for Mozilla Firefox web browser - via web browser extensions feature;
- for Google Chrome web browser - via web browser extensions feature (navigate to chrome://plugins link);
- for Safari web browser - as defined by add-in developer.

IMPORTANT! Attachment of add-ins from files stored in the hard drive is not supported in web client for safety reasons.

Also you should keep in mind, that operations with COM objects are available only in Microsoft Internet Explorer web browser and Windows OS. Also, the `COMSafeArray` type is not supported as well as the feature to call methods of COM objects, that operate with this kind of values.

To update add-ins, perform one of the following operations:

- remove add-in as described above and re-install it using `InstallAddIn()` method;

- edit the name of an add-in object in its source text and manifest file (the name of the file with the add-in may not change) and re-install it using the `InstallAddIn()` method.

When using add-ins in configuration, you should follow certain rules and recommendations. Following these rules is required to successfully install add-ins on all supported web browsers.

- Do not use these types of statements:

```
If Not AttachAddIn() then
  InstallAddIn()
EndIf;
```

- Installation of an add-in or a platform extensions (file system extension and cryptography extension) should be interactive. The application user decides on installation. In the installation dialog box it must be specified for what purposes the add-in is required and what will not operate, if it will not be installed.
- It is recommended to integrate the add-in installation into the execution process of the application action. Example:
  1. User can use the Send report command.
  2. To do so, the application requires an add-in to be installed.
  3. Application checks, whether the add-in is installed or not.
  4. If an add-in is not installed, a dialog box is displayed to the user, containing information about the need to install the add-in to send the report and two buttons: to install the add-in and to continue sending the report.
  5. Then the user sequentially performs two interactive actions:
     1. Installing the add-in;
     2. Continue sending report.
- The application must have implemented tools for the user to install add-ins and platform extensions at any time. Actions related to add-ins and platform extensions installation do not have to be performed during the execution of any action.

38.4. Features of operations on server

When operating on the 1C:Enterprise server, only add-ins developed with Native API technologies, that can be separate files as well as appropriate ZIP archives, are used.

To attach an add-in use the `AttachAddIn()` method (without using the `InstallAddIn()` method that is unavailable on server).

When operating on server, it is advisable to attach the add-in right before using it. Moreover, add-ins used on the server must support all architectures and platforms. This is because server code can be generally performed in various operating processes (in different time points) that can be used on various computers. Computers can have different architectures as well as different OS.
Add-in file is saved by working process until restarted, that's why the add-in reattachment (before the restart of working process or switching to another working process) takes less time than the first attachment.

38.5. Features of operations in mobile application

In a mobile application only add-ins developed with Native API technologies that can be separate files as well as appropriate ZIP archives are used.

38.6. Asynchronous operation with an add-in

Using add-ins in asynchronous mode has several features:

1. Add-in attachment is made using the asynchronous method `BeginAttachingAddIn()`.
2. To refer to add-in properties, use the asynchronous methods `StartInstall<PropertyName>()` and `StartAcquiring<PropertyName>()`.
3. Instead of referring to add-in method, use asynchronous methods `StartCall<MethodName>()`.

`Start… ()` methods are automatically added by the platform to the existing properties and methods of the add-in. There is no need to separately process the add-in itself to support operations in asynchronous mode, but it is necessary to recompile the existing add-in to support asynchronous operations.
For the `StartCall...()` method, the first parameter to be specified should be the notification description and the remaining parameters match the parameter set in the original method of the add-in.
When using Google Chrome and Mozilla Firefox web browsers, only asynchronous attachment and use of add-ins is supported, while other web browsers can use both synchronous and asynchronous ways of operation. This is determined by the configuration property Using synchronous calls of platform extensions and add-ins mode.

38.7. Examples of add-in use

38.7.1. Native API technology

Native API add-ins can be attached in both client applications and 1C:Enterprise application server.
When the add-in is operating on the server, `AttachAddIn()` call needs to be executed every time before creating an add-in instance, i.e. in general, it can not be predicted at what server the call will be made (it can be Windows OS, Linux OS, 32-bit or 64-bit).

38.7.1.1. Add-in installation on client

```
InstallAddIn("CommonTemplate.BarcodeScannerDriver");
```

38.7.1.2. Load from file on hard drive

Add-in file must be located on user's computer and available in the search path (`PATH` operating system environment variable).

```
SystemInfo = NewSystemInformation;

If SysInfo.PlatformType = PlatformType.Windows_x86 Then
  AttachAddIn("AddInCPP.dll", "MyName", AddInType.Native);
ElseIF SysInfo.PlatformType = PlatformType.Windows_x86_64 Then
  AttachAddIn("AddInCPP64.dll", "MyName", AddInType.Native);
ElseIf SysInfo.PlatformType = PlatformType.Linux_x86 Then
  AttachAddIn("libAddInCPP.so", "MyName", AddInType.Native);
Else
  AttachAddIn("libAddInCPP64.so", "MyName", AddInType.Native);
EndIf;

AddInObject = New("AddIn.MyName.ComponentExtension");
```

NOTE. Please note the parameter generation in the `New` command.

38.7.1.3. Load add-in from template

```
SystemInfo = NewSystemInformation;

If SysInfo.PlatformType = PlatformType.Windows_x86 Then
  AttachAddIn("DataProcessor.AddIn.Template.AddInWindows32", "MyName",
AddInType.Native);
ElseIF SysInfo.PlatformType = PlatformType.Windows_x86_64 Then
  AttachAddIn("DataProcessor. AddIn.Template.AddInWindows64",
"MyName", AddInType.Native);
ElseIf SysInfo.PlatformType = PlatformType.Linux_x86 Then
  AttachAddIn("DataProcessor.AddIn.Template.AddInLinux32", "MyName",
AddInType.Native);
Else
  AttachAddIn("DataProcessor. AddIn.Template.AddInLinux64", "MyName",
AddInType.Native);
EndIf;

AddInObject = New("AddIn.MyName.ComponentExtension");
```

NOTE. Please note the parameter generation in the `New` command.

38.7.1.4. Load from infobase

Infobase must contain the `AddIns` catalog and attributes for specified names of `ValueStorage` type with add-in files.

```
Variable link;

SystemInfo = NewSystemInformation;

If SysInfo.PlatformType = PlatformType.Windows_x86 Then
  Ref = GetURL(Catalogs.AddIns.FindByCode("000000001"),
"AddInWindows32");
ElseIF SysInfo.PlatformType = PlatformType.Windows_x86_64 Then
  Ref = GetURL(Catalogs.AddIns.FindByCode("000000001"),
"AddInWindows64");
ElseIf SysInfo.PlatformType = PlatformType.Linux_x86 Then
  Ref = GetURL(Catalogs.AddIns.FindByCode("000000001"),
"AddInLinux32");
Else
  Ref = GetURL(Catalogs.AddIns.FindByCode("000000001"),
"AddInLinux64");
EndIf;

AttachAddIn(Ref,"MyName", AddInType.Native);
AddInObject = New("AddIn.MyName.ComponentExtension");
```

NOTE. Please note the parameter generation in the `New` command.

In these examples of operating with Native API add-ins there also add-ins developed with COM technologies only for Windows OS can be attached.

38.7.1.5. Attach Add-in from a ZIP archive

The configuration must contain `BarcodeScannerDriver` common template of `BinaryData` type. This template stores a ZIP archive of appropriate structure containing add-ins for all supported operating systems, browsers and processor architectures.

```
AttachAddIn("CommonTemplate.BarcodeScannerDriver", "Scanner");
AddIn = New("AddIn.Scanner.BarcodeScanner");
```

NOTE. Please note the parameter generation in the `New` command.

38.7.2. COM technology

38.7.2.1. Load add-in

```
LoadAddIn("MyComponent.dll");
AddIn = New("AddIn.ComponentExtension");
```

38.7.2.2. Attach add-in

```
AttachAddIn("AddIn.MyObject");
AddIn = New("AddIn.ComponentExtension");
```

NOTE. `LoadAddIn()` command is used for compatibility with previous 1C:Enterprise versions.

Chapter 39. Specifics of cross-platform application development

39.1. General information

This topic contains recommendations for developing applications running on Windows OS as well as Linux OS.

39.2. File mode features

When operating with files in Linux OS, note that file and directory names are case-sensitive. For example, files with file.txt and File.txt names are different files that can be located in the same directory.
There is no concept of hard drives in Linux OS that is widely used in Windows OS. Linux OS has a root file system ("/"); other file systems located on other media (hard drives and articles, flash drives, ISO files etc.) can be mounted at any place in the root file system. The most frequently used (but not mandatory!) locations of external media mounting are /mnt or /media mounting points (directories). When mounting external media, they can be mounted as /media/<section label> or /media/<Device ID>. Automatic external media mounting is not guaranteed.
There is no letter assignment to hard drives in Linux OS. This can cause the following problems:

- Application uses explicitly defined paths oriented for using Windows OS only;
- Application uses path check mechanisms that verify hard drive letter assignment as well;
- When specifying a path starting with root directory, note that in Linux OS this path definitely determines the location of a directory or file but in Windows OS - it does not. For example, \temp\file.txt (or /temp/file.txt in Linux OS) path definitely specifies file location and name in Linux OS: file.txt file located in file.txt directory of root file application. For Windows OS, it is not that simple. This path means that file.txt file is located in temp directory of the current hard drive root

directory. The current hard drive can be different for two startups of the same client application with the same infobase.

When developing applications, note the following moment: Windows OS and Linux OS use different path separators. Windows OS uses "\" ("back slash"), and Linux OS uses - "/" ("forward slash"). When operating with 1C:Enterprise language methods, there is no need to consider this difference as they operate correctly with any separator. However, if it is required to pass the path to an external application, you should specify the path separators used in OS that 1C:Enterprise operates with. To receive path separators information, use `GetPathSeparator()`, `GetClientPathSeparator()`, `GetServerPathSeparator()` functions.

In addition to different path separators, file masks are also processed differently in Windows OS and Linux OS. For example, the `*.*` mask is used to indicate all files in Windows OS and all files containing "." - in their name in Linux OS. This difference can cause unexpected behavior of the`FindFiles()`, `DeleteFiles()` 1C:Enterprise language methods and the `FileDialog` object. Features of specifying file masks in different operating systems are described in Syntax Assistant for `FindFiles()`,`DeleteFiles()` functions. To receive a file mask describing all files, use the `GetAllFilesMask()`, `GetClientAllFilesMask()`, `GetServerAllFilesMask()` functions.

Another feature of Linux OS is that there is no need to call network resources using the UNC notation (in this case, the address looks as follows: \\server\folder\folder\file.ext). Therefore, if you need to use any external network resource, you must first mount it to the root file system. You should keep in mind, that every mounting can be performed to a different mounting point (different directory) can be used. Also, access to external network resources can be performed using appropriate applications (for example, smbclient), but note that these applications can be not installed in the used Linux OS copy.

Detailed information about Linux file application and operations with it is available in operating system documentation or internet resources.

39.3. Operating with external devices

External devices are named (and accessed) differently in different operating systems. For example, sequential ports in Windows OS are named COM1, COM2 etc. In Linux OS (depending on distribution package and the device) ports can be named: /dev/ttyS0, /dev/ttyS1 etc. or: /dev/ttyUSB0, /dev/ttyUSB1 etc. Note this feature when developing add-ins operating with the device as well as developing add-ins setting dialog box. For example, a list of ports that can be specified in data processor should be directly received from add-in that is set up using an appropriate method.

39.4. Rights limitations

In Linux OS, operating on behalf of a super-user account (root) is not customary. For a regular user, only the home directory and, as a rule, the temporary files directory are always available for recording. The remaining part of the file system can be used only either in read-only mode or is unavailable (the entire application or just certain directories). In this regard, when developing, you should keep in mind what access the user that is running "1C:Enterprise" will have to directories that return methods of the global context:

- DocumentsDir() - read and write access;
- BinDir() - read access;
- TempFilesDir() - read and write access;
- UserDataWorkDir() - read and write access.

To access external devices, user can be required to join a specialized group (or groups). For more information, see the used OS documentation.

39.5. Fonts

When developing applications it is recommended to use style fonts without explicitly specified typeface. If required, use fonts included in Microsoft Core Fonts (http://sourceforge.net/projects/corefonts/). Do not use other fonts in applications, as they can be missing in some used OS, and the application appearance would be distorted.

39.6. HTMLDocumentField

To use HTML in 1C:Enterprise (all operating systems), WebKit is supported. When you use HTML, note the following:

- An HTML document can be displayed differently in different OS.
- When accessing DOM model of an HTML document using the `HTMLDocumentField.Document` property, you should use only the properties and methods available in all web browsers supported by 1C:Enterprise. This is due to using various tools to display HTML document in client application as well as due to the fact that web access can be used to operate with application, increasing the number of used web browsers and their versions.
- If you use HTML document field to load an external web site, note that the said web site can restrict embodiment of `iframe` using `X-Frame-Options: DENY` response header. As such, this web site is not displayed in the web client in HTML document field.

39.7. COM, OLE and ActiveDocument mechanisms

These mechanisms are unavailable in Linux and macOS OS. Use alternative tools for integration, such as file exchange in XML format or Web services. For add-ins implemented using with COM technologies it is recommended to refactor them using NativeAPI.

Chapter 40. Other mechanisms

40.1. Solving systems of linear equations

40.1.1. General information

System of linear equations (SLE abbreviation can be used) is a system of equations in which each equation is a first-degree algebraic equation (linear equation). Variables coefficients, free terms, and unknown variables are deemed to be real numbers. In particular, systems of linear equations are used in various economic tasks such as the manufacturing range self-cost calculation.

NOTE. A description of the systems of linear equations solving methods used and their specific features is beyond the scope of this documentation. The systems of linear equations solving methods description shall be looked for in the relative educational materials.

Systems of linear equations solving mechanism used in 1C:Enterprise uses two algorithms: iteration and direct solution. A number of optimizations developed by 1C Company are used to chose a certain mechanism.
Let's consider the application of an implemented mechanism as exemplified by a simple system of linear equations:

```
(1)   X1 +   X2 =  5
(2) 2*X1 + 3*X2 = 13
```

The numbers in brackets determined the equation number.
`SolveLinearEquationsSystem` object will be used to solve the system. Two tables are used as the input data:

1. The table containing the values of the free coefficients of the system of linear equations.
2. The table containing the values of variables coefficients of the system of linear equations.

These tables can be obtained both "manually", by filing the lists of values in, and on the basis of a query to a database. In the current example, the data will be prepared in the lists of values. The equations are numbered from top to bottom and the variables in the system are numbered -from left to right.

Let's draw a list of free coefficients:

```
NodesData = New ListOfValues;
NodesData.Colums.Add('EquationNumber");
NodesData.Colums.Add("FreeCoefficient");
String = NodesData.Add();
String.EquationNumber = 1;
String.FreeCoefficient = 5;
String = NodesData.Add();
String.EquationNumber = 2;
String.FreeCoefficient = 13;
```

Let's draw a list of variables coefficients:

```
LinksData = New ListOfValues;
LinksData.Columns.Add("EquationNumber");
LinksData.Columns.Add("VariableNumber");
LinksData.Columns.Add("Coefficient");
String = LinksData.Add();
String.EquationNumber = 1;
String.VariableNumber = 1;
String.Coefficient = 1;
String = LinksData.Add();
String.EquationNumber = 1;
String.VariableNumber = 2;
String.Coefficient = 1;
String = LinksData.Add();
String.EquationNumber = 2;
String.VariableNumber = 1;
String.Coefficient = 2;
String = LinksData.Add();
String.EquationNumber = 2;
String.VariableNumber = 2;
String.Coefficient = 3;
```

Now, the drawn lists of coefficients shall be transferred to the calculation mechanism. To do it, fill in the properties `NodesDataSources` (list of free coefficients) and `LinksDataSources` (list of coefficients):

```
Solver.NodesDataSource = NodesData;
Solver.LinksDataSource = LinksData;
```

Now, the columns holding the numbers of equations and coefficients in the transferred sources shall be specified. Only the column holding the numbers of equations (with the help of the property `EquationColumnInNodes`) must be specified for the list of free coefficients but both where the numbers of equations (`EuationColumnInLinks` property) and the numbers of variables (`VariablesColumnInLinks`) are located must be specified for the list of variables coefficients:

```
Solver.EquationColumnInLinks = "EquationNumber";
Solver.VariablesColumnInLinks = "VariableNumber";
Solver,EquationColumnInNodes = "EqautionNumber";
```

The solved system of equations description creation is the last step. To do this, use `SolveLinearEquationSystem.SystemsDescriptions` property. The column responsible for the system free coefficient and the column responsible -for the solved system variables coefficients must be specified for each added system:

```
LinearEquationsSystemDescription = Solver.SystemDescription.Add();
LinearEquationsSystemDescription.CoefficientColumnInNodes =
"FreeCoefficient";
LinearEquationsSystemDescription.CoefficientColumnInLinks =
"Coefficient";
```

Now all parameters are set and the given system of equations can be solved:

```
Result = Solver.SolveLinearEquationsSystems();
```

Out system of linear equations solving will result in a list of values consisting of two columns:

- `NodeNumber` column contains the system of linear equations variable number.
- `Solution1` column contains the solution value for each variable.

For the used example the solution will be as follows: $X_1 = 2$, $X_2 = 3$.

Example resulting text:

```
Solver = New SolveLinearEquationsSystesm;
NodesData = New ListOfValues;
NodesData.Colums.Add('EquationNumber");
NodesData.Colums.Add("FreeCoefficient");
String = NodesData.Add();
String.EquationNumber = 1;
String.FreeCoefficient = 5;
String = NodesData.Add();
String.EquationNumber = 2;
String.FreeCoefficient = 13;
LinksData = New ListOfValues;
LinksData.Columns.Add("EquationNumber");
LinksData.Columns.Add("VariableNumber");
LinksData.Columns.Add("Coefficient");
String = LinksData.Add();
String.EquationNumber = 1;
String.VariableNumber = 1;
String.Coefficient = 1;
String = LinksData.Add();
String.EquationNumber = 1;
String.VariableNumber = 2;
String.Coefficient = 1;
String = LinksData.Add();
String.EquationNumber = 2;
String.VariableNumber = 1;
String.Coefficient = 2;
String = LinksData.Add();
String.EquationNumber = 2;
String.VariableNumber = 2;
String.Coefficient = 3;
Solver.NodesDataSource = NodesData;
Solver.LinksDataSource = LinksData;
LinearEquationsSystemDescription = Solver.SystemDescription.Add();
LinearEquationsSystemDescription.CoefficientColumnInNodes =
"FreeCoefficient";
LinearEquationsSystemDescription.CoefficientColumnInLinks =
"Coefficient";
Solver.EquationColumnInLinks = "EquationNumber";
Solver.VariablesColumnInLinks = "VariableNumber";
Solver,EquationColumnInNodes = "EqautionNumber";
Result = Solver.SolveLinearEquationsSystems();
```

Let's consider how to solve several systems of linear equations simultaneously. To simultaneously solve several systems of linear equations, perform the following preliminary action: -bring the systems to the same number of variables and number of equations. In the simplest case, it is done by forming the lacking equations or insertion of variables with zero coefficient into the equation. Thus, before solving several systems, we need the uniform numbering of both equations and their variables.

After that, the correct formation of input data for the system of linear equation solving mechanism is necessary. As has been pointed out earlier, the list of free coefficients (`Solve.LinearEquationsSystems.NodesDataSource` property) and the list of system variables coefficients (`SolveLinearEquationsSystems.LinksDataSource` property) are input in the system solution. A new column is added for each additional system in each date source. The names of the columns responsible for the coefficients on one system in each source are specified with the help of the property `SolveLinearEquationsSystems.SystemDescription`.
Let's consider an example of two systems of linear equations:

```
(1) 2*X₁ + 3*X₂ = 10              4*X₁ + 15*X₂ = 40
(2) 3*X₁ + 2*X₂ = 15              5*X₁ +  3*X₂ = 10
```

There are two systems of two equations each of which contains two variables. The following list shall be drawn to specify the free coefficients:

EquationNumber	CK1	CK2
1	10	40
2	15	10

In the example above, the column of the list of values for the free coefficients is called `FreeCoefficient`. But there are more than one systems now and we can't specify two columns in the list of values with the same name. Therefore, the columns will be called `CK1` and `CK2`.
The list of systems of linear equations variables coefficients will look analogously:

EquationNumber	VariableNumber	K1	K2
1	1	2	4
1	2	3	15
2	1	3	5
2	2	2	3

Now, instead of the name `Coefficient`, the columns will be called by "numbers" of the systems of linear equations: `K1` (for the first system of linear equations) and `K2` (for the second system of linear equations).
The system specification of the names of columns containing the numbers of equations and variables is left unchanged:

```
Solver.EquationColumnInLinks = "EquationNumber";
Solver.VariablesColumnInLinks = "VariableNumber";
Solver,EquationColumnInNodes = "EqautionNumber";
```

But the system of linear equations description will somewhat change. Now, the description of two systems will be instead of the description of one system:

```
LinearEquationsSystemDescription = Solver.SystemDescription.Add();
LinearEquationsSystemDescription
LinearEquationsSystemDescription.CoefficientsColumnInLinks = "K1";
LinearEquationsSystemDescription = Solver.SystemDescription.Add();
LinearEquationsSystemDescription.CoefficientsColumnInNodes = "CK2";
LinearEquationsSystemDescription.CoefficientsColumnInLinks = "K2";
```

Formally, each system of linear equations is defined by the names of columns of free coefficients and variables coefficients in the lists input in the systems of linear equations solving mechanism. Therefore, solving several systems of linear equations which differ only in the free coefficients it is not necessary to repeat the variables coefficients for each system. In this case, it will be enough to create the columns only for the unique parameters - free coefficients. Based on the system of linear equations formal description (from the point of view of solving mechanism), two more systems of linear equations can be set in the example above by combining the variables coefficients and free coefficients from different systems. It is evident that such combination is meaningful only if it useful.
Two system of linear equations solving will result in a list of values consisting of two columns:

- `NodeNumber` column contains the system of linear equations variable number.
- `Solution1` column contains the solution value for each variable from the first system.
- `Solution2` column contains the solution value for each variable from the second system.

It is also worth pointing out when several systems of linear equations are recommended to be solved by one call and when by -different calls. If several systems of linear equations have the same dimension and evident dependence by the coefficients they are recommended to be solved by one call. For example, if three systems with forecast variants are solved: pessimistic, realistic, and optimistic. If the systems of linear equations have no evident relation (just two different systems) they are recommended to be solved separately. Such difference is explained by the fact that the platform analyzes the transferred systems and presupposes their similarity. Some internal optimizations are performed on the basis of the carried-out analysis. Wrong decisions -can be made if the systems are not related.

Let's additionally consider some parameters of the object `SolveLinearEquatiosnsSystems`:

- `AdequateAccuracy` property -is the maximum solution error. It is calculated as the difference between the left and right part of each equation of the solved system by inserting the current solution. Excessing accuracy setting will make the solving process much longer. To get in the result the accurate to the ith decimal place, it is recommended to set the accuracy in form of 10 raised to `-(i+1)` power. Thus, to get the accurate 5th decimal place, set `AdequateAccuracy` property in the value 10^{-6}.
  When changing the required accuracy, it is recommended to change `AdequateAccuracy` property first. If no result is achieved it may be caused by an insufficient number of iterations used for solving. In this case, `IterationCount` property value can be increased.
- `IterationCount` property determines the number of iterations to be used by the iteration algorithm to solve a system of linear equations. This setting is secondary as compared to `AdequateAccuracy` property. It is recommended to choose the minimal possible value to obtain the solution required accuracy in the result (`AdequateAccuracy` property). It is pointless to increase the value above the required value because the algorithm will stop calculating after reaching adequate accuracy.
- `ComputationalResourcesUse` property allows setting how many computing threads will participate in the system of linear equations solving. The default value provides the maximal use of accessible computing power. This value shall not exceed the number of accessible processor cores (including the cores accessible in the result of using hyper-threading technology). It is recommended to change this property only if it is required to decrease the number of computing threads of the mechanism as compared to the default value.
  If all processor resources can't be allocated for the systems of linear equations solving but only one computing thread is clearly not enough then it is recommended to set the number equal to the number of simultaneously solved systems (in this case each system will be computed independently) or divisible by it (in this case the number of threads equal to the multiplicity factor will be solving each system) as `ComputationalResourcesUse` property value. In this case, the resulting value shall not exceed the number of processor cores.

As has been stated at the beginning of the section, system of linear equations solving mechanism uses two algorithms: iteration and direct. Each of these algorithms has its advantages and disadvantages:

- Iteration algorithm:

    - The operation speed depends on the required accuracy of the solution. It is simultaneously an advantage and disadvantage of this algorithm. It is an advantage because it allows limiting the required accuracy instead of solving with excessive accuracy (for this subject field). It is a disadvantage because the unjustified increase in the accuracy can either fatally affect the solving speed or disallow any solution at all.
    - Algorithm advantages:
        - Works faster in large matrixes (asymptotics $O(n^2)$).
    - Algorithm disadvantages:
        - Higher accuracy may require a considerable number of iterations (the computation will take longer as the result).
        - There are theoretical limitations on the algorithm convergence).
- Direct algorithm:
    - Algorithm advantages:
        - Guaranteed accurate solution (limited by the accuracy of mathematical calculations of the computer processor).
        - Works faster on the small matrixes.
    - Algorithm disadvantages:
        - Works considerably slower on large matrixes (asymptotics $O(n^3)$).

`SolvingLinearEquationsSystems.AlgorithmChangeLimit` property determines what algorithm will be preferred after the preliminary analysis of the systems of linear equations transferred to be solved by one call. The smaller the value -the higher the probability of iteration algorithm use is and the higher the value -the higher the probability of the direct algorithm use is. It shall be understood that shifting the priority to any algorithm, not only the advantages but also disadvantages of the prioritized algorithm use are escalated. The default value provides the rational balance of mechanism operation and it shall be changed only in the case of a very good understanding of the general scheme of operation and the consequences of the decision made.

40.1.2. Graph cohesion components

If the task to solve can be modeled with the help of a graph then the necessity to decompose the resulting graph may arise. Faster solving is a decomposition purpose. It can be done by operating the notion "graph cohesion components". The platform allows distinguishing the following cohesion components (the notion is presented by the enumeration `ConnectedComponentsTypeOfSystemOfLinearEquetionsCalculationGetMethod`):

- `WeaklyConnectedComponents`. Weakly connected components are maximally included graph components (subgraphs), such that there is a path from any component (subgraph) vertex to another vertex of this component (subgraph). Graph edge orientation is ignored (oriented edges are considered as non-oriented).
- `StronglyConnectedComponents`. Strongly connected components are maximally included graph components (subgraphs), such that there is a path from any component (subgraph) vertex to another vertex of this component (subgraph). The path is built based on edge orientation.
- `StronglyConnectedComponentsWithoutLinkRequirementWithinComponents`. This term means the cohesion components guaranteeing that simultaneously there is no path from A to B and from B to A for any two tops A and B. At the same time, no guarantees of the presence of paths from one component for any two tops A and B. In this variant, the system calculates the connected components that are the most optimal to analyze and debug mechanisms for solving systems of linear equations.

Let's consider the examples of distinguishing the components with different cohesion. There is a system of equations:

```
(0)   X0 + 5*X1 = 1
(1) 6*X0 + 2*X1 + X2 = 1
(2) 3*X2 = 1
(3) 4*X3 = 1
```

This system can be represented the following graph:

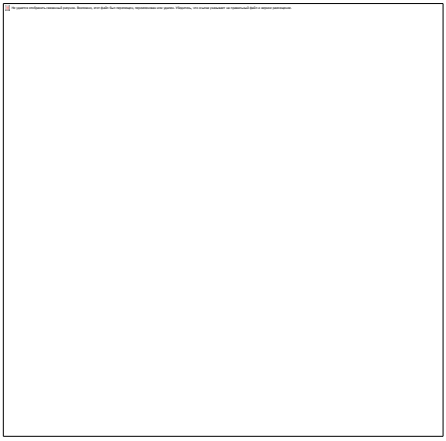


Fig. 494. Initial graph

If we obtain the low cohesion components for this graph two components will be distinguished:

- Component 1: {0, 1, 2}.
- Component 2: {3}.

Or graphically:

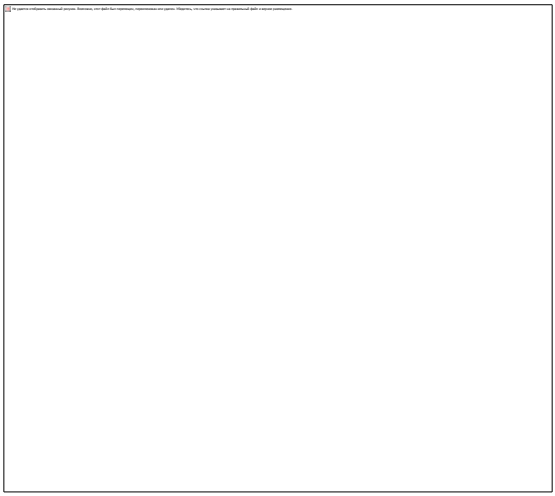


Fig. 495. Low cohesion components

If we obtain the high cohesion components for this graph thee components will be distinguished:

- Component 1: {0,1}.
- Component 2: {3}.
- Component 3: {2}.

Or graphically:

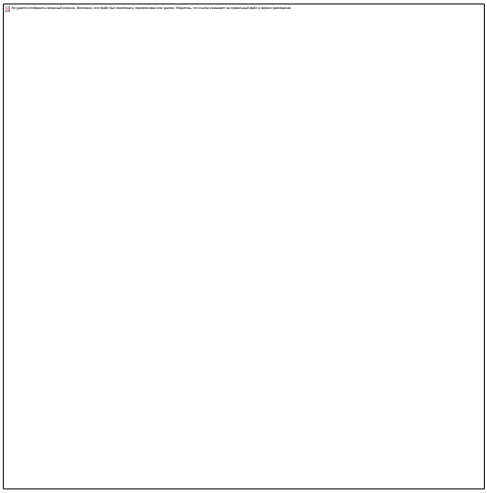


Fig. 496. High cohesion components

The distinguishing of components of weak and high cohesion does not depend on the order of equations and variables coefficients setting. The distinguishing of components of high cohesion without any requirement of the link inside the component depends on the order of input of equations and variables definitions to the system of linear equations solving mechanism. Thus, if the equations and coefficients are set in the natural order (top to bottom, left to right) then three components (analogously to the distinguishing of components of high cohesion) will be distinguished:

- Component 1: {0, 1}.
- Component 2: {2}.
- Component 3: {3}.

It is graphically shown on fig. 496.

But if to bring the equation (2) parameters formation to the beginning then the distinguishing of components of high cohesion without any requirement to the link inside the component will result in two components (analogously with the distinguishing of components of low cohesion):

- Component 1: {0, 1, 2}.
- Component 2: {3}.

It is graphically shown on fig. 495.

In 1C:Enterprise language, enumerated above will look as follows:

```
Solver = New SolveLinearEquationsSystesm;
NodesData = New ListOfValues;
NodesData.Colums.Add('EquationNumber");
NodesData.Colums.Add("FreeCoefficient");
String = NodesData.Add();
String.EquationNumber = 0;
String.FreeCoefficient = 1;
String = NodesData.Add();
String.EquationNumber = 1;
String.FreeCoefficient = 1;
String = NodesData.Add();
String.EquationNumber = 2;
String.FreeCoefficient = 1;
String = NodesData.Add();
String.EquationNumber = 3;
String.FreeCoefficient = 1;

LinksData = New ListOfValues;
LinksData.Columns.Add("EquationNumber");
LinksData.Columns.Add("VariableNumber");
LinksData.Columns.Add("Coefficient");
String = LinksData.Add();
String.EquationNumber = 0;
String.VariableNumber = 0;
String.Coefficient = 1;
String = LinksData.Add();
String.EquationNumber = 0;
String.VariableNumber = 1;
String.CoefficientNumber = 5;
String = LinksData.Add();
String.EquationNumber = 1;
String.VariableNumber = 0;
String.Coefficient = 6;
String = LinksData.Add();
String.EquationNumber = 1;
String.VariableNumber = 1;
String.Coefficient = 2;
String = LinksData.Add();
String.EquationNumber = 1;
String.VariableNumber = 2;
String.Coefficient = 1;
String = LinksData.Add();
String.EquationNumber = 2;
String.VariableNumber = 2;
String.Coefficient = 3;
String = LinksData.Add();
String.EquationNumber = 3;
String.VariableNumber = 3;
String.Coefficient = 4;

Solver.LinksDataSource = LinksData;
Solver.EquationColumnInLinks = "EquationNumber";
Solver.VariablesColumnInLinks = "VariableNumber";
```

```
LinearEquationsSystemDescription = Solver.SystemDescription.Add();
LinearEquationsSystemDescription.CoefficientColumnInLinks =
"Coefficient";
LowCohesion =
Solver.GetCohesionComponents(ConnectedComponentsTypeOfSystemOfLinearEq
uationsCalculationGetMethod.WeakCohesionComponents);
HighCohesion =
Solver.GetCohesionComponents(ConnectedComponentsTypeOfSystemOfLinearEq
uationsCalculationGetMethod.HighCohesionComponents);
HighCohesionWithoutRequirement -
Solver.GetCohesionComponents(ConnectedComponentsTypeOfSystemOfLinearEq
uationsCalculationGetMethod.HighCohesionComponentsWithotRequirementToL
inkInsideComponent);
```

The following values will be in the variables in the result of this example operation:

- `LowCohesion` -list of values containing the low cohesion components.
- `HighCohesion` -list of values containing the high cohesion components.
- `HighCohesionWithoutRequirement` -list of values containing the high cohesion components without any requirement to the link inside the component. This variant will distinguish the components analogously to the high cohesion components.

If the example is modified, namely:

- strings:

```
String = NodesData.Add();
String.EquationNumber = 2;
String.FreeCoefficient = 1;
```

shift right after the string:

```
NodesData.Colums.Add("FreeCoefficient");
```

- strings:

```
String = LinksData.Add();
String.EquationNumber = 2;
String.VariableNumber = 2;
String.Coefficient = 3;
```

shift right after the string:

```
LinksData.Columns.Add("Coefficient");
```

then, after the example execution, `HighCohesionWithoutRequirement` table will contain the components analogous to the components of low cohesion.

40.1.3. Cohesion components search mechanism use recommendations

The following scenarios of cohesion components search mechanism use are recommended.

Low cohesion components search use

The simplest algorithm dividing the system of linear equations into absolutely unconnected parts. Each of these parts can be:

- Changed without any influence on the solution of the system of linear equations other parts.
- Solved independently of the system of linear equations other parts.
- Subjected to any additional processing without any influence on the solution of the system of linear equations other parts.

It is not recommended to separately and independently solve any parts of the system of linear equations if they are small (for example, it is more efficient to solve 100 part of the system of 10 equations together than separately).
The search for the low cohesion components is also recommended when the iteration solution of the system of linear equations is necessary: solve any part of the system of linear equations, subject the source system to additional processing, and solve the changed system. In this case, the following method is recommended:

- Separate low cohesion components.
- Find the components containing the changed part of the system of linear equations (these components will be called "changed"). The remaining components will be called "unchanged". Then follow these steps:
  - Solve the changed and unchanged parts independently.
  - Change the components as required.
  - Solve the changed components again.
  - Unite the solutions of the unchanged components and the resulting solution of the changed components to obtain the general solution of the source system of linear equations.

High cohesion components search use

This mechanism use is recommended when the use of the low cohesion components search mechanism has yielded no expected result: too few components of low cohesion were separated (or just one component was separated).
High cohesion components - "denser" subgraphs of the source graph.

It is recommended to build the high cohesion components, to manually or automatically analyze the cohesion between the selected high cohesion components in the source graph, and to try to remove them using one of the methods:

- Source data editing.
- Change of the model according to which the system of linear equations was formed.
- Temporary deletion of these edges according to the following scenario:
  - Temporary deletion of the edges connecting the high cohesion components.
  - Low cohesion components calculation. If all connecting edges are deleted between two high cohesion components then in the subsequent building of a low cohesion component these very components will form the low cohesion components. It means that when all paths between the high cohesion components are deleted -then subsequently the high and low cohesion components will look the same.
  - The necessary calculation in accordance with one of the scenarios specified in Low Cohesion Components Search Use is made.
  - The temporarily deleted edges are returned to the source graph
  - Additional processing on the system of linear equations solution relaxation is made. The relaxation is necessary because the temporarily deleted edges may (and most probably will) change the system of linear equations final solution. For a more efficient and simpler relaxation, it is also recommended to test the stability of the solutions of the system of linear equations parts separated with the help of the low cohesion components search.

Use of the search for the components from the system of linear equations solution

This mechanism is first of all intended for a more efficient tuning and deeper understanding of the systems of linear equations solving mechanism work. It is recommended in the cases of a manual analysis of the systems of linear equations. The components building this mechanism (for any component top there is no path to any other top of any other component) depend on the order of numeration of tops but allow the iteration analysis of the system of equations starting from the components numbered 0 and moving farther along the components because the component numbered 0 is guaranteed to have no cohesion with the subsequent components oriented by edge.

This method is recommended for the following situations:

1. The system of linear equations is very large but the user wants to understand the main connections -what tops render the greatest influence on

other tops. It requires the iteration movement through the cohesion components analyzing what tops are included in the equations with the largest coefficient. The solution stability analysis can be performed on the results of this analysis: inconsiderably changing the coefficients of the key tops, see how considerably the system of linear equations final solution changes.

2. The calculation result manual check: the analysis of the situation in which the incomplete filling in of the system of linear equations solutions values or the absence of a top from the solution (for example, the specification of not all coefficients of the system of linear equations or incorrect specification of its coefficients leads to such result).

Appendix 1. Conversion of "1C:Enterprise 7.7 " infobases

Installation of both 1C:Enterprise and 1C:Enterprise 7.7 (release no earlier than 7.70.021) on the computer is required.
To start the converter, you need to start the V7Cnv.exe program.
If there is more than one 1C:Enterprise 7.7 release installed on the computer, then you should be positive that the release not earlier than 7.70.021 was started last; otherwise, when the converter starts, a message about the discrepancy of the release will be displayed and the program will be terminated.
The converter window is displayed.

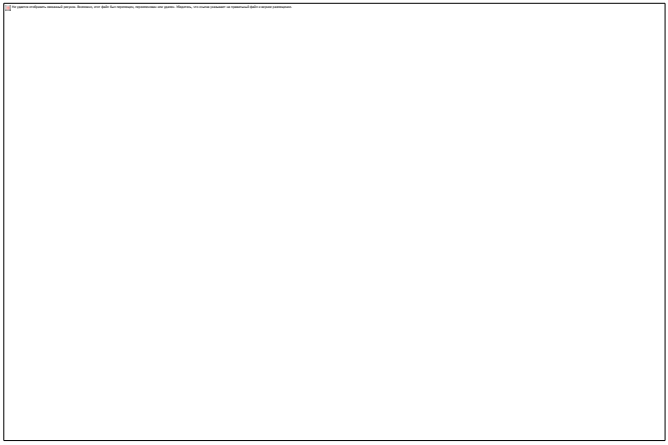


Fig. 497. Converter window

Converter set-up is done in steps. On the first step, you need to specify the parameters of the infobase that you need to convert, as well as the parameters of the resulting infobase. If you need to convert an infobase into the database of client/server format, then in the Information base directory string you should specify the connection string with the base on the server. Basic configurations and configurations with a set password are not converted.

On the second step, the conversion components are configured.

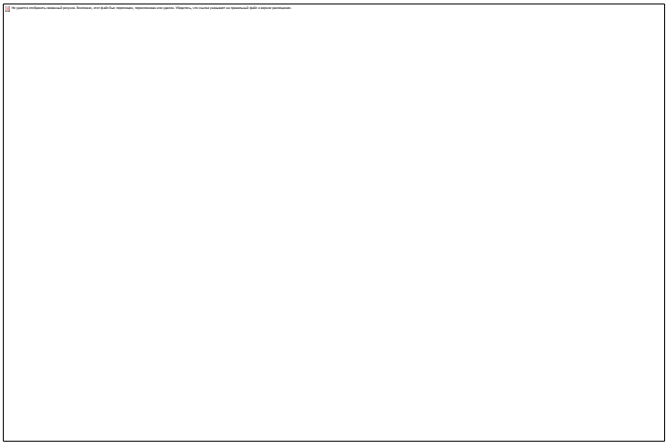


Fig. 498. Conversion component settings

Select the check boxes for the actions to be performed.
On the third step, to start the analysis of the source infobase, press the Execute button.

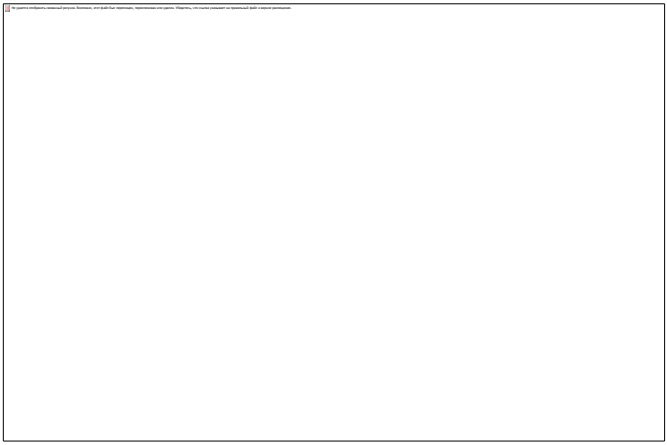


Fig. 499. Before the infobase analysis

The source infobase is opened in batch mode, and the data is downloaded. After downloading, the base will close.

The converter displays information about periodic attributes of configuration objects and constants to select the conversion option.

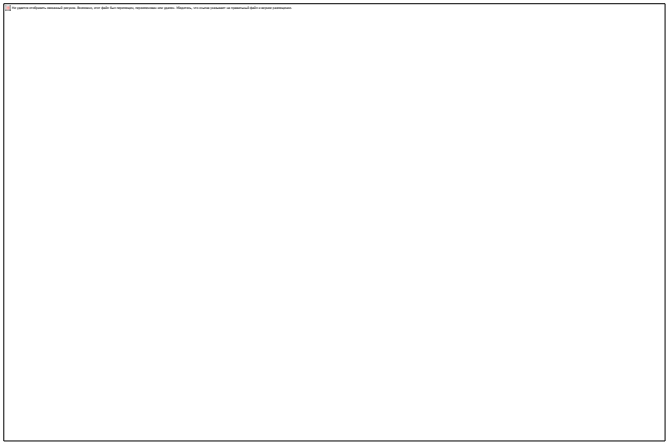


Fig. 500. Conversion settings

In the Option column, the user can choose how to convert the attribute (constant): in attribute directory (constant) or information register.
On the next step, to start the conversion, press the Execute button.

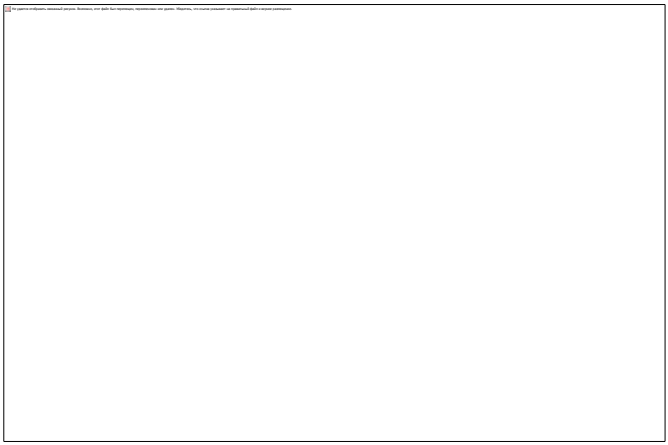


Fig. 501. Before performing the conversion

Converter transfers metadata structure, forms, spreadsheet documents, interfaces, sets of rights, pictures, as well as database data and user list.

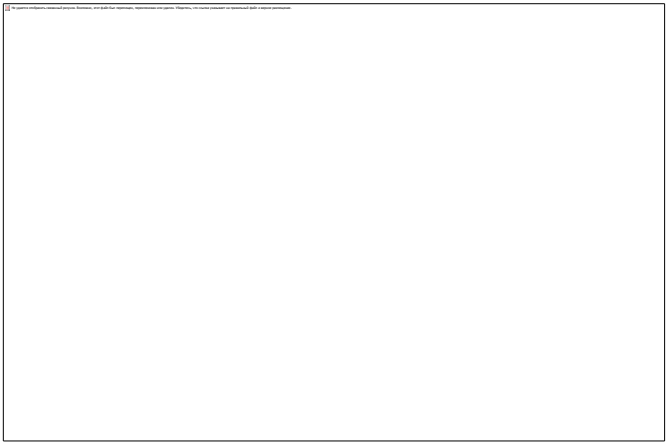


Fig. 502. Conversion process

When creating documents in the process of configuration conversion, the use of document in accounting types flags are considered. If the document has a use in operational accounting flag, when converting the document, all accumulation registers created during the conversion are added as movements; when used in accounting -all accounting registers; when used in calculation- all calculation registers.

The conversion is performed with the following restrictions:

- Declarations of procedures, functions, and variables are transfered. All executable code is commented, with the exception of the program code created by the converter itself.
- In the process of conversion, document journals graphs are converted into filter criteria. However, if, for example, the attribute of a document has the `Directories` type, then, although it will be included in the filter criterion components, which has the `Directories.Employees` type, the next time you save the configuration, you will receive a message about the wrong type of this attribute. Although, formally the `Directories.Employees` type is included in the `Directories` type, the filter criterion does not allow including attributes of such “composite” type. You should either correct the type of the attribute by specifying certain directories to the objects, the attribute of which can store references, or to exclude the document attribute from the filter criteria components.

- User passwords are not transfered.
- The converter does not fill the `Movement` property for converted documents, that determines the registers components, where the document stores registers. After conversion, this property should be set manually for all documents.

After the conversion is completed, a completion message is displayed in the window.

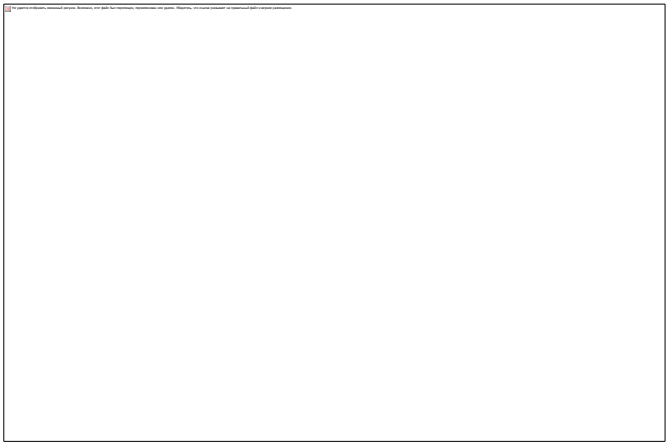


Fig. 503. Conversion completed

Appendix 2. URL formats

This Appendix describes the link formats used in 1C:Enterprise. Links can be external or internal. External links are used outside the 1C:Enterprise, while internal links are used internally (including adding to favorites list and operation history).

2.1. General description of the link format

A link consists of two parts:

- Infobase host address - describes infobase location relative to client application. This part of the link can be received using `GetInfoBaseURL()` method.
- Internal link - describes the location of requested information within the infobase.

There are several types of **Internal links**:

- infobase object,
- infobase object attribute,
- infobase object tabular section attribute,
- infobase register record attribute,
- infobase register record,
- report
- data processor
- section,
- section command,
- temporary storage;
- collaboration system conversation.

External link is created by adding an internal link to the host address. If a link opens a form or performs navigation in the main window, the following rule is used to create an external link:

```
<Infobase host address>#<internal link>
```

These links are references to an infobase object, report, data processor, section and section commands links.

If a link is a resource, an external link is generated as follows:

```
<Infobase host address>/<internal link>
```

These links are references to attributes of infobase object or information register record or temporary storage links.

TIP. If multiple clients are operating with the infobase, it is not recommended to save external links to infobase objects because this may lead to the inoperability of the link created in one client in another client. For example, an external link created in the web client can be unavailable in thin client "directly" attached to 1C:Enterprise server. It is recommended to save internal links.

2.2. Host address format

Depending on the infobase type, the string describing host address can look differently.

Infobase file mode

If a UNC format is used to define the database file path, host address looks as follows:

```
e1c:/file/<UNC-path>
```

Example:

```
// Infobase file path
\\dbsrvr\bases\mybase

// Host address
e1c:/file/dbsrvr/bases/mybase
```

If the file path includes a hard drive letter, host address looks as follows:

```
e1c:/filev/<Hard drive letter>/<Base path>
```

Example:

```
// Infobase file path
s:\bases\mybase

// Host address
e1c:/filev/s/bases/mybase
```

Client/server mode of infobase

For client/server mode of the infobase, link looks as follows:

```
e1c://server/<server name>/<infobase name>
```

Example:

```
// Infobase connection string
srvsr="srv1C";ref="mybase"

// Host address
e1c://server/srv1C/mybase
```

When operating via web server (thin client or web client)

If a web server is used to operate with the infobase, link looks as follows:

```
http://<host address>/<infobase path>
https://<host address>/<infobase path>
```

Example:

```
// Infobase connection string
http://localhost/mybase

// Host address
http://localhost/mybase
```

2.3. Internal links

2.3.1. General information

Common internal link format looks as follows:

```
<affiliation>/<type>/<path>[?<parameter>=<parameter
value>[&<parameter>=<parameter value>]]
```

Internal link affiliation describes where the object identified by this link is located:

- `e1cib` - information is stored in the infobase.
- `e1ccs` - information is stored on the collaboration system server.

A detailed description of type-dependent link is given below.

2.3.2. Infobase objects links

2.3.2.1. Infobase object link

Format:

```
e1cib/data/<путь к метаданным>?ref=<идентификатор ссылки>
```

Where:

- metadata path - type of the linked object, for example, `Document.GoodsConsumption`;
- link ID - infobase object UUID.

2.3.2.2. Infobase object attribute link

Format:

```
e1cib/data/<путь к метаданным>.<имя реквизита>?ref=<идентификатор
ссылки>
```

Where:

- metadata path - type of the linked object, for example, `Document.GoodsConsumption`;
- attribute name - infobase object attribute name;
- link ID - infobase object UUID.

2.3.2.3. Infobase object tabular section attribute link

Format:

```
e1cib/data/<путь к метаданным>.<имя табличной части>.<имя
реквизита>?ref=<идентификатор ссылки>&index=<индекс строки
табличной части>
```

Where:

- metadata path - type of the linked object, for example, `Document.GoodsConsumption`;
- tabular section name - metadata object tabular section name, for example, `Goods`;
- attribute name - infobase object tabular section attribute name;
- link ID - infobase object UUID;
- tabular section line index - tabular section line index.

2.3.2.4. Infobase register record link

Format:

```
e1cib/data/<metadata path>?<key field name>=<value>[&<key field
name>=<value>]
```

Where:

- metadata path - type of linked object, for example, `InformationRegister.ProductBalance`;

- key field name - register dimension name (`Period` dimension is added to periodic registers);
- value - internal presentation of filter value. Filter value is coded according to section 2.2. URL Character Encoding Issues of RFC 1738 standard (http://tools.ietf.org/html/rfc1738) - using the «%» symbol and two hexadecimal characters.

2.3.2.5. Infobase register record attribute link

Format:

```
e1cib/data/<metadata path>.<attribute name>?<key field
name>=<value>[&<key field name>=<value>]
```

Where:

- metadata path - type of linked object, for example, `InformationRegister.ProductBalance`;
- attribute name - record attribute name;
- key field name - register dimension name (`Period` dimension is added to periodic registers);
- value - internal presentation of filter value. Filter value is coded according to section 2.2. URL Character Encoding Issues of RFC 1738 standard (http://tools.ietf.org/html/rfc1738) - using the «%» symbol and two hexadecimal characters.

2.3.2.6. Report link

Format:

```
e1cib/app/<metadata path>
```

Where:

- metadata path - type of linked object, for example, `Report.ProductBalance`.

2.3.2.7. Data processor link

Format:

```
e1cib/app/<metadata path>
```

Where:

- metadata path - type of linked object, for example `DataProcessing.OrderGeneration`.

2.3.2.8. Link to section

Format:

```
e1cib/navigationpoint/<section name>
```

Where:

- section name - the linked first-level subsystem name, for example `GoodsSales`.

2.3.2.9. Link to navigation point created by standard command

Format:

```
e1cib/navigationpoint/<section name>/<command name>
```

Where:

- section name - the linked first-level subsystem name, for example, `GoodsSales`;
- command name - command name, for example, `Catalog.Goods.OpenList`.

2.3.2.10. Temporary storage link

Format:

```
e1cib/tempstorage/<temporary value ID>
```

Where:

- temporary value ID - temporary value ID.

2.3.2.11. Link to external data source table record

Format:

```
e1cib/data/<metadata path>?<key field name>=<value>[&<key field
name>=<value>]
```

Where:

- metadata path - describes external data source table, for example, `ExternalDataSource.Receipts.Table.CheckHeaders`.
- key field name - external data source table key field name.
- value - internal presentation of filter value. Filter value is coded according to section 2.2. URL Character Encoding Issues of RFC 1738

standard (http://tools.ietf.org/html/rfc1738) - using the «%» symbol and two hexadecimal characters.

2.3.2.12. Links to commands of the global command interface

Format:

```
e1cib/command/<command name>
```

Where:

- command name - command name:
  - standard command - `Catalog.Goods.Create`;
  - common command - `CommonCommand.ObjectSearch`;
  - command - `Catalog.Goods.Command.PrintBarcode`.

2.3.2.13. Lists links

Format:

```
e1cib/command/<list name>
```

Where:

- list name - the name of the list to be opened, for example `Catalog.Goods`, `Documents.GoodsConsumption`.

2.3.2.14. Home page links

Format:

```
e1cib/navigationpoint/desktop
```

In version 8.3.2 or later compatibility mode.

Format:

```
e1cib/navigationpoint/startpage
```

In versions earlier than 8.3.2 compatibility mode.

2.3.3. Collaboration system objects links

2.3.3.1. Conversation link

Format:

```
e1ccs/data/conv?id=<conversation ID>
```

Where:

- conversation ID - collaboration system conversation UUID.

2.3.3.2. Link to the message.

Format:

```
e1ccs/data/msg?id=<message ID>
```

Where:

- message ID - collaboration system message UUID.

The obtainment of links to the service and sent messages is not supported.

Appendix 3.

Rules of generating standard command texts and automatic form headers

To predefine (or specify) objects presentation in an interface, a set of properties is implemented in metadata properties.

Object	**Extended presentation**	**Object presentation (for register - record)**	**Extended object presentation (for register - record)**	**List presentation**	**Extended list presentaion**	**Description**	**Picture**
Subsystems						+	+
Constants	+					+	
Exchange plans		+	+	+	+	+	
Filter criteria				+	+	+	
Common forms	+					+	
Constants	+					+	
Catalogs		+	+	+	+	+	
Documents		+	+	+	+	+	
Documents journals				+	+	+	
Enumerations				+	+	+	
Reports	+					+	
Data processors	+					+	
Charts of characteristic types		+	+	+	+	+	

Object	Extended presentation	Object presentation (for register - record)	Extended object presentation (for register - record)	List presentation	Extended list presentaion	Description	Picture
Charts of accounts		+	+	+	+	+	
Charts of calculation types		+	+	+	+	+	
Information registers (subordinate)				+	+	+	
Information registers (independent)		+	+	+	+	+	
Accumulation registers				+	+	+	
Accounting registers				+	+	+	
Calculation Registers				+	+	+	
Business processes		+	+	+	+	+	
Tasks		+	+	+	+	+	

When generating the standard commands presentation (and their tooltips), application uses the following rules (Property1 → Property2 means that if Property1 is not specified, Property2 is used instead):

- Opening list:
  - command presentation: List presentation;
  - command tooltip: Explanation → Extended list presentation → List presentation.
- Creating new object (record):
  - command presentation: Object presentation: create or Record presentation: create;
  - command tooltip: Extended object presentation(Extended record presentation) → Object presentation(Record presentation).
- Create new group:
  - command presentation: List presentation: create group;
  - command tooltip: Extended list presentation → List presentation.
- Navigation to any subordinate list (command of Go to group in form navigation panel):

    - command presentation: List presentation;
    - command tooltip: Explanation → Extended list presentation → List presentation.
- Generate:
    - command presentation: Object presentation: generate;
    - command tooltip: Extended object presentation → Object presentation.
- Opening a report or data processor:
    - command presentation: report or data processor presentation;
    - command tooltip: Explanation → Extended presentation.
- Opening a common form:
    - command presentation: common form presentation;
    - command tooltip: Explanation → Extended presentation.
- Opening the constant editing form:
    - command presentation: constant presentation;
    - command tooltip: Explanation → Extended presentation.
- Navigation to subsystem (in sections panel):
    - command presentation: subsystem presentation;
    - graphical presentation: Picture → standard picture of sections panel section;
    - command tooltip: Explanation.

NOTE. If none of the used (in and/or tooltips presentation generation) properties is specified, text presentation of metadata object is used as it is returned by the `Presentation()` method.

When placing a command into the actions panel or form command bar, the following rule applies: command group name (preceded by «:» symbol) is removed from command presentation text, if this combination is a completion of command presentation. For example, a standard command to create the `Goods` catalog item as Goods: create, will be displayed as Goods in actions panel. If you create a command represented as My command: generate and place it into the standard Generate group of the form command bar, the command presentation will be My command. At the same time, if this command is placed into the Print group (that is created by developer) in the form command bar, its presentation will remain as My command: generate.

Properties associated with objects presentation (see table above) are also used in form headers generation if Automatic header form property is set to `True`. In this case, the form header combines Header form property and an automatically generated header. If this property is set to `False`, the Header form property is used as a header without any changes.

When automatically generating a form header, 1C:Enterprise follows the following rules:

- list form header: Extended list presentation → List presentation;
- headers for catalog form, chart of accounts, chart of calculation types, exchange plan, chart of characteristic types, tasks:
  - existing item: text presentation of object (Extended object presentation → Object presentation), for example:
    - Zinoviev Anton (Employee);
    - 70.1 (Account);
  - new item: Extended object presentation (Creation) → Object presentation (Creation);
- document or business process form header:
  - existing item: text presentation of object, for example:
    - invoice № 12001 from 12.05.2008 16:15:32;
  - new item: Extended object presentation (Creation) → Object presentation (Creation);
- information register record form header:
  - existing record: Extended record presentation → Record presentation;
  - new record: Extended record presentation (Creation) → Record presentation (Creation);
- group form header of catalog or chart of characteristics type:
  - existing group: text presentation of object (Extended list presentation→ List presentation);
  - new group: Extended list presentation (Group creation) → List presentation (Group creation);
- form header of report or data processor: Extended presentation.

NOTE. If none of the properties used in name generation is specified, text presentation of metadata object is used as it is returned by the `Presentation()` method.

Appendix 4. List of automatically saved settings

The appendix describes settings automatically stored in 1C:Enterprise storage by the platform.

4.1. Storage: report options

Setting: report options

- Object key - full report name.

Example of an object key:

```
Report.Sales
```

- Setting key - text ID of option.

Example of the setting key:

```
SalesByRegion
```

- Saved value type - `DataCompositionSettings`.

4.2. Storage: report settings

Setting: report settings

- Object key - text line consisting of full report name, «/» symbol and option key.

Example of an object key:

```
Report.Sales/SalesByRegion
```

- Setting key - text ID of setting.

Example of the setting key:

```
MyClients
```

- Saved value type - `DataCompositionUserSettings`.

4.3. Storage: form data

Setting: form field value

- Object key - full form name.

Example of an object key:

```
Processing.DataExport.Form.DefaultForm
```

- Setting key - text ID of saved values.

Example of the setting key:

```
ExportToSberbank
```

- Saved value type - `Map`. Map key is a text path to the saved data attribute and its value is the attribute value.

4.4. Storage: dynamic list settings

Setting: dynamic list settings

- Object key - is a line value consisting of full form name and `DynamicList` attribute type name.

Example of an object key:

```
Document.GoodsArrival.Form.ListForm.List
```

- Setting key - text ID of saved values.
- Saved value type - `DataCompositionUserSettings`.

4.5. 1C:Enterprise storage

Setting: current report option key

- Object key - text line consisting of full report name and line /CurrentVariantKey.

Example of an object key:

```
Report.Sales/CurrentVariantKey
```

If the form property is set to `PurposeUseKey`, the object key is generated as follows: full report name, "/" symbol, `PurposeUseKey` property value and /CurrentVariantKey line.

Example of an object key:

```
Report.Sales/ForDiscountChargeForm/CurrentVariantKey
```

In this example `PurposeUseKey`property is set to ForDiscount-ChargeForm.

- Setting key - blank line.
- Saved value type - any value containing current option key.

Setting: current report setting key

- Object key - is a line value consisting of full report name, report option key (represented as a line) and /CurrentUserSettingsKey text (/Curren-tUserSettingsKey).

Example of an object key:

```
Report.Sales/SalesByRegion/CurrentUserSettingsKey
```

If the form property is set to `PurposeUseKey`, the object key is generated as follows: full report name, "/" symbol, report option key as line, "/" sym-bol, `PurposeUseKey` property value and /CurrentUserSettingsKey line.

Example of an object key:

```
Report.Sales/SalesByRegion/ForDiscountCalculationForm/CurrentUserSetti
ngsKey
```

In this example, report option key is set to SalesByRegion and `PurposeUseKey`property is set to ForDiscountChargeForm.

- Setting key - blank line.
- Saved value type - any value containing current option key.

Setting: report option setting on report closing or option changes

- Object key - is a line value consisting of full report name, report option key (represented as a line) and /CurrentUserSettings text (/CurrentUserSet-tings).

Example of an object key:

```
Report.Sales/SalesByRegion/CurrentUserSettings
```

If the form property is set to `PurposeUseKey`, the object key is generated as follows: full report name, "/" symbol, report option key represented as a line, "/" symbol, `PurposeUseKey` property value and /CurrentUserSet-tings line.

Example of an object key:

```
Report.Sales/SalesByRegion/ForDiscountCalculationForm/CurrentUserSettings
```

In this example, report option key is set to SalesByRegion and PurposeUseKeyproperty is set to ForDiscountChargeForm.

- Setting key - blank line.
- Saved value type - DataCompositionUserSettings.

Setting: form field current setting key

- Object key - is a line value consisting of full form name and /CurrentDataSettingsKey text (/CurrentDataSettingsKey).

Example of an object key:

```
Processing.DataExport.Form.DefaultForm/CurrentDataSettingsKey
```

If the form property is set to PurposeUseKey, the object key is generated as follows: full form name, "/" symbol, PurposeUseKey property value and /CurrentDataSettingsKey line.

Example of an object key:

```
DataProcessor.DataExport.Form.DefaultForm/ForDiscountChargeForm/CurrentDataSettingsKey
```

In this example PurposeUseKeyproperty is set to ForDiscountChargeForm.

- Setting key - blank line.
- Saved value type - any value containing the current setting key.

Setting: form field values on closing

- Object key - is a line value consisting of full form name and /CurrentData text (/CurrentData).

Example of an object key:

```
DataProcessor.DataExport.Form.DefaultForm/CurrentData
```

If the form property is set to PurposeUseKey, the object key is generated as follows: full form name, "/" symbol, PurposeUseKey property value and /CurrentData line.

Example of an object key:

```
DataProcessor.DataExport.Form.DefaultForm/ForDiscountChargeForm/CurrentData
```

In this example PurposeUseKeyproperty is set to ForDiscountChargeForm.

- Setting key - blank line.
- Saved value type - `Map`. Map item key is a text path to saved data attribute and its value is the attribute value.

Setting: form display setting

- Object key - is a line value consisting of full form name and /FormSettings text (/FormSettings).

Example of an object key:

```
DataProcessor.DataExport.Form.DefaultForm/FormSettings
```

If the form has a set `PurposeUseKey` property (property value is not blank), object key is generated as follows: full form name, "/" symbol, `PurposeUseKey` property value and /FormSettings line.

Example of an object key:

```
DataProcessor.DataExport.Form.DefaultForm/DefaultOption/FormSettings
```

In this example `PurposeUseKey` property is set to DefaultOption value.

- Setting key - blank line.
- Saved value type - `FormSettings`. Object with no properties and methods.

Setting: form and control size, separator location, status of the conversation panel (size and is it open/close) and table column width

- Object key - is a line value consisting of full form name and /WindowSettings text (/WindowSettings).

Example of an object key:

```
DataProcessor.DataExport.Form.DefaultForm/WindowSettings
```

If the form has a set `WindowOptionsKey` property, when operating with the form, object key includes this key value <WindowOptionKeyValue>/WindowSettings.
For example, if windows option key value is DefaultMode, the above example looks as follows:

```
DataProcessor.DataExport.Form.DefaultForm/DefaultMode/WindowSettings
```

- Setting key - blank line.
- Saved value type - `WindowSettings`. Object with no properties and methods.

To remove the settings of any form, perform this operation **before** the first opening of the form in current session.

Setting: application's main window settings

- Object key - line value MainWindow/WindowSettings (MainWindow/WindowSettings).
- Setting key - blank line.
- Saved value type - `WindowSettings`. Object with no properties and methods.

Setting: desktop separators location

- Object key - line value. Depending on the used compatibility mode:
  - Version 8.3.2 and later: Desktop/WindowsSettings (Desktop/WindowSettings).
  - Version 8.3.3 and later:
    - Regular interface: HomePage/WindowSettings.
    - Interface Taxi: HomePage/Taxi/WindowSettings.
  - Version 8.3.8 and earlier:
    - Regular interface: HomePage/WindowSettings (HomePage/WindowSettings).
    - Interface Taxi: HomePage/Taxi/WindowSettings (HomePage/Taxi/WindowSettings).
- Setting key - blank line.
- Saved value type - `WindowSettings`. Object with no properties and methods.

Setting: form and control size, separator location, status of the conversation panel (size and is it open/close) and table column width in web client

- Object key - is a line value consisting of full form name and /WebClientWindowSettings text (/WebClientWindowSettings).

Example of an object key:

```
DataProcessor.DataExport.Form.DefaultForm/WebClientWindowSettings
```

- Setting key - blank line.
- Saved value type - `WindowSettings`. Object with no properties and methods.

Setting: home page (desktop) settings

- Object key - line value. Depending on the used compatibility mode:
  - Version 8.3.2 and later: Common/DesktopSettings(Common/DesktopSettings).
  - Version 8.3.2 and earlier: Common/HomePageSettings.
  - Version 8.3.8 and earlier: Common/HomePageSettings(Common/HomePageSettings).

- Setting key - blank line.
- Saved value type - `HomePageSettings`. Object with no properties and methods.

Setting: favorites

- Object key - line value Common/UserWorkFavorites (Common/UserWorkFavorites).
- Setting key - blank line.
- Saved value type - `UserWorkFavorites`.

Setting: dynamic list settings

- Object key - is a line value consisting of full form name, `DynamicList` type attribute name and /CurrentUserSettings text (/CurrentUserSettings).

Example of an object key:

```
Document.GoodsArrival.Form.ListForm.List/CurrentUserSettings
```

If the form has a set `WindowOptionsKey` property, when operating with form, object key includes this key value <WindowOptionKeyValue>/CurrentUserSettings.
For example, if windows option key value is DefaultMode, the above example looks as follows:

```
Documents.GoodsArrival.Form.ListFor.List/DefaultMode/CurrentUserSettin
gs
```

- Setting key - blank line.
- Saved value type - `DataCompositionUserSettings`.

Setting: Search history in a form table

- Object key - Common/TableSearchHistory/<Form name>.<Table name> (Common/TableSearchHistory/<Form name>.<Table name>).

Example of an object key:

```
Common/TableSearchHistory/Catalog.Goods.Form.ListForm.List
```

- Setting key - blank line.
- Saved value type - `TableSearchHistory`. Object with no properties and methods.

Setting: global command interface settings

- Object key - line value, one of the options:

- Common/SectionsPanel/CommandInterfaceSettings (Common/SectionsPanel/CommandInterfaceSettings) - sections panel settings;
- full name of command interface fragment and /ActionsPanel/CommandInterfaceSettings (/ActionsPanel/CommandInterfaceSettings) - subsystem actions panel settings;
- full name of command interface fragment and /NavigationPanel/CommandInterfaceSettings (/NavigationPanel/CommandInterfaceSettings) - subsystem navigation panel settings;
- Desktop/ActionsPanel/CommandInterfaceSettings (Desktop/ActionsPanel/CommandInterfaceSettings) - desktop actions panel settings (in version 8.3.2 and later compatibility mode);
- MainSection/ActionsPanel/CommandInterfaceSettings (MainSection/ActionsPanel/CommandInterfaceSettings) - desktop actions panel settings (in versions earlier than 8.3.2);
- Desktop/NavigationPanel/CommandInterfaceSettings (Desktop/NavigationPanel/CommandInterfaceSettings) - desktop navigation panel settings (in versions 8.3.2 and later compatibility mode);
- MainSection/NavigationPanel/CommandInterfaceSettings (MainSection/NavigationPanel/CommandInterfaceSettings) - desktop navigation panel settings (in versions earlier than 8.3.2 compatibility mode).

Example of an object key:

```
Subsystem.Sales/NavigationPanel/CommandInterfaceSettings
Desktop/ActionsPanel/CommandInterfaceSettings
```

- Setting key - blank line.
- Saved value type - `CommandInterfaceSettings`. Object with no properties and methods.

Setting: spreadsheet document print settings

- Object key - is a line value consisting of Common/SpreadsheetDocumentPrintSettings/ (Common/SpreadsheetDocumentPrintSettings/) and print parameter name.

Example of an object key:

```
Common/SpreadsheetDocumentPrintSettings/PurchaseInvoice
```

- Setting key - blank line.
- Saved value type - `SpreadsheetDocumentPrintSettings`. Object with no properties and methods.

Setting: client application settings

- Object key - line valueCommon/ClientSettings (Common/ClientSettings).
- Setting key - blank line.
- Saved value type - `Common/ClientSettings`. Object with no properties and methods. Contains the flags that define whether the All functions command is displayed, whether the debug mode is set on startup, whether low speed connection mode is set (for the web client), External flag of the dialog box parameter of link acquisition.
  For Taxi interface in these settings, panel changes relative to content and location specified in Designer are also stored.

Setting: client application interface settings

- Object key - line value Common/ ClientApplicationInterfaceSettings (Common/ ClientApplicationInterfaceSettings).
- Setting key - blank line.
- Stored value - `ClientApplicationInterfaceSettings` object. Contains application panel location settings.

Setting: add-in settings

- Object key - line value AddInSettings/<AddInSettingsName> (AddInSettings/<AddInSettingsName>). Line value <AddInSettingsName> is defined by the add-in using the`RegisterProfileAs()` method.

Example of an object key:

```
AddInSettings/BarCodeScanner
```

- Setting key - blank line.
- Stored value type - `AddInSettings`. Object with no properties and methods.

Setting: help settings

- Object key - line value Common/HelpSettings (Common/HelpSettings).
- Setting key - blank line.
- Stored value type - `HelpSettings`. Object with no properties and methods.

Setting: color settings of file comparison parameters

- Object key - line value Common/ComparisonSettings (Common/ComparisonSettings).
- Setting key - blank line.

- Stored value type - `ComparisonSettings`. Object with no properties and methods.

Setting: search history

- Object key - line value Common/TableSearchParameters (Common/TableSearchParameters).
- Setting key - blank line.
- Stored value type - value list containing history lines list. The latest items are in the beginning of the list.

Setting: history of value choice in managed form input field

- Object key - ChoiceHistory (ChoiceHistory) line value.
- Setting key - blank line.
- Stored value type - `ChoiceHistorySettings` object. Object with no properties and methods.

Setting: result reduction width used for the report formation

- Object key - text line consisting of full report name and depending on the client application:
  - Thin and thick client: (/ResultReductionWidthThinClient).
  - Web client: (/ResultReductionWidthWebClient).
  - Mobile client: (/ResultReductionWidthMobileClient).
  - Mobile platform: (/ResultReductionWidthMobilePlatform).
- Saved value type - `Number.`

Setting: global search history

- Object key - text line (Common/GlobalSearchHistory).
- Saved value type - `GlobalSearchHistory`. Object with no properties and methods.

Setting: Consistency of series objects and colors to be displayed in charts

- Object key- text string Common/ChartColors (Common/ChartColors).
- Kind of object being saved:- `ValuesAppearance`.

Appendix 5. Full-text search expressions

You can search by multiple words using search operator and search by exact phrases. By default, search with synonyms considered and fuzzy search are not used. To access these features, use specific search operators.
Following search operators are used in input line (both for Help and Syntax Assistant search and for 1C:Enterprise language `FullTextSearch` object):

Operator	Expression example	Description

`Space`
`AND`
`AND`
`&`
	Business Success
	Business `And` Success
	Business `AND` Success
	Business `&` Success
		Both "Business" and "Success" words must be included

`Or`
`OR`
`|`
`,`
	Business `OR` Success
	Business `OR` Success
	Business `|` Success
	Business `,` Sucess
		At least one of "Business" and "Success" words must be included

`NOT`
`NOT`
`~`
	Business `NOT` Success
	Business `NOT` Success
	Business `~` Success
		"Business" word must be included, while "Success" word must not be

`NEAR/[±]n`	*Example 1:* dryer `NEAR/3` air *Example 2:* dryer `NEAR/+3` air *Example 3:* dryer `NEAR/-3` air	Searching for data containing specified words in one attribute with morphology considered at the distance of n words between words. Sign determines the relative position of the second word («`+`» - if the second word comes after the first one; «`-`» - if the second word comes before the first one).

Operator	Expression example	Description
		If a sign is not specified, data containing specified words that are separated by n words will be found. their order does not matter. In example 1, data where the "air" word comes before or after the "dryer" word and is separated from it by a maximum of 1 words will be found. In example 2, data where the "air" word comes after the "dryer" word and is separated from it by a maximum of 2 words will be found. In example 3 data where "air" word comes before "dryer" word and is separated from it by a maximum of 3 words will be found
NEAR	posting NEAR document	A simplified version of the previous operator. The query will find data where the words are present in the same attribute and separated by a maximum of 8 words
""	«network administrator»	Exact words search (equivalent to «network NEAR/+1 administrator»)
( )	(technology | production) & (cheese | cottage cheese)	Use to specify the order of operators (the number of nesting levels is not limited).
*	arch*i arch*&documentoob*	Searching using a wildcard - the "*" is supported in any place of a word and in any quantity. Thus, "arch*i" query can find "archive" and "archeology"

Operator	Expression example	Description
#Hashtag #Hasht*	#Order #Order*	Hashtag is searched (#Hashtag) and all hashtags starting from a specific substring (#Hasht*).

THE FOLLOWING OPERATORS MUST BE PROVIDED IN CAPITAL LETTERS: AND, OR, NOT, NEAR. Unary operators (operators at the beginning of search strings) are not supported. For example, you cannot select all the chapters, where the specified text is missing. All characters in the search field, except for search operators, letters and numbers characters, are ignored.

When using the 1C:Enterprise language FullTextSearch object keep in mind the following:

Operator	Expression example	Description
*	arch * arch*&documentoob*	Searching using a wildcard - only one "*" character is supported and it should be at the end of the word. Example: the search for "post*" returns "posting" and "posted".
~	~System System~2	Fuzzy search of words with a number of differences from the specified in the search line. "~System" query (equivalent to "System~1") finds "systwm" and "sydtem". "System~2" query finds "syttwm" and "suttem". Only for full-text search in data!

Operator	Expression example	Description
!	! red tile	The search is performed with synonyms of English, Russian and Ukrainian languages considered. "!" should precede the word whose synonyms you want to find. Example: search query "!red tile" will also find "scarlet tile" and "coral tile"

THE FOLLOWING OPERATORS MUST BE PROVIDED IN CAPITAL LETTERS: AND, OR, NOT, NEAR. Unary operators (operators at the beginning of search strings) are not supported. For example, you cannot select all the chapters, where the specified text is missing.

To search for special characters used in text enclose them in quotation marks. Example: "NEAR".

Appendix 6. Access rights description

This appendix describes all access rights that can be managed when editing roles in 1C:Enterprise. Access rights supported in mobile platform are marked with "*" symbol.

- `Automation` (`Automation`) - using automation server.
- `Administration` (`Administration`):
  - Administrative rights for the whole infobase and user list editing (configuration right). `DataAdministration` right is required.
  - Connection parameters setting feature (for external data source).
  - Display of license list, used when operating with a configuration and infobase, in About 1C:Enterprise window.
  - Removing data areas (including removing all areas at once).
  - Infobase configuration background update feature (on client side).
  - Security profile name setting feature (in extensions management dialog box).
  - More detailed information is in the event log.
  - Database copy management.
- `DataAdministration` (`DataAdministration`) - this right regulates administrative operations with data. These operations include:
  - User can view the following event log events: Authentication, Authentication error (for Session group), Access, Access denied (for Access group), Confirmed, Rejected (for OpenID provider group).
    User can get event filter values without restrictions.
    However, to view the event log, `EventLog` right is required.
  - Infobase time zone (data areas) setting - call `SetInfoBaseTimeZone()` method.
  - Creating initial image of distributed infobase subordinate node (call `CreateInitialImage()` method).
  - Objects numbering update (call `RefreshObjectsNumbering()` method).
  - executing the `StandardSettingsStorageManager` object methods, if non-current user settings are operated.
  - Execution methods `GetInfoBaseRegistrationData`/`SetInfoBaseRegistrationData()`.
  - Management of the properties of the tables included in the database copy.

- ConfigurationExtensionsAdministration (ConfigurationExtensionsAdministration) - manages access to extensions management interface (both manual and program) in 1C:Enterprise mode. Extension administration can not be performed in safe mode.
- ActiveUsers (ActiveUsers) - view active users list.
- InputByString* (InputByString) - use of input by string mode.
- WebClient (WebClient) - right to start a web client.
- ExternalConnection (ExternalConnection) - external connection.
- Output* (Output) -printing, recording and copying to clipboard.
- Execute (Execute) - task execution.
- Insert (Insert) - insertion (main)
- EventLog (ExentLog) - right to view the event log. Completeness of event log display is controlled by the Administration and DataAdministration rights.
- Update (Update) - change (main).
- StandardAuthenticationChange (StandardAuthenticationChange) - user has the right to change his saved standard authentication parameters (Standard authentication, UserName Password properties) of external data source.
- SessionOSAuthenticationChange (SessionOSAuthenticationChange) - user has the right to change external data source OS authentication parameters in his settings and in session parameters.
- UpdateDataHistory (UpdateDataHistory) - allows to change data history (to create new version or to remove existing one).
- UpdateMissingDataHistory - allows to change data history (to create new version or to remove existing one) for missing data. Missing data - is removed data or data that is unavailable because of data access restriction.
- UpdateDataHistorySettings (UpdateDataHistorySettings) - allows to change the data history settings.
- UpdateDataHistoryVersionComment (UpdateDataHistoryVersionComment) - allows to change data version comment.
- SessionStandardAuthenticationChange (SessionStandardAuthenticationChange) - user has the right to change standard authentication parameters (Standard authentication, UserName, Password properties) of external data source for current session.
- InteractiveActivate (InteractiveActivate) - interactive activation.

- InteractiveUndoPosting* (InteractiveUndoPosting) - interactive posting cancellation.
- InteractiveSetDeletionMark* (InteractiveSetDeletionMark) - interactive deletion mark.
- InteractiveSetDeletionMarkPredefinedData* (InteractiveSetDeletionMarkPredefinedData) - allows interactive setting of deletion mark for predefined data.
- InteractiveExecute (InteractiveExecute) - interactive task execution.
- InteractiveInsert* (InteractiveInsert) - interactive insertion.
- InteractiveChangeOfPosted* (InteractiveChangePosted) - interactive editing of posted document. If the right is not set, the user cannot remove posted document, set deletion mark, repost or clear posting. Form of this document is opened in view mode.
- (InteractiveOpenExtDataProcessors)- interactive opening of external data processors.
- InteractiveOpenExtReports (InteractiveOpenExtDataReports) - interactive external reports opening.
- InteractivePosting* (InteractivePosting) - interactive posting.
- InteractivePostingRegular (InteractivePostingRegular) - interactive document posting (by standard form commands) in non-real-time mode.
- InteractiveClearDeletionMark* (InteractiveClearDeletionMark) - interactive deletion mark removing.
- InteractiveClearDeletionMarkPredefinedData* (InteractiveClearDeletionMarkPredefinedData) - allows to remove the deletion mark from predefined data interactively.
- InteractiveDelete (InteractiveDelete) - direct interactive removing.
- InteractiveDeleteMarked (InteractiveDeleteMarked) - interactive removing of marked objects.
- InteractiveDeleteMarkedPredefinedData* (InteractiveDeleteMarkedPredefinedData) -allows to remove the marked predefined data interactively.
- InteractiveDeletePredefinedData (InteractiveDeletePredefinedData) - allows to remove the predefined data interactively.
- InteractiveStart (InteractiveStart) - interactive start of business process.

- Use (Use) - use of some particular configuration objects, for external data source function - allows to use them from 1C:Enterprise language and query language.
- ExclusiveMode (ExclusiveMode) - use of exclusive mode.
- UpdateDataBaseConfiguration (UpdateDataBaseConfiguration) - allows to update the database configuration. To perform an update of configuration on server or background configuration update on server, it is enough to have only this right, Administration right is not required.
- UndoPosting (UndoPosting) - document posting cancellation.
- SwitchToDataHistoryVersion - gives an interactive right to restore versioned object attributes at the moment of time corresponding to current version. If this right is missing, Switch to version command is unavailable in object changes history form.
- Get (Get) - getting the value that is not stored in the database.
- Posting (Posting) - document posting.
- View* (View) - viewing. In Select type dialog box, types that user has no right to use are unavailable. For external data source function - allows to use function in data composition system field list.
- ViewDataHistory (ViewDataHistory) -gives an interactive right to view data history. If this right is missing, Update history command for object form and list form is unavailable.
- CollaborationSystemInfoBaseRegistration (CollaborationSystemInfoBaseRegistration) - user having this right can register an infobase in 1C:Dialog service.
- Edit* (Edit) - editing.
- EditDataHistoryVersionComment (EditDataHistoryVersionComment) - gives an interactive right to change data version comment. If this right is missing, editing of version comment in data history version list is unavailable.
- AllFunctionsMode* (AllFunctionsMode) -manages All Functions command availability in Service and Settings (main menu) and availability of corresponding client application settings.
- MainWindowModeEmbeddedWorkplace (MainWindowModeEmbeddedWorkplace) - enables it to start the main window of a client application in the embedded workplace mode.
- MainWindowModeKiosk (MainWindowModeKiosk) - enables it to start the main window of a client application in the kiosk mode.
- MainWindowModeNormal (MainWindowModeNormal) - enables it to start the main window of a client application in the normal mode.

- `MainWindowModeFullscreenWorkplace` (`MainWindowMode-FullscreenWorkplace`) -enables it to start the main window of a client application in the full screen workplace mode.
- `MainWindowModeWorkplace` (`MainWindowModeWork-place`) - enables it to start the main window of a client application in the workplace mode.
- `SaveUserData` (`SaveUserData`) - manages the capability to save user settings. If this right is disabled, then:
  - History button is unavailable, accessing history from 1C:Enterprise language invokes an exception, interactive recording operations are not recorded in history.
  - Calling favorites list is unavailable in 1C:Enterprise command bar. Add to favorites button is unavailable in Get link and Go to link forms. When referring to favorites, an exception is invoked from 1C:Enterprise language.
  - Form setting is unavailable (Change form command is missing).
  - Setting commands for Section panel, navigation panel and home page are unavailable.
  - User settings and report options saving commands are unavailable in reports. A prompt to save the changes performed by a user in the current report option is not shown. Changes are not saved Select report settings command is available only if user settings storage is set for the report or configuration. When the `SetCurrentUserSettings()` method is called, an exception is raised.
  - Save and restore form data commands are unavailable (the Save parameters and Restore parameters commands). These forms are not saved automatically.
  - Window settings (size and position) are saved only during the session.
  - Main window panel size settings are saved only during the session.
  - When settings are programmatically saved, an exception is raised.
  - Client application settings are saved only during the session. The Allow debug mode on startup check box is not shown (for thin client and thick client). The Set Slow connection speed on startup check box is not shown (for a web client).
  - Period settings in the list are not saved. The Use this period setting upon opening check box is unavailable in the period settings dialog box.
- `Start` (`Start`) - start the business process.
- `ThickClient` (`ThickClient`) - the right to start a thick client.
- `ThinClient` (`ThinClient`) - the right to start a thin client.
- `Delete` (`Delete`) - deletion (main).

- `TotalsControl` (`TotalsControl`) - managing accounting register totals and accumulation register totals (setting the period until which the totals are calculated and totals recalculated).
- `Set` (`Set`) -setting a value that is not saved in the database.
- `Read` (`Read`) - reading (main).
- `ReadDataHistory` (`ReadDataHistory`) - allows reading of data history.
- `ReadDataHistoryOfMissingData` (`ReadMissingDataHistory`) - allows reading of data history records for missing data. Missing data - is removed data or data that is unavailable because of data access restriction.

Appendix 7. System behavior features in different modes

7.1. Features of slow connection speed mode

When operating in slow connection speed mode there are several system behavior features, that are listed below:

- Home page forms are not displayed automatically. To display the desktop, press the Show Desktop hyperlink located in the top left corner of the desktop working area. In Taxi interface, the home page is displayed instantly.
- When a form is generated, presentations for form reference data is obtained (no additional calls are made to the server to acquire presentations after opening the form).
- When a form is generated, "over-the-period" form data values displayed by form items are obtained (no additional calls are made to the server to acquire these values after opening the form).
- The server is notified about the closure of the form (as well as unlocking the object if it is locked) in the following cases:
  - receiving a form from a server using the `OpenForm()`, `GetForm()` methods;
  - calling server (context and out-of-context) form methods;
  - closing 20 forms;
  - system is in waiting status for more than 20 seconds;
  - when the following dynamic list commands are executed:
    - deleting an item;
    - marking an item for deletion;
    - posting or canceling of posting of a document;
    - moving a catalog item to another group.
- Settings are only saved in case they have been changed and it does not happen instantly after closing the form, but is delayed -the next time you open any form, or when you accumulate 20 delayed saving settings, or after 20 minutes (when 1C:Enterprise is idle), or when the application is closed.
- Client application does not receive or display section panel pictures.
- The input field does not generate a selection list while waiting in string input process.

- An appropriate data cache is generated for quick selection and input-by-string lists. Cache data is stored for the duration of the session. A maximum of 20 latest lists can be stored for each object type. Cache can store a maximum of 200 lists.
  Every 20 minutes cache is cleared of obsolete data related to objects that have changes displayed on the client side in this session.
  Repeated selection or input by string with the same parameters without moving to another field operations create a new data query to the server.
- In normal connection speed mode, a dynamic list search is performed during input process, with a standard delay. In slow connection speed mode, the search is performed after the search string loses the input focus.

7.2. Compatibility mode features

7.2.1. Previous versions

A number of system mechanisms have different behavior that depends on the Compatibility mode configuration property. This section lists these possible behavior differences.

- Default start mode configuration feature cannot be set to Managed application value when the Compatibility mode configuration property is set to Version 8.1 value. These settings are checked when updating the configuration database.
- In compatibility mode with version 8.1, the `Subsystems` property of the metadata objects that could belong to subsystems is available from 1C:Enterprise language. Moreover, in compatibility mode with version 8.1 the `Components` configuration property is available, which is populated with references to objects belonging to the root element of the subsystem tree. When compatibility mode with version 8.1 is turned off, the Components configuration property will be cleared and will become unavailable in the configuration properties panel.
- In ordinary forms, object filling is similar to Version 8.1, i.e. the filling handler is called only when entering on basis and `Fill ()` method call.
- The `CheckFilling()` method is unavailable for metadata objects, automatic checking is not performed.
- The call for help from the form is performed in the same way as in version 8.1, i.e. a dialog box of help section selection is displayed if there is no help information for this form.
- Document journal tables do not contain the `Type` virtual field.

- Infobase table list of the query wizard displays tables and fields for which the current user does not have viewing rights.
- The data composition system does not check interactive rights for tables.
- Pictures selected in the picture library and inserted into an external report or processing are converted to external pictures.
- The Type property is available DataCompositionAvailableField,DataCompositionFilterAvailableField and DataCompositionAvailableParameter objects.
- The DataVersion property is unavailable for catalogs, documents, charts of characteristic types, charts of accounts, charts of calculation types, exchange plans, business processes and tasks. Access restrictions to data for DataVersion field are checked based on the restrictions set in the Reference field.
- Setting theInteractiveInsert, InteractiveDelete, InteractiveSetDeletionMark, InteractiveClearDeletionMark, InteractiveDeleteMarked, InteractivePosting, InteractivePostingRegular, InteractiveUndoPosting, InteractiveChangeOfPosted, InteractiveStart, InteractiveActivate,InteractiveExecute rights is not related to setting the Edit right.
- When downloading configuration files, ordinary application module is downloaded to Configuration.ApplicationModule.txt file.
- Formatting of Number, Date and Boolean type values is performed in accordance with the parameters specified in the regional infobase settings.
- In the color selection dialog box, web colors are ordered by their English names.
- You can assign any values to Output and PageOrientation spreadsheet document properties. Values of unsuitable type are ignored.
- You can interactively select the following pictures DocumentObject, ChartOfCharacteristicTypesObject, ScheduledJobs, CatalogObject, BusinessProcessObject, TaskObject, ChartOfCalculationTypesObject , ExchangePlanObject, ChartOfAccountsObject , InformationRegisterRecord, ChooseFromList, AddListItem, NewFolder, ChangeListItem, SetListItemDeletionMark , DeleteListItemDirectly, DeleteListItem .
- Data composition system comparison type Contains counts _, % and [characters as special characters.

- CAST operation of the query language returns a fixed-length string (with trailing spaces) when casting to String type.
- Print parameters settings for separate spreadsheet documents (objects) are not different and are changed simultaneously (even if the print parameters key is the same).
- A query that has DISTINCT keyword specified and the ORDER BY sentence has an expression specified that is not in the selection list, is considered correct, and an error will not be raised when executing such query.
- Nested query balance fields in external query are not treated as balance fields and totals for these fields are calculated in the same way as for standard fields.
- Maximum size of String of Fixed Size metadata attribute is 1024 characters.
- For new accumulation and accounting registers, the totals splitting flag is not set by default.
- Representation of complete module names in technological information (technological log, event log, etc.) is changed - there is no module specification in names.
- Checking the lock values for correspondence to the type of lock field in the control lock items when the Lock() method is executed is performed only for the first lock value for each item.
- Changing a synonym in the form and template wizards changes the name, and changing the name will change the synonym, if none of these autogenerated field values has been changed after opening the wizard.
- The picture editor does not support operations with alpha channel. There is no conversion of pictures to PNG format after the completion of editing.
- Title of an active table column in a form is highlighted.
- Pressing Tab or Shift + Tab keys in a form table that contains no rows or with the Row selection mode leads to activation of the next or previous column of the table respectively. The next or previous form item is activated only after you reach the last column of the table.
- These properties become available for spreadsheet documents GroupBackColor, GroupTextColor , HeaderBackColor, HeaderTextColor.
- If you select a custom border or border color for a form, or a custom button background color for Button, Input field or Slider, the form item is displayed in the selected color and the border does not have rounding.
- BeforeClose form handler is not called if modal form closes on timeout.
- When downloading an event log in XML format, the UserName field displays the full username.
- User with no administrative rights can change the CannotChangePassword property of the InfoBaseUser object.

- When a template is received, on server and in an external connection the default configuration language is set as the template language.
- The Data Path property of the data composition template user field is generated in English.
- When an ordinary form of an information register record is opened, the `Period` property value is set to match the current data, but only if this property is not filled and the form record is opened from a form list. In all other cases, this property is not filled.
- Components of tables describing scheduled jobs are changed:
- Compatibility mode is used - all scheduled jobs are stored in one table.
- Compatibility mode is not used - each scheduled job is stored in a separate table.
- Components of tables storing constants are changed:
  - Compatibility mode is used - one table is used to store constants. The same table is used to register changes.
  - Compatibility mode is not used - each constant is stored in its native table. Changes for each constant are also registered in a native table.
- Components of tables storing accumulation register settings are changed:
  - Compatibility mode is used - accumulation register settings are stored in one table.
  - Compatibility mode is not used - the settings of each register are stored in a native table.
- Components of tables storing accounting register settings are changed:
  - Compatibility mode is used - accounting register settings are stored in one table.
  - Compatibility mode is not used - the settings of each register are stored in a native table.
- Help inclusion for a subsystem in help components was changed:
  - Compatibility mode is used - subsystem help was shown in help contents, if the Include in help components property was set for any object included in the subsystem, and it was disabled for the subsystem.
  - Compatibility mode is not used - inclusion of subsystem help in help components completely depends on the Include in Help components subsystem property status and does not depend on the properties of the objects included in the subsystem.
- A number of resources displayed in the chart by the data composition system was changed:
  - Compatibility mode is used - only one resource (the first one) is displayed in the chart.
  - Compatibility mode is not used - several resources are displayed in the chart.
- The option to group by data composition system field resources was changed:

  - Compatibility mode is used - grouping by field resources is not supported.
  - Compatibility mode is not used - grouping by field resources is available.
- Diagnostics of key values identifying a string displayed for a dynamic list with a set arbitrary query was changed:
  - Compatibility mode is used - errors occurring when receiving dynamic list data are not displayed.
  - Compatibility mode is not used - if duplicated key values are found in the selection, a warning is shown and data display becomes impossible.
- The binding of `Period` parameter of the periodic information registers manager method `Get()` was changed:
  - Compatibility mode is used - the parameter is optional and the method execution result is not defined.
  - Compatibility mode is not used - the option is required.
- System behavior was changed for a scenario when the form item size is determined automatically, if the font size specified in the form item properties is not the default font size:
  - Compatibility mode is used - item font size is not considered when form item size is calculated.
  - Compatibility mode is not used - item font size is considered when item size is calculated. Font changes performed using conditional appearance of the form or dynamic list do not affect form item size.
- `DataVersion` standard attribute text representation was changed:
  - Compatibility mode is used - representation is equal to a blank string.
  - Compatibility mode is not used - representation is a string containing 12 white spaces.
- When transferring the value from an attribute to a form item, the updated text in the input field was changed:
  - Compatibility mode is used - update always takes place, except for the case when `AutoComplete` event handler sets the same type of value that is set in the value input field. The `EditTextUpdate` property is ignored.
  - Compatibility mode is not used - updating can be managed with the `EditTextUpdate` input field property and the `UpdateEditText()` method of the input field form extension.
- `When simultaneosly using` SetValue() `and` UseFromDataSource() `methods of` DataLockItem `object` :
  - Compatibility mode is used - the value set by the `UseFromDataSource()` method has priority.
  - Compatibility mode is not used - an exception is raised.
- Keys of objects designed to store desktop command interface settings were changed:

    - Compatibility mode is used:
        - The Subsystem.desktop/NavigationPanel/CommandInterfaceSettings key is used for navigation panel settings .
        - The Subsystem.desktop/ActionsPanel/CommandInterfaceSettings key is used for actions panel settings .
    - Compatibility mode is not used:
        - The Desktop/NavigationPanel/CommandInterfaceSettings key is used for navigation panel settings .
        - The Desktop/ActionsPanel/CommandInterfaceSettings key is used for actions panel settings .
- The algorithm for automatic filling of form attribute list that has to be checked was changed:
    - Compatibility mode is used - attributes disabled by functional options are included in the list of checked attributes.
    - Compatibility mode is not used - attributes disabled by functional options are not included in the list of checked attributes.
- Reading data for managed form tables linked with dynamic lists, that is executed when the form is opened:
    - Compatibility mode is used - reading is performed for all tables linked with dynamic lists, including tables that are not visible to the user.
    - Compatibility mode is not used - reading is performed only for tables displayed to the user when the form is opened.
- Converting the Name in data source property of external data source table field when placing in the SQL query:
    - Compatibility mode is used - the property value is always enclosed in double quotes, regardless of the character components in the name.
    - Compatibility mode is not used - the property value is enclosed in double quotes only in case the field contains special characters and is not enclosed in single quotes.
- Defining the calculated field column type when data composition results are outputted to the collection of values:
    - Compatibility mode is used - the column type is defined automatically on the basis of the calculated field expression.
    - Compatibility mode is not used - the value type specified for the calculated field is set as the value type of the column. If the value type is not specified, then the type is defined based on the calculated field expression.
- Outputting data composition results to the collection of values in case one data composition field is included in different grouping:
    - Compatibility mode is used - different grouping fields that are referencing to one data composition field were included in different collection columns.

- Compatibility mode is not used - different grouping fields that are referencing to one data composition field were included in different collection columns.

With 8.2.16 version

- Formation of arbitrary representation of data object:
  - Compatibility mode with 8.2.16 version is used - formation of arbitrary representations is not supported.
  - Compatibility mode is not used - formation of arbitrary representations is supported.
- Output of additional data for hierarchical records:
  - Compatibility mode with 8.2.16 version is used - data for hierarchical recording is used.
  - Compatibility mode is not used - data for detailed records is used.

With 8.3.2 version

- Acquiring the value for the functional option, acquisition of which is set to privileged mode:
  - Compatibility mode with 8.3.2 version is used -An error is generated in safe mode in case there are no access rights to an object storing the functional option value to be acquired.
  - Compatibility mode is not used - rights verification is not performed.

With 8.3.3 version

- Use of `DefaultRoles` feature:
  - Compatibility mode with 8.3.3 version is used -only one role can be included in the property. When switching to compatibility mode with version 8.3.3, it is checked that only one role is specified in the `DefaultRoles` property.
  - Compatibility mode is not used - several roles can be selected.
- Available dynamic list settings that are disabled using functional options:
  - Compatibility mode with 8.3.3 version is used -are not deleted from the available settings list.
  - Compatibility mode is not used - are deleted from the available settings list. In some cases, if there are settings that refer to the fields that are disabled using functional options, an error may be diagnosed field not found.
- Automatic name of the button associated with the Cancel command of the report managed form field extension, report option managed form field, and the settings managed form:
  - Compatibility mode with version 8.3.3 is used - `FormCancel` (`FormCancel`) name was assigned to the button.
  - Compatibility mode is not used - `FormCancelEdit` `FormCancelEdit` name is assigned to the button.

With 8.3.4 version

- Calculation of data composition system expression:
  - Compatibility mode with version 8.3.4. is used - calculation of expression is performed with the current user rights.
  - Compatibility mode is used - in calculation of expression application starts safe mode.
- Dragging in the table of managed form:
  - Compatibility mode with version 8.3.4. is used - dragging starts from any table cell. If the cell where dragging starts from is available - firstly the cell is activated and then - dragging itself. If the cell is unavailable - the current active cell of managed form table is dragged.
  - Compatibility mode is not used - dragging can be used from an available cell of managed form table only.
- Generation of data composition template from predefined templates:
  - Compatibility mode with version 8.3.4 is used - when generating data composition template from predefined templates details expressions are copied completely.
  - Compatibility mode is not used - in generation of data composition template from predefined templates, details expressions containing fields unavailable due to functional options are removed.
- Text parameter of data composition system conditional appearance that is defined for the area of use in the fields header:
  - Compatibility mode with version 8.3.4. is used - is not applied to grouping fields headers.
  - Compatibility mode is not used - applied to grouping fields headers.
- Behavior of `GroupPlacement` output parameter of data composition system:
  - Compatibility mode with version 8.3.4. is used - common grouping total is not displayed if `GroupPlacement` output parameter of data composition system is set to `End` or `No` for selected grouping.
  - Compatibility mode is not used - does not affect displaying of overall totals.
- Behavior of the template data composition generator for nested reports:
  - Compatibility mode with version 8.3.4. is used - the parent report groupings are not applied in nested totals reports.
  - Compatibility mode is not used - always generates `VerticalLevel` property for nested reports of data composition system templates. The parent report level containing nested report is added to this property.
- Adding strings to form table:
  - Compatibility mode with version 8.3.4. is used - a table string is automatically removed if it was added on previous string editing completion or when pressing the Down arrow button and if this string was edited manually after adding completion.

  - Compatibility mode is not used - a table string is automatically removed if it was added on previous string editing completion or on hitting the Down arrow button and if this string was not edited both programmatically and manually after adding completion.
- Limit of maximum managed form PurposeUseKey property value length:
  - Compatibility mode with version 8.3.4. is used - no length limit is set.
  - Compatibility mode is not used - value length of this attribute is limited by 128 characters.
- Additional method parameters FindMarkedForDeletion(), FindByRef, DeleteObjects():
  - Compatibility mode with version 8.3.4. is used- additional parameters are not used. Their usage causes a runtime error.
  - Compatibility mode is not used - additional parameters are used.
- Automatic generation of form displaying setting key:
  - Compatibility mode with version 8.3.4. is used - object key is generated as <Full form name>/FormSettings.
  - Compatibility mode is not used - object key is generated as <Full form name>/<PurposeUseKey>/FormSettings.
- Available settings of reports and dynamic lists:
  - Compatibility mode with version 8.3.4. is used - settings items that contain links to unavailable fields, disabled by functional options or unavailable according to access rights are removed.
  - Compatibility mode is not used - settings items that contain links to unavailable fields, disabled by functional options unavailable according to access rights or associated with non-existent data composition schema fields are removed.
- Availability of the <Type>Object and DynamicList managed form main attribute:
  - Compatibility mode with version 8.3.4. and later is used - in version 8.2 <Type>Object form's main attribute was not disabled even when corresponding configuration object was disabled by any functional option.

    In version 8.3 <Type>Object form's main attribute was disabled when corresponding configuration object was disabled by functional option. All functional options were analyzed.

    In previous 1C:Enterprise versions DynamicList main attribute was disabled when the configuration object, specified as main table of dynamic list, was disabled by functional option. All functional options were analyzed.

- Compatibility mode is not used - `<Type>Object` (`CatalogObject`, `DocumentObject` etc.) managed form's main attribute is disabled when corresponding configuration object is disabled by functional option.
  The main attribute of the managed form of the `DynamicList` type will be disabled if the functional option disables the configuration object that is specified as the main table of the dynamic list.
  In both cases only functional options without parameters are analyzed.

7.2.2. With version 8.3.5

- The button of managed form associated with command:
  - Compatibility mode with version 8.3.5 is used - always displayed as available, regardless of command availability.
  - Compatibility mode is not used - the button is unavailable if the current user cannot use the command associated with the button.
- Managed form's behavior when creating a source of dynamic list's available settings fails:
  - Compatibility mode with version 8.3.5 is used - when creating a source of data composition's available settings for dynamic list fails, no errors are generated, and managed form operating continues as if there is no available fields in source of dynamic list data. As a result, all form items, associated with dynamic list data, are removed from managed form.
  - Compatibility mode is not used - when creating a source of dynamic list's available settings fails, an exception is generated and the form is not created.
- Filling values of independent and shared separators in information register's record set record:
  - Compatibility mode with version 8.3.5 is used - values of separators containing information register are always replaced by values of separators set in the current session.
  - Compatibility mode is not used - all default values of independent and shared separators containing information register are replaced by values of separators set in the current session.
    If a separator value does not match the default value of separator type or is not equal to the value of separator set in the current session - error is generated.
- Behavior of system when operating with an `xBase` object in case of `.dbf` file operating error:
  - Compatibility mode with version 8.3.5 is used - in case of an error, dialog boxes were generated.

  - Compatibility mode is not used - in case of an error, an exception is generated.
- Idle handler operations (attached by `AttachIdleHandler()`) on temporary locks (by the Temporary Locks of a command or by the `LockApplication()` method) client application interface:
  - Compatibility mode with version 8.3.5 is used - handlers operations do not depend on temporary locks of client application.
  - Compatibility mode is not used - idle handlers are not called.
- Use of the value specified in the enabled parameter of the Text conditional appearance:
  - Compatibility mode with version 8.3.5 is used - not applied to predefined templates fields of data composition system groupings.
  - Compatibility mode is not used - is applied to predefined templates fields of groupings and data composition system report fields.
- Behavior of the output processor when exporting the data composition result to the value collection if there are columns with duplicated names:
  - Compatibility mode with version 8.3.5 is used - column uniqueness was not checked, columns are always created.
  - Compatibility mode is not used - uniqueness of column name in collection is checked. Columns with duplicated names are not re-created.
- Exporting data composition result to the value collection if there are nested reports:
  - Compatibility mode with version 8.3.5 is used - export is not possible. When attempting to export, an exception is generated.
  - Compatibility mode is not used - export is possible.
- Getting proxy server parameters from environment variables of Linux OS:
  - Compatibility mode with version 8.3.5 is used - environment variables are not used when creating the `InternetProxy` object. If proxy server parameters are not specified when creating the `HTTPConnection` and `FTPConnection` objects, determination of proxy server parameters from environment variables is attempted when executing object methods.
  - Compatibility mode is not used - When creating the `InternetProxy` object, `http_proxy`, `https_proxy`, `ftp_proxy`, `ftps_proxy`, `all_proxy` values of OS environment variables are used, if the `DefaultProxy` constructor parameter is set to `True` and 1C:Enterprise ca not detect the inetcfg.xml file.
    Methods of the `HTTPConnection` and `FTPConnection` objects do not attempt to define proxy server settings, but settings located in the `Proxy` property of the corresponding object are used.
- Export of spreadsheet document comments to files of Microsoft Excel 2007-2010 format:

    - Compatibility mode with version 8.3.5 is used - comments are exported as Excel table pictures.
    - Compatibility mode is not used - comments are exported as Excel table comments.
- Getting detailed record fields in aggregate functions for hierarchical groupings:
    - Compatibility mode with version 8.3.5 is used - hierarchical data set fields are used, if corresponding fields in this data set are available.
    - Compatibility mode is not used - detailed data set fields are used. To retrieve hierarchical record values inside aggregate functions, use the `CalculateExpression()` function.
- Substitution of access restriction template parameter specified in comments:
    - Compatibility mode with version 8.3.5 is used - replaced by the actual value.
    - Compatibility mode is not used - is not replaced.
- Use of conditional appearance feature of data composition system for grouping fields and detailed records:
    - Compatibility mode with version 8.3.5 is used - when styling the cell, where parent grouping field is displayed in, field values retrieved for parent grouping are used.
    - Compatibility mode is not used - when styling the cell where parent grouping field is displayed, field values retrieved for the current grouping are used.
- Loading the form table extension property for the `Period` dynamic list and the `Display` property of the form table associated with dynamic list, when opening a form:
    - Compatibility mode with version 8.3.5 is used -form table extension property for the `Period` dynamic list is loaded from user settings before event handler execution of the `OnCreateAtServer` managed form.

      The `Display` property of the form table associated with dynamic list is loaded from client settings when opening a form.
    - Compatibility mode is not used - are loaded from settings simultaneously with loading of the dynamic list user settings.
- System behavior when the `Cancel` parameter is set to `True` in event handler of the `OnOpen` managed form:
    - Compatibility mode with version 8.3.5 is used - the `OnClose` event handler and the notification specified in the `NotifyOnCloseDescription` property of managed form:
        - Called in thin client;
        - Not called in web client.

    - Compatibility mode is not used - the `OnClose` event handler is not called as well as the notification set in the `NotifyOnCloseDescription` property of managed form.
- Choice history on input for enumerations:
    - Compatibility mode with version 8.3.5 is used - choice history is not displayed and not saved.
    - Compatibility mode is not used - displaying and saving of choice history is managed by the `ChoiceHistoryOnInput` property.
- Calling (in the built-in mechanisms of the platform) to properties of a method written in 1C:Enterprise language, in case object's properties components, passed as a parameter of this method, change:
    - Compatibility mode with version 8.3.5 is used - call is performed using the property number, defined at the beginning of the call. If property number changes - the call will not be executed correctly.
    - Compatibility mode is not used - call is made using the property name. If property name changes - the call will not be executed correctly.
- The behavior of 1C:Enterprise in managed form event handler `OnOpen` if the `Close()` method is called for the same form:
    - Compatibility mode with version 8.3.5 is used -in web client, the event handler and notification set in the `NotifyOnCloseDescription` property will not be called. In thin client, event handler and notification will be called.
    - Compatibility mode is not used - event handler and notification set in `NotifyOnCloseDescription` property will not be called in any client application.

7.2.3. With 8.3.6 version

- Behavior of `Get()` method for the object of `ValueStorage` type:
    - Compatibility mode with version 8.3.6 is used - using this method in thin client does not generate an exception. In this case, the `undefined` value returns. Using this method in a web client generates an exception.
    - Compatibility mode is not used -using this method in thin client generates an exception.
- Behavior of `ValueType()` query language function with an `Undefined` parameter type:
    - Compatibility mode with version 8.3.6 is used - returns the `NULL`.
    - Compatibility mode is not used - returns the `Undefined` value.
- Behavior of the `AccessRight()` function to check the rights for the log column:
    - Compatibility mode with version 8.3.6 is used - the rights for attributes included in log columns components are not checked. Column has always been considered available.

- Compatibility mode is not used - column access right is "added up" from access rights for included attributes. Graph is considered unavailable if at least one attribute included in the column components is unavailable.

- Determining the first day of the week:
  - Compatibility mode with version 8.3.6 is used - always Monday.
  - Compatibility mode is not used - is set using the First day of the week property in regional infobase settings dialog box. The first day of the week is considered when operating with various mechanisms of 1C:Enterprise (including the query language, data composition system, calendar presentation, etc).
- Components of selection history list:
  - Compatibility mode with version 8.3.6 is used - five of the last selected values.
  - Compatibility mode is not used- no more than 7 values, filtered using an appropriate algorithm.
- Object value type, storing the choice history:
  - Compatibility mode with version 8.3.6 is used - `Array` object.
  - Compatibility mode is not used- `ChoiceHistorySettings` object.
- Property `template`object`SpreadsheetDocument`:
  - Compatibility mode with version 8.3.6 is used -spreadsheet document with `Template` property set is edited in managed form. always in current language.
  - Compatibility mode is not used- spreadsheet document with `Template` property set is edited in managed form in several languages. Language components, that are used for spreadsheet document editing, are acquired from the spreadsheet document itself.
- Spreadsheet document display language:
  - Compatibility mode with version 8.3.6 is used - in thin client, spreadsheet document is always displayed and edited in current language; in web client, the `LanguageCode` property is used. When editing a spreadsheet document, only the line in current language is set, lines in other languages are lost.
  - Compatibility mode is not used - spreadsheet document is displayed and edited in the language set using the `LanguageCode` property. If property value is not set - display and editing is performed in the current language. Editing only changes the lines in the specified language, lines in other languages are not changed.
- Export of overall totals in columns of the data composition system table:
  - Compatibility mode with version 8.3.6 is used - in the header of data composition system table, in totals column, the titles of the resource fields located in groups were not displayed, if table columns are set to grouping templates.

    - Compatibility mode is not used - In overall total columns of data composition system, resource fields headers located in groups are displayed, if grouping templates are set in table columns.
- Automatic filling of available filter fields if there is nested query in data composition schema set:
    - Compatibility mode with version 8.3.6 is used - nested query fields are unavailable for setting a filter (besides virtual tables parameters).
    - Compatibility mode is not used - fields from the nested query selection list becomes available for filter, besides fields that are aggregate function parameters in parent report.
- Type of data set connection in data composition system:
    - Compatibility mode with version 8.3.6 is used - replaced by internal when using it in data entry fields filter, that are connected.
    - Compatibility mode is not used - replaced by internal, if the filter containing only set fields data that user is connected with is used. Conditions that have no internal connection are added to the `RecordsFilter` grouping property containing at least one data set, fields of which are used in conditions.
- System behavior if object (document, catalog item, chart of characteristic types, business process or task) number was allocated to transaction and object was not saved in tis transaction:
    - Compatibility mode with version 8.3.6 is used - allocated number is not reused if autonumbering mode is set to Do not release automatically.
    - Compatibility mode is not used - will be reused even if autonumbering mode is set to Do not release automatically, including a situation when object was not saved in transaction.
      Autonumbering mode Do not release automatically affects only numbers assigned outside of the transaction. I.e., if a number was assigned outside of the transaction and a code was not used (object was not saved), a number is not reused in the Do not release automatically mode, but a number is reused in the Release automatically mode.
- The option to place the `NULL` value in external data source fields (in terms of data base scheme):
    - Compatibility mode with version 8.3.6 is used - the `NULL` value can be placed in all external data source fields.
    - Compatibility mode is not used - the `NULL` value can be placed in table fields (and cube dimensions tables) of external data sources only if the Allow NULL check box is set in metadata field. The `NULL` value can be placed in all cube table fields of external data sources.

7.2.4. In version 8.3.7

- Total ordering if ordering is specified in data set field of data composition scheme and in data composition settings:
  - Compatibility mode with version 8.3.7 is used - if ordering expression is specified in data set field of data composition scheme, and ordering by this field with the Descending ordering direction is specified in data composition settings, then the ordering direction specified in the ordering expression for this field is replaced with the opposite one.
  - Compatibility mode is not used - if ordering expressions are specified in data set field of data composition scheme, then the ordering direction is always retrieved from the ordering expression when ordering by this field. Ordering direction from this field settings is ignored.
- Use of the cacert.pem file to check server certificates when creating HTTPS/FTPS connection:
  - Compatibility mode with version 8.3.7 is used - the cacert.pem file is used.
  - Compatibility mode is not used - the cacert.pem file is not used. Root certificates list of the used operating system is used.
- Extensions attached when performing a background job:
  - Compatibility mode with version 8.3.7 is used - all extensions registered for the current infobase are attached.
  - Compatibility mode is not used - the same extensions as in the session, where background job starts from, are attached.
- `Behavior of the` RepresentableDocumentBatch.Print() method`:`
  - Compatibility mode with version 8.3.7 is used - each document is printed in a separate job.
  - Compatibility mode is not used:
    - On Windows OS:
      - one job for one printer is created;
      - the whole batch is output to one file with it's name asked before output.
    - On Linux:
      - if print parameters of all batch documents are the same - behavior is the same as the behavior on Windows OS;
      - if parameters are different - each document is printed in a separate printer job.
- Maximum size of data, indexed by full-text search, before the first call of the `SetMaxIndexedDataSize()` method:
  - Compatibility mode with version 8.3.7 is used - no limit is set.
  - Compatibility mode is not used - 1 MB.
- Use of the Mark negatives conditional appearance property of data composition system:

  - Compatibility mode with version 8.3.7 is used - used only if the Usage property is set to `True`.
  - Compatibility mode is not used - used when the property itself is set to `True` and Usage property is set to `True` too.
- Calculation of the `Selection` query language construct:
  - Compatibility mode with version 8.3.7 is used - the `THEN` branches, containing constant expression with the `NULL` value, are ignored in calculation.
  - Compatibility mode is not used - types of expressions from all the `THEN` branches are used in type calculation.
- Type of division operation result (in the query language), where the `NULL` is a divisor:
  - Compatibility mode with version 8.3.7 is used - not interpreted as a number and can serve as operand in operation, where the number value is not allowed (for example, the `And`, `Or`, `Not` operations).
  - Compatibility mode is not used - interpreted as a number of arbitrary bitness. Such operation result can not be interpreted as operand in an operation, where number value is not allowed (for example, the `And`, `Or`, `Not` operations).
- Conversion of characters missing in the current font when exporting a spreadsheet document to PDF format:
  - Compatibility mode with version 8.3.7 is used - characters missing in the current font are displayed as lost characters («square»).
  - Compatibility mode is not used - font matching is performed for characters that are not in the current font.
- Type of query field and type of query parameter in data composition scheme (in the `SELECT &Parameter AS a Field` query):
  - Compatibility mode with version 8.3.7 is used - the type specified for parameter of data composition scheme does not affect query field type, where this parameter is used.
  - Compatibility mode is not used - type specified for parameter of data composition scheme affects query field type, where this parameter is used.
- Use of the Text and Format conditional appearance items in chart template field of data composition system:
  - Compatibility mode with version 8.3.7 is used - only if the Format conditional appearance was not applied to this field before.
  - Compatibility mode is not used - regardless of the Format appearance previously set (including the one taken from data composition scheme). The Text conditional appearance item has a greater priority than the Format conditional appearance item.

- Use of system enumerations field (links to the table of external data source or UUID) in the `Group By` query operation, when retrieved value is not combined with other types:
  - Compatibility mode with version 8.3.7 is used - in some cases, grouping by such field may not be performed.
  - Compatibility mode is not used - grouping by such field is always performed.
- Availability on the client side (using the `CurrentData` table property and the `RowData()` method) of dynamic list data for which the current user has no `View` right and the AlwaysUse check box is set:
  - Compatibility mode with version 8.3.7 is used - data is available and filled with default values of corresponding types.
  - Compatibility mode is not used - data is missing and access to it is unable.
- Dynamic list operation when the current user has no `View` right for key field of the dynamic list:
  - Compatibility mode with version 8.3.7 is used - in most cases operates successfully.
  - Compatibility mode is not used - causes access rights violation error.
- Availability in the source of available data composition settings for a dynamic list for selecting, sorting and grouping the following fields: child fields of fields that are aggregate functions of query language; child fields of fields from other (external) data sources; child fields-characteristic of tabular section fields.
  - Compatibility mode with version 8.3.7 is used - available. When setting a filter on such fields, dynamic list can operate incorrectly: no displaying all data or its part.
  - Compatibility mode is not used - unavailable.
- Searching using the search strings for reference fields, which are aggregate functions of the query language:
  - Compatibility mode with version 8.3.7 is used - search attempted to be performed. While doing so, errors could be issued, and in case errors were not issued - the dynamic list could operate incorrectly: no displaying all data or its part.
  - Compatibility mode is not used - not performed.
- Availability of filtering fields of the main table (and their child fields) in a dynamic list with a given arbitrary query and specified main table:
  - Compatibility mode with version 8.3.7 is used - only the fields present in query text and standard fields (`Reference`, etc.) of the main table were available for filtering. If unavailable fields with names that match the names of the fields from the main table are placed in filter - such filtering could be applied successfully.

- Compatibility mode is not used - does not depend on the presence of these fields in text of the dynamic list request. Fields of the main table are implicitly added to the query text if they are missing from the query text.
  If the query text, set for dynamic list, already has a field, whose alias matches the alias of the field of the main table - this field will be used for filtering, regardless of whether it really is that field of the main table. Fields that have several path options (using different options of the embedded language) are available only on the current option of the embedded language.

7.2.5. With version 8.3.8

- Path separators for `Path` and `FullNameOfTheFile` Object `ZipFileEntry`:
  - Compatibility mode with version 8.3.8 is used - "/" is always used as path separator.
  - Compatibility mode is not used - separator depends on the operating system used.
- Behavior of the `Send()` method of an `InternetMail` object in case of errors in sending e-mails via SMTP protocol:
  - Compatibility mode with version 8.3.8 is used - an exception is raised.
  - Compatibility mode is not used - `Send()` method returns the object of `Correspondence` type, which describes errors for each mailing address for which the delivery failed.
- Behavior of the `GetURL()` method if a value is passed as the value of the `Object` parameter for which the URL cannot be obtained:
  - Compatibility mode with version 8.3.8 is used - returns a blank string.
  - Compatibility mode is not used - an exception is generated.
- Temporary tables in temporary table manager after transaction rollback:
  - Compatibility mode with version 8.3.8 is used - all temporary tables created within transaction are removed. Temporary tables created by this query within transaction can not be used in file mode query after the transaction rollback. In operating with other database management systems, the use of temporary tables after transaction rollback is unavailable.
  - Compatibility mode is not used - temporary tables are not removed automatically. Temporary tables created within transaction can be used in a query of file mode database after transaction rollback. When operating with other DBMS, the use of temporary tables after transaction rollback is unavailable.
- Behavior of various DBMS options when passing a values table, in which the `String` type of more than 1024 characters length or the

BinaryData of more than 2048 bytes size is specified for a column, as the In operation parameter of query language:
  - Compatibility mode with version 8.3.8 is used - the capability to execute such query depends on the used DBMS.
  - Compatibility mode is not used - when executing this query, an exception is generated.
- Behavior of various DBMS options when passing a values table, in which the String type of more than 1024 characters length or the BinaryData of more than 2048 bytes size is specified for a column, as the In operation parameter of query language:
  - Compatibility mode with version 8.3.8 is used - the capability to execute such query depends on the used DBMS.
  - Compatibility mode is not used - when executing this query, an exception is generated.
- Aliases generation in query wizard (in Designer and thick client) and in the QuerySchema object (in thick client):
  - Compatibility mode with version 8.3.8 is used - aliases are generated only for field expressions and fields used for ordering and totals calculation.
  - Compatibility mode is not used - aliases are generated for all first query fields (among merged) for each batch query.
- Parameter name generation by data composition template composer:
  - Compatibility mode with version 8.3.8 is used - can assign names existing in data composition scheme to parameters.
  - Compatibility mode is not used - does not assign names existing in data composition scheme to parameters.
- Setting visibility and group name, when automatically grouping strings/columns in a spreadsheet document:
  - Compatibility mode with version 8.3.8 is used - visibility and group name from area displayed within grouping is provided.
  - Compatibility mode is not used -visibility and group name from area displayed within grouping header area is provided.
- Default value of the DuplexPrinting property for spreadsheet document, graphical schema, representable document batch, text document:
  - Compatibility mode with version 8.3.8 is used - the DontUse value.
  - Compatibility mode is not used or compatibility mode with version 8.2.16 and earlier is used - the UsePrinterSettings property.
- Recording or reading an HTML document containing the <?xml string:
  - Compatibility mode with version 8.3.8 is used - causes an error.
  - Compatibility mode is not used - is executed correctly.
- Generation of parameters in data composition scheme template, for parameters which names can be in Russian and English:

    - Compatibility mode with version 8.3.8 is used - parameter is always generated with a name in English.
    - Compatibility mode is not used - names are generated in the same language as the 1C:Enterprise language option.
- Behavior when executing the following scenario: any text is entered in the input field and a hyperlink located in decoration is clicked without closing the input field, context server call is performed in hyperlink decoration clicking handler:
    - Compatibility mode with version 8.3.8 is used:
        - in thin and thick client applications:
            - for decoration the `URLProcessing` event handler is called;
            - the input field displays the value that was present before text entering.
        - in web client:
            - for decoration the `URLProcessing` event handler is called;
            - the input field displays entered text.
    - Compatibility mode is not used:
        - for the input field the `OnChange` event handler is called;
        - for decoration the `URLProcessing` event handler is called;
        - the input field displays a value matching the entered text.
- Performing template composition of data composition system, retrieving fields and using filters to fields retrieved by a dot from fields containing aggregate functions that are retrieved from query:
    - Compatibility mode with version 8.3.8 is used - performed in the query itself. An incorrect result can be retrieved or the generated query can cause error while executing.
    - Compatibility mode is not used - executed using additional data sets, automatically generated by the template composer.
- Export of a spreadsheet document to the XLS format (Microsoft Excel 97), Microsoft Office OpenXML (Microsoft Excel 2007 - 2010) containing cells with multiline text that is aligned to the cell center and does not fit in the cell entirely:
    - Compatibility mode with version 8.3.8 is used - additional cell merges are not created.
    - Compatibility mode is not used - additional cell merges are created. Merging is performed horizontally, to both sides of the original cell, on equal distances sufficient to place all the text of the original cell. Columns containing left and right borders of a rectangle filled with text are split in two.
- Export of a spreadsheet document, containing pictures located in cells, to Microsoft Office OpenXML (Microsoft Word 2007 and earlier) format:
    - Compatibility mode with version 8.3.8 is used - pictures in the spreadsheet document cells are not exported.

- Compatibility mode is not used - pictures in the spreadsheet document cells are exported. There are the following limitations:
  - Only horizontal direction of text is supported in picture cells.
  - If text direction is different from horizontal in such spreadsheet document cell, it is displayed horizontally on export.

7.2.6. With version 8.3.9

- Automatic filling of available fields in a dynamic list:
  - Compatibility mode with version 8.3.7 is used - automatic filling of available fields is enabled for any query.
  - Compatibility mode is not used - automatic filling of available fields can be disabled using the `AutoFillAvailableFields` property.
- Usability of SystemFields folder contents as selection fields in a dynamic list:
  - Compatibility mode with version 8.3.9 is used - folder contents can be used as selection fields.
  - Compatibility mode is not used - unavailable for using as selection fields.
- Grouping fields setting of a dynamic list the main table for which is not specified or key field that can use a dynamic list is missing in main table:
  - Compatibility mode with version 8.3.9 is used - supported, but causes errors when retrieving data.
  - Compatibility mode is not used - not supported.
- Availability of the DynamicDataRead, MainTable, CustomQuery and QueryText properties for the DynamicList object in thick client:
  - Compatibility mode with version 8.3.9 is used - available in thick client. But changing them does not cause any changes in dynamic list operating.
  - Compatibility mode is not used - unavailable in thick client.
- Assigning a value to a managed form attribute of `DynamicList` type:
  - Compatibility mode with version 8.3.9 is used - assigning is supported. Further behavior is undefined.
  - Compatibility mode is not used - not supported.
- Passing the `DynamicList` type value in parameters or return value of client/server call:
  - Compatibility mode with version 8.3.9 is used - passing is available. Further behavior is undefined.
  - Compatibility mode is not used - not supported.
- Retrieving presentations of reference type values that are not displayed on a form or are associated to invisible form items:

  - Compatibility mode with version 8.3.7 is used - presentations of all reference type values that are passed on client application side are retrieved.
  - Compatibility mode is not used - capability to retrieve presentations of such fields is regulated by the `GetInvisibleFieldPresentations` property of the `DynamicList` managed form attribute.
- Import of a spreadsheet document sheet from a file of Microsoft Office OpenXML (Microsoft Excel 2007 and earlier) format with a missing name:
  - Compatibility mode with version 8.3.9 is used - such sheet is not imported.
  - Compatibility mode is not used - such sheet is imported successfully to an area with an automatically generated name.
- Conversion of cell text to a number when exporting a spreadsheet document to XLS (Microsoft Excel 97-2003) and Microsoft Office OpenXML (Microsoft Excel 2007- 2010) formats:
  - Compatibility mode with version 8.3.9 is used - cell text is converted to a number in any horizontal alignment of the cell contents.
  - Compatibility mode is not used - only if horizontal right alignment is set in the cell. Otherwise, the cell contents are saved as text.
- Name of an XML component of common string when exporting a spreadsheet document to the file of Microsoft Office OpenXML (Microsoft Excel 2007- 2010) format:
  - Compatibility mode with version 8.3.9 is used - a common string XML component is named SharedStrings.xml (begins with a capital letter).
  - Compatibility mode is not used - common string XML component is named sharedStrings.xml (begins with a lowercase letter).
- Displaying trendline in marker legend when generating a chart:
  - Compatibility mode with version 8.3.9 is used - a square marker of a color used for displaying corresponding series in a chart, is always displayed in chart legend.
  - Compatibility mode is not used - series information (a marker) in chart legend depends on how (string color, string style, a marker) this series is displayed in a chart.
- Selecting a chart series marker (the `ChartSeries.Marker` property):
  - Compatibility mode with version 8.3.9 is used - marker selection is based on a series number from the following list: `Square`, `Circle` or `Diamond`.
  - Compatibility mode is not used - the property is set to `Auto`. Marker selection is performed automatically.
- Default page size setting in document when printing or exporting to PDF format:
  - Compatibility mode with version 8.3.9 is used - the A4 format is used.

- Compatibility mode is not used - the `PrintSettings` object is used for default page size setting. In web client the Default page size setting of the parameter settings dialog box is used.

- If a series number of cryptography certificate begins with an old bit set, then:
  - Compatibility mode with version 8.3.8 is used - series number is not complemented by a leading zero byte.
  - Compatibility mode is not used - series number is complemented by a leading zero byte.
- When collapsing a node of a value tree with subordinates displayed in a managed form, the `BeforeCollapse` event:
  - Compatibility mode with version 8.3.9 is used - generated only for top-level tree items.
  - Compatibility mode is not used - generated for each collapsed node in all hierarchy.
- Calling the `AccessRight()` method if the checked access right can not be applied to metadata object:
  - Compatibility mode with version 8.3.9 is used - returns the `True` value.
  - Compatibility mode is not used - an exception is generated.
- `The` HTTPMethod parameter value `when invoking the` HTTP-Connection.CallHTTPMethod `method:`
  - Compatibility mode with version 8.3.9 is used - can have any value.
  - Compatibility mode is not used - a value must compile with RFC 7230 standard (https://tools.ietf.org/html/rfc7230).
- When operating with a query, if the field name is missing in an expression following the temporary table name:
  - Compatibility mode with version 8.3.9 is used - name of the first field in the temporary table is used as field name.
  - Compatibility mode is not used - error is generated when analyzing the query.
- Visibility of data composition scheme parameters, the types of which contain only types disabled for functional options:
  - Compatibility mode with version 8.3.9 is used - functional options do not affect the visibility of data composition parameters. When loading a report option, the usage check box selected for such data parameters is not deselected. The `DataCompositionSettingsComposer.Restore()` method does not change the `Use` property for such parameters.
  - Compatibility mode is not used - such parameters are generated with disabled visibility. As a result, these parameters can not be edited by the user, but they are still available from 1C:Enterprise language. Beside that, such parameters are not added to available fields-parameters list. When loading report option, the usage check box is cleared for such

parameters. The `DataCompositionSettingsComposer.Restore()` method disables the `Use` property for such parameters.

- Capability to set fractional values for the `Start` and `End` properties of the `GaugeChartQualityBand` object:
  - Compatibility mode with version 8.3.9 is used - properties can have only integer values. When attempting to set fractional values, the actual set value is rounded to integer.
  - Compatibility mode is not used - properties can have fractional values.
- Setting series/points colors in chart deserialization, when Series settings or Points settings properties are set to `False`:
  - Compatibility mode with version 8.3.9 is used - performed from the default palette.
  - Compatibility mode is not used - performed after loading from the color palette.
- Actions performed if the Series setting or Points setting check box values are set to `False` in manual chart editing:
  - Compatibility mode with version 8.3.9 is used - setting of default values for design parameters of series and points is not performed.
  - Compatibility mode is not used - setting of default values for design parameters of series and points is performed, the chart is redrawn.
- Series/points colors source in thin client, if the Series settings and Points settings check boxes are cleared in the chart:
  - Compatibility mode with version 8.3.9 is used - default palette.
  - Compatibility mode is not used - the set palette.
- An attempt to change any properties of series or point using 1C:Enterprise language, if the Series settings and Points settings properties are set to `False` in the chart:
  - Compatibility mode with version 8.3.9 is used - an exception is not generated. Value is set to series or point.
  - Compatibility mode is not used - an exception is generated.
- Templates of resources created at the intersection with corresponding predefined grouping headers templates:
  - Compatibility mode with version 8.3.9 is used - contain resources.
  - Compatibility mode is not used - created blank.
- All template areas text colors (besides report header) in the «Main» appearance template of data composition used in the Taxi interface:
  - Compatibility mode with version 8.3.9 is used - gray color is used as text color.
  - Compatibility mode is not used - the `Auto` value is used as text color.
- Behavior of a managed form table, if table docking area is larger than the visible table area:

    - Compatibility mode with version 8.3.9 is used - table scrolling is unavailable.
    - Compatibility mode is not used - automatic disabling of columns docking is performed.
- Template processing in spreadsheet document cell:
    - Compatibility mode with version 8.3.9 is used- when processing a template in a spreadsheet document cell, the result does not include contents of the template located between the [] section and the [escape character.
    - Compatibility mode is not used - when template processing in spreadsheet document cell, the result includes contents from the last [] section to the [escape character.
- Behavior of an input field displaying the `Number` or `String` data in selection from a list mode:
    - Compatibility mode with version 8.3.9 is used - when editing the `Number` value, number qualifiers set for an attribute continue to affect edited text format.
    - Compatibility mode is not used - limit for the entered string length is removed, and input field displaying such data is automatically stretched by width.
- Text color in spreadsheet document when displaying in the Taxi interface:
    - Compatibility mode with version 8.3.9 is used - black color is always used as the Auto text color (both for displaying and for printing).
    - Compatibility mode is not used - the Auto text color is:
        - when displaying: style color of editing text;
        - when printing: black color.
- Windows OS presentation in the `SystemInfo.OSVersion` property from Vista version and later:
    - Compatibility mode with version 8.3.9 is used - property text does not contain the user presentation of OS name.
    - Compatibility mode is not used - property text contains user presentation of OS name in addition to technical information (version, build number etc.).
- Verification of compliance of a subscription event and subscription source type:
    - Compatibility mode with version 8.3.9 is used - verification of compliance of an event and source type is not performed for event subscriptions.
    - Compatibility mode is not used - verification is performed before saving configuration. If an event does not comply the source type - an error is generated and configuration is not saved.
- Adding a filter condition to a query using the QuerySchemaExpression object wizard:

    - Compatibility mode with version 8.3.9 is used - filter expression is added incorrectly.
    - Compatibility mode is not used - filter expression is added with no conversions.
- System behavior when double clicking the managed form table cell, if a hyperlink is displayed in this cell and the OnlyView mode is set for this cell:
    - Compatibility mode with version 8.3.9 is used - generation of two `Selection` table events.
    - Compatibility mode is not used - generation of one `Selection` table event.
- Use of dimension fields data composition system in virtual table parameters of information register when automatically filling the available data-query set fields:
    - Compatibility mode with version 8.3.9 is used - dimension fields are unavailable for filtering by virtual table parameters.
    - Compatibility mode is not used - dimension fields are available for filtering by virtual table parameters.
- Location of commands for navigation to associated objects for item forms and groups of hierarchical catalogs and charts of characteristic types:
    - Compatibility mode with version 8.3.9 is used - all commands for navigation to subordinate catalogs are located in the navigation panel of item form. Groups of commands for navigation to subordinate catalogs are not placed in form navigation panel.
      For a hierarchical catalog and a chart of characteristic types, all commands for navigation to information registers are located in the navigation panel of item form. In form navigation panel, groups of commands for navigation to information registers are not located.
    - Compatibility mode is not used - in the navigation panel, item forms of hierarchical catalog or chart of characteristic types are located:
        - Commands for navigation to subordinate catalogs for which items or items and groups subordination is set.
        - Commands of navigation to information registers, a leading dimension type of which is the same as a type of a catalog or a chart of characteristic types and where the Selection of groups and items property is set to Items or Groups and items for this dimension.

      In navigation panel, forms of a group of hierarchical catalog or chart of characteristic types are located:
        - Commands for navigation to subordinate catalogs for which items or items and groups subordination is set.
        - Commands of navigation to information registers, a leading dimension type of which is the same as a type of a catalog or a chart of

characteristic types and where the Selection of groups and items property is set to Groups or Groups and items for this dimension.

- Enumerations in a dialog box type:
  - Compatibility mode with version 8.3.9 is used - in thin and thick client applications operating in managed application mode, a type selection dialog box and a type description dialog box do not contain enumerations. Enumerations are available in these dialog boxes in web client.
  - Compatibility mode is not used - enumerations are available in all client applications operating in managed application mode, in a type selection dialog box and in a type description dialog box.
- System behavior when attempting to create a managed form group from 1C:Enterprise language with incorrect value of the `View` property:
  - Compatibility mode with version 8.3.9 is used - system behavior is undefined (including a chance of crash).
  - Compatibility mode is not used - an exception is generated.
- Controlling the height of a managed form table:
  - Compatibility mode with version 8.3.9 is used - only the table height can be specified in conditional units. An more accurate control of the form table height is available only in the mobile platform.
  - Compatibility mode is not used - controlling the height of a managed form table that displays data different from dynamic list is available. The `ItemHeightControlVariant` property is used.
- Form behavior if the `Cancel` parameter is set to `True` in the `OnOpen` event handler:
  - Compatibility mode with version 8.3.9 is used - in some scenarios, a form cannot be exported from a memory after handler operation completion. As a result, all resources (temporary storage, locks etc.) associated with it are saved, and set idle handlers operated.
  - Compatibility mode is not used - after handler execution, the form is exported from memory, all associated with it resources (temporary storage, locks etc.) are released, and set idle handlers stop operating.
- Merged cells outputting to spreadsheet document by the processor of data composition result output:
  - Compatibility mode with version 8.3.9 is used - when outputting merged cells, the processor of data composition result output to spreadsheet document in some cases does not change columns width where merged cells are located. As a result, columns can be too small.
  - Compatibility mode is not used - when outputting merged cells, the processor of data composition result output to spreadsheet document changes columns width where merged cells are located in such way that these cells contents take minimum place in height.
- Displaying a managed form table, when the `FixingInTable` property is set to `Left` or `Right` in a table columns group:

- Compatibility mode with version 8.3.9 is used - only the docked columns that are located to the left (for the `Left`) and to the right (for `Right`) of such a group. Columns included in the group are not docked.
- Compatibility mode is not used - all columns in the group are docked as well as all other columns and table groups where docking is specified.

- The list of formal parameters of the `DetailProcessing` and the `AdditionalDetailProcessing` spreadsheet document events handlers:
  - Compatibility mode with version 8.3.9 is used - event handlers have two parameters: `Details`, `StandardProcessing`. If the `AdditionalParameters` parameter is added to the event handler procedure, the value of this parameter will be `Undefined`.
  - Compatibility mode is not used - for these handlers, the `AdditionalParameters` parameter is implemented. If the event is generated by a Gantt chart, then in this parameter the date on which the decryption is initiated is transmitted. In other cases, a blank structure will be transmitted.
- The `Use` property state for data parameters, disabled by functional options, after loading the report option or after the `DataCompositionSettingsComposer.Restore()` method calling:
  - Compatibility mode with version 8.3.9 is used - the property is always disabled.
  - Compatibility mode is not used - the property is disabled only if the parameter can be edited by user.

7.2.7. With version 8.3.10

- `Setting the` LineSplitter `and the` ConvertibleSplitterOfLines `parameter values of the` TextReader object constructor`:`
  - Compatibility mode with version 8.3.10 is used - if parameter values are not set, a blank string is used as a value. If a type or a value of the passed parameter are invalid, an exception is generated.
  - Compatibility mode is not used - if parameter values are not specified, default values are used:
    - LineSplitter `-` LF`,`
    - ConvertibleSplitterOfLines `-` CR + LF`.`

    If a type or a value of the passed parameter are invalid, an exception is generated.
- Usage of the `Move()` method when operating with an object model of a query scheme:
  - Compatibility mode with version 8.3.10 is used:

    - When performing the method for the `QuerySchemaQueryBatch` and the `QuerySchemaOperators` objects, the maximum one of the `<Index>` and the `<New index>` parameters is selected, and the item with the largest index moves to the position marked by another index.
    - A collection item moves to a specified position when performing this method for following objects: `QuerySchemaColumns`, `QuerySchemaIndexExpressions`, `QuerySchemaOrderExpressions`, `QuerySchemaTotalCalculationFields`.
  - Compatibility mode is not used - performing this method in objects listed below moves a collection item to a specified number of positions: `QuerySchemaQueryBatch`, `QuerySchemaOperators`, `QuerySchemaColumns`, `QuerySchemaIndexExpressions`, `QuerySchemaOrderExpressions`, `QuerySchemaTotalCalculationFields`.
- In mobile platform, when an error is caused while sending notifications associated with access marker obsolescence:
  - Compatibility mode with version 8.3.10 is used - an exception is generated.
  - Compatibility mode is not used - error handling must be performed in the configuration code.
- Chart palette used by default:
  - Compatibility mode with version 8.3.10 is used - the `Palette32` palatte.
  - Compatibility mode is not used - the `SoftAdaptive` palette.
- The `ChartPlotArea` object property values:
  - Compatibility mode with version 8.3.10 is used - the `ScaleColor.Auto` value is interpreted as `0xA9A9A9`. The `BackColor.Auto` value is interpreted as `0xFFFFFF`.
  - Compatibility mode is not used - the `ScaleColor.Auto` value is interpreted as `0xDDDDDD`. The `BackColor.Auto` value is interpreted as the `EditBackground` color.
- If the `ScaleLineColor` property of the `ChartScale` object is set to `Auto` and the `ScaleColor` property of the `ChartPlotArea` object is equal to `Auto`, then the following is used for scale string drawing:
  - Compatibility mode with version 8.3.10 is used - the `0xA9A9A9`color.
  - Compatibility mode is not used - the `0x7D7D7D` color.
- When performing a query containing the `MAXIMUM()` and the `MINIMUM()` functions where a parameter is set to `UUID` or where system

enumeration with each other combination or with primitive type values is retrieved:

  - Compatibility mode with version 8.3.10 is used - an error is not diagnosed. Functions result can be incorrect, or performance of a query can cause platform crash.
  - Compatibility mode is not used - an exception is generated.

- When performing queries containing the `ISNULL` or the `SELECTION` operations applied to the `MAXIMUM()` and the `MINIMUM()` functions applied to the `UUID` values or system enumeration, if primitive type expression or another system enumeration is used in another operand of the `ISNULL` or `SELECTION` operation:
  - Compatibility mode with version 8.3.10 is used - returns `Undefined`.
  - Compatibility mode is not used - a correct result is displayed.
- Definition of parameter types used in a query of the addition operation and the `EQUAL`, `NOT EQUAL`, `LESS`, `GREATER`, `LESS OR EQUAL`, `GREATER OR EQUAL` comparison operations:
  - Compatibility mode with version 8.3.10 is used - a parameter type is a type of another operation operand including qualifiers.
  - Compatibility mode is not used - type qualifiers are unlimited. For example, if `Number` is a type, a number without value size limit is used, if `String` - a string without size limit.
- Use of the `Parameters.SearchString` parameter in standard processing of the `ChoiceDataGetProcessing` configuration object manager event:
  - Compatibility mode with version 8.3.10 is used - if the `Parameters.SearchString` parameter value is set to `Undefined`, it is the same as specifying a blank string as a search string.
  - Compatibility mode is not used - if the `Parameters.SearchString` parameter value is set to `Undefined`, system behavior is the same as quick choice and returns a list of all items that are in database for this configuration object.
- Displaying of folder pictures in strings of a table that displays an hierarchical list:
  - Compatibility mode with version 8.3.10 is used - displaying of folder pictures in strings for hierarchical dynamic lists can not be disabled. Clearing the `StringPictureDataPath` property of a table displaying a hierarchical dynamic list does not disable folder picture displaying.
  - Compatibility mode is not used - a picture set in the `RowPictureDataPath` property is displayed as a folder picture. If this property is cleared - a folder picture is not displayed.

- Verification of a user command name for matching with a standard command name in verification of configuration logical integrity:
  - Compatibility mode with version 8.3.10 is used - for commands with names that failed verification, a warning that a command name is the same as a standard command name is generated. Infobase can be updated.
  - Compatibility mode is not used - for commands with names that failed verification, an error with information saying that a command name is the same as a standard command name is generated. Infobase cannot be updated.
- Verification of the right to access processed objects during data full-text search in the enabled privileged mode:
  - Compatibility mode with version 8.3.10 is used - verification of the access rights is performed.
    The `FullTextSearchList.GetPresentation` property does not affect a search result.
    Verification of the right to access object fields that are used in object description generation, if the `FullTextSearchList.GetDescription` property is set to `False`, is not performed.
  - Compatibility mode is not used - verification of the access rights is not performed.
    Verification of the rights to access object fields that are used in presentation generation is always performed.
    Verification of the rights to access object fields that are used in object description generation is always performed.
- Retrieving metadata for the `InformationRegisterRecordKey` objects:
  - Compatibility mode with version 8.3.10 is used - this feature is not provided.
  - Compatibility mode is not used - the `Metadata()` method is available.
- Import of the default and the fixed properties for links to global items and attributes when importing the XSD schema to the XDTO batch:
  - Compatibility mode with version 8.3.9 is used - import of the default and the fixed properties that are specified locally is not performed.
  - Compatibility mode is not used - the default and the fixed properties are interpreted regardless of a type of a corresponding item declaration or an attribute. The default and the fixed properties are imported to the By default and the Fixed properties respectively.
- The language of a generated name of a spreadsheet document area:
  - Compatibility mode with version 8.3.10 is used - a name is generated in the current interface language.

    - Compatibility mode is not used - a name is generated in the same language as selected in 1C:Enterprise language option.
- Behavior when assigning a variable value with a common module name on client application side when operating in thin client and web client:
    - Compatibility mode with version 8.3.10 is used - does not operate and is not diagnosed by a platform. When operating in web client, client application crashes.
    - Compatibility mode is not used - does not operate and an error message is generated.
- Use of the `PredefinedDataName` and the `DataVersion` fields in a dynamic list:
    - Compatibility mode with version 8.3.10 is used:
        - All comparison types are available.
        - When searching using a search by these fields string, an error is generated.
        - An advanced search allowed the user to specify any search option, but when searching by occurrence or by string beginning, an error was generated.
    - Compatibility mode is not used:
        - Only the Equal, Not equal, in list, Not in list, Filled, Not filled types of comparison are available.
        - Searching using a search string by these fields is not supported.
        - Advanced search can be used by an exact match only.
- The `LabelAngle`, `ScaleLocation`, `MaximumNumberOfLabelRows`, `ScaleMarkLocation`, ScaleLabelLocation charts scale properties:
    - Compatibility mode with version 8.3.10 is used - not supported and not displayed in a chart properties palette.
    - Compatibility mode is not used - available. Displaying of charts, scales, scale labels and data table drawing can be different from displaying in the enabled compatibility mode.
- The `ChartPlotArea` object properties:
    - Compatibility mode with version 8.3.10 is used - properties are used to chart: `DisplaySeriesScaleLabels`, `DisplayPointsScaleLabels`, `DisplayValuesScaleLabels`, `DisplayScaleValueStrings`, `ValueScaleFormat`, `LabelsOrientation`.
    - Compatibility mode is not used - the following properties are expired and not recommended to use:
        - `DisplaySeriesScaleLabels`. To access, it is recommended to use `SeriesScale.ScaleLabelLocation`.

    - `DisplayPointsScaleLabels`. To access, it is recommended to use `PointsScale.ScaleLabelLocation`.
    - `DisplayValuesScaleLabels`. To access, it is recommended to use `ValuesScale.ScaleLabelLocation`.
    - `DisplayScaleValuesStrings`. To access, it is recommended to use `ValuesScale.GridLinesShowMode`.
    - `ValueScaleFormat`. To access, it is recommended to use `ValuesScale.LabelFormat`.
    - `LabelsOrientation`. To access, it is recommended to use `PointsScale.LabelOrientation`.
- The order of iteration of managed form group items, where the Merged property is set to False, in thin client.
  - Compatibility mode with version 8.3.10 is used - this group was ignored and did not affect the iteration order.
  - Compatibility mode is not used - the group is not ignored and group items are iterated according to group settings (vertical and horizontal).
- `Support of the following properties in the` ChartAxis stock chart object: `MinValueDetectionMethod`, `MinValue`, `MaxValueDetectionMethod`, `MaxValue`:
  - Compatibility mode with version 8.3.10 is used - properties are not supported.
  - Compatibility mode is not used - properties are supported.
- Behavior of a data composition system when using nested queries of the `Not filled` comparison type, that are applied to fields from a nested query of the right part of the left connection, in a query:
  - Compatibility mode with version 8.3.10 is used - a condition is always placed to a nested query, if it is available.
  - Compatibility mode is not used - a condition is placed to nested queries located in the right part of left connection only if this condition cannot be placed to a nested query. Otherwise, a condition is not placed to a nested query.
- Behavior of the query language, if in a `SELECTION` construct, another `SELECTION` construct or the `HASNULL` function is used, and a virtual field and not a virtual field is used in them:
  - Compatibility mode with version 8.3.10 is used - a calculation result can be incorrect or program can crash.
  - Compatibility mode is not used - a calculation result is correct.
- When using prohibited characters in the `<CatalogName>` parameter of the `CreateDirectory()` and the `BeginCreatingDirectory` methods:
  - Compatibility mode with version 8.3.10 is used - a directory is not created. An exception will not be generated.

    - Compatibility mode is not used - an exception will be generated.
- A location relative to a parent group of another managed form group, for which the `HorizontalAlignInGroup` property is set to the `Center` value:
    - Compatibility mode with version 8.3.10 is used - location is shifted to the left relative to a parent group.
    - Compatibility mode is not used - location is in the center relative to a parent group.
- Visibility of a managed form items associated with commands of table extensions or a form for a dynamic list that user has no right to access:
    - Compatibility mode with version 8.3.10 is used - the following commands are not hidden, if they were added by a developer:
        - Change - if a filter criterion is the main table of a dynamic list.
        - Remove - if an external data source table is the main table of a dynamic list.
        - The Display list, Load dynamic list settings, Save dynamic list settings, commands for any dynamic list.

      Automatically placed commands are not displayed in a form.
    - Compatibility mode is not used - commands that are unavailable for the user are not displayed regardless of how they are placed in the form.
- Form table commands depending on different parameters:
    - Compatibility mode with version 8.3.10 is used:
        - If the main table of the dynamic list is the selection criterion or enumeration -no advanced search commands or list settings control are added.
        - The elements connected with the table or form extension commands for the dynamic list, which are inaccessible due to the table bound properties, are deleted if they are added automatically (by the table or form extensions) and are not deleted if they are added by the application developer. In this case, the commands left (in the second case) execute no actions.
        - If in the process of form creation, it becomes clear that a table displayed in the form is inaccessible then the better part of commands for this table is deleted from the form. Only one command (as a rule it is Update command) remains in the form.
    - Compatibility mode is not used:
        - If the main table of the dynamic list is the selection criterion or enumeration - the advanced search commands and list settings control are added.
        - The elements connected with the table or form extension commands for the dynamic list, which are inaccessible due to the table bound properties, are deleted in any case.

    - If in the process of form creation, it becomes clear that a table displayed in the form is inaccessible then all commands for this table are deleted from the form.
- Placing tab characters (separating table columns) in a text document when displaying a managed form table using the command Export list:
  - Compatibility mode with version 8.3.10 is used - inserted after the first not blank column value.
  - Compatibility mode is not used - inserted between all columns.
- Displaying the `Date` and the `Number` blank strings when displaying a managed form table using the Output list command:
  - Compatibility mode with version 8.3.10 is used - can be displayed as a type different from the type displayed on the form.
  - Compatibility mode is not used - displayed as the same type as the type displayed on the form.
- Displaying the background of a managed form button and of a command bar button where the `ShapeRepresentation` property is set to `No` and the `BackColor` property is different from `Auto`:
  - Compatibility mode with version 8.3.10 is used - background is displayed in the specified color.
  - Compatibility mode is not used -background is not displayed.
- System behavior when accessing a file name in a ZIP archive (using the `ZipFileReader` object), if this file name contains the prohibited characters ((>, <, |, ?, *, /,\, :,"):
  - Compatibility mode with version 8.3.10 is used - a file name is cut to the first prohibiter character.
  - Compatibility mode is not used - the prohibited characters in a file name will be replaced by underscore «_».
- A location of nested tables and charts in a resulting spreadsheet document when data composition system is operating:
  - Compatibility mode with version 8.3.10 is used - nested charts grouping and table grouping is not applied. A template generator does not create blank templates with a set `VerticalLevel` property that are located after chart templates. For tables nested in groupings, a blank template is created and the `VerticalLevel` property is set to 0 for it.
  - Compatibility mode is not used - placed in spreadsheet document groupings. Generator of data composition system templates creates blank templates with a set `VerticalLevel` property, that are located after templates of charts and tables nested in groupings.
- Result of operations of`ThisNode()` exchange plan method called for an exchange plan item where `ThisNode` property is set to `True`:
  - Compatibility mode with version 8.3.10 is used - not always returns a link to this item.
  - Compatibility mode is not used - always returns a link to this item.

- Displaying selected table strings (using the DisplayList command) in a spreadsheet or a text document:
  - Compatibility mode with version 8.3.10 is used:
    - strings are displayed in the order of selection;
    - not all strings are displayed, if a table displays a dynamic list in the Hierarchical list or the Tree mode.
  - Compatibility mode is not used:
    - strings are displayed in the same order as they are located in a table;
    - all strings are displayed regardless of a hierarchy level, if a table displays a dynamic list in the Hierarchical list or the Tree mode.

7.2.8. With version 8.3.11

- A pop-up group in the managed form:
  - Compatibility mode with version 8.3.11 is used - looks and behaves as a collapsible group.
  - Compatibility mode is not used - is a pop-up group.
- A standard group width where the width is specified explicitly, but subordinate items do not fit in it and are stretched:
  - Compatibility mode with version 8.3.11 is used - the specified width is set as the minimum for a group, even if it is less than the minimum width of subordinates.
  - Compatibility mode is not used - the minimum group width cannot be less than the minimum width of subordinate items.
- System reaction to filling the `Group` cube command property and the dimension tables of external data sources when verifying configuration before saving:
  - Compatibility mode with version 8.3.11 is used - configuration can be saved.
  - Compatibility mode is not used - configuration cannot be saved. Before saving configuration, fill this property.
- Displaying a spreadsheet document in black and white viewing mode:
  - Compatibility mode with version 8.3.11 is used:
    - charts are displayed in color;
    - a geographical schema is displayed in color;
    - OLE objects are displayed in color;
    - pictures of the svg, emf and wmf formats are displayed in color;
    - pictures of other formats are displayed in shades of grey.
  - Compatibility mode is not used:
    - charts are displayed in black and white mode;
    - geographical schema is displayed in black and white mode;
    - pictures and OLE objects are displayed in shades of grey.
- Displaying a spreadsheet document in black and white printing mode:

- Compatibility mode with version 8.3.11 is used:
  - charts are displayed in color;
  - a geographical schema is displayed in color;
  - OLE objects are displayed in color;
  - pictures of the svg, emf and wmf formats are displayed in color;
  - pictures of other formats are displayed in shades of grey.
- Compatibility mode is not used:
  - charts are printed in black and white mode;
  - geographical schema is printed in black and white mode;
  - pictures and OLE objects are printed in shades of gray.

- An algorithm of the fill pattern selection in printing a chart in black and white:
  - Compatibility mode with version 8.3.11 is used - picked on the basis of the color, without analyzing of the use. 12 fill patterns are available.
  - Compatibility mode is not used -picked on the basis of the color, but if a retrieved pattern is already used in this chart - the next available pattern is used. If all available patterns are used - they will be reused. 27 fill patterns are available.
- Re-specifying a value, where the `DeleteFromTempStorage()` method is used, as the `Address` parameter of the `PutToTempStorage()` method:
  - Compatibility mode with version 8.3.11 is used - available only if a form, UUID of which is passed as the `PutToTempStorage()` method calling, is open during the following storing to a temporary storage.
  - Compatibility mode is not used - causes an exception generation.
- Behavior of the `SwitchInterface()` global context method when operating in the managed mode in any client application:
  - Compatibility mode with version 8.3.11 is used -causes a display of command bars set in the configuration.
  - Compatibility mode is not used - ignored. It is not recommended to use standard forms when operating in the managed mode of a client application.
- Result of operating of the `CurrentLanguage()` global context function when performing on a server, in thick client and in an external connection:
  - Compatibility mode with version 8.3.11 is used - when performing on server, in thick client and in an external connection, returned the `Undefined` value, if a language not existing in a configuration is specified for the current user.
  - Compatibility mode is not used - returns an interface language selected for the current user or a default language of a configuration, if the current user is missing, a language is not specified for the current user or a language not existing in a configuration is specified.

- Priority of using conditional appearance items:
  - Compatibility mode with version 8.3.11 is used:
    - The closer a conditional appearance item is to an area of use, the higher it's priority.
    - Within one list of conditional appearance items, the lower a conditional appearance item is in a list of conditional appearance items, the higher it's priority. The exception is the case when a list of conditional appearances contains appearances with specified and not specified formatted fields and specified filters. In this case, conditional appearances with specified formatted fields are applied first and then conditional appearances with non-specified formatted fields.
  - Compatibility mode is not used:
    - The closer a conditional appearance item is to an area of use, the higher it's priority. For example, if there is a conditional appearance for the whole report, a table and a table grouping, then a grouping conditional appearance has the highest priority, then - a table, then - the whole report.
    - Within one list of conditional appearance items, the lower a conditional appearance item is in a list of conditional appearance items, the higher it's priority.
- Generation of a filter presentation string in a data composition system (in reports and dynamic lists):
  - Compatibility mode with version 8.3.11 is used - the qualifier specified for the data composition field for which the selection is performed is not taken into account.ored.
  - Compatibility mode is not used - the generation considers the qualifier that is specified for the data composition field for which selection is performed.
- Reaction to the «divide by 0» operation in the `CalculateExpression()` function of data composition system:
  - Compatibility mode with version 8.3.11 is used - an expression calling this function is fully calculated, but the Divide by 0 string is used as a result of the `CalculateExpression()` function operating.
  - Compatibility mode is not used - a result of calculation of an expression calling the `CalculateExpression()` function will be equal to the Divide by 0 string.
- A calculated field value in a detailed record of a data composition system in case a blank expression is specified for this field:
  - Compatibility mode with version 8.3.11 is used - an error with a message that a field is not found is generated.
  - Compatibility mode is not used - `NULL` will be used.

- The Scale property when specifying an absolute font in the font selection dialog box (in the Designer, thin and thick clients):
  - Compatibility mode with version 8.3.11 is used - ignored.
  - Compatibility mode is not used - included.
- Behavior of data composition system setting composer, if the capability to use a value list is set for a parameter of a data composition system, and a set parameter is an array:
  - Compatibility mode with version 8.3.11 is used - a value list, containing one item with the same value as the default value for a value type specified in a parameter, is placed in a parameter.
  - Compatibility mode is not used - an array is converted to a value list correctly.
- Parameter values and behavior of the `OnIntervalEditEnd` event handler of a form item displaying Gantt chart:
  - Compatibility mode with version 8.3.11 is used:
    - A copy of a chart interval was passed as the `Interval` parameter value.
    - Changes made in a passed interval were never saved in a chart.
  - Compatibility mode is not used:
    - A chart interval itself is passed as the `Interval` parameter value.
    - Values edited in a passed interval will be moved to a chart, if the `CancelEdit` handler parameter is not set to `True`.
    - If the `CancelEdit` handler parameter is set to `True`, values that were before editing start will be set in the `Start` and the `End` interval properties, and values from an interval will be copied to other values.
- If a query parameter is located in an expression of a query language for which a name was set, or an expression contains a parameter as an independent expression:
  - Compatibility mode with version 8.3.11 is used - this parameter is not placed in an available data parameters list of the data composition system.
  - Compatibility mode is not used - this parameter will be available for usage in data parameters of the data composition system, with a visibility restriction for end user.
- If the `UserSettingsKey` parameter is specified and the `UserSettings` parameter is not specified on a report form opening, then:
  - Compatibility mode with version 8.3.11 is used:
    - If user settings are stored in the system settings storage for the current report option, these setting will be loaded.
    - Otherwise, if there are settings with a specified key in a user settings storage, these settings will be loaded.

        - Otherwise, default settings for the current option will be set.
    - Compatibility mode is not used:
        - If there are settings with a specified key in a report user settings storage - these settings will be loaded.
        - Otherwise, default user settings for the current report option will be loaded.
- Поведение системы в том случае, если во вложенном запросе, находящемся в правой части левого соединения, поле явно не указано в предложении `{ГДЕ}`, и во внешнем запросе для поля с таким же именем компоновки данных используется не точно это же поле:
    - Compatibility mode with version 8.3.11 is used - when using automatic filling of available data set fields, a composer of a data composition template always places a filter in the nested query.
    - Compatibility mode is not used - a composer of the data composition template will not place a filter in a nested query.
      При необходимости обеспечить прежнее поведение, разработчику следует явно указать поле доступным для отбора в запросе с помощью предложения `{ГДЕ}`.
- Displaying a template with a blank string after header templates (a report, a table, a grouping, a chart) of a data composition system:
    - Compatibility mode with version 8.3.11 is used - displayed in some cases.
    - Compatibility mode is not used - not displayed.
- If in data composition system, fields that are child fields-characteristics of tabular sections fields retrieved by using a dot (for example, `Link.Goods.Good.[Country of origin]`), then in the source of available settings of data composition system for a dynamic list, availability of these fields for filter, sorting and groupings depends on the set compatibility mode:
    - Compatibility mode with version 8.3.11 is used - fields are available.
    - Compatibility mode is not used - fields are unavailable.
- Displaying of overall totals vertically in report tables of a data composition system, depending on the `ResourcesAutoPosition` parameter value of data composition system settings:
    - Compatibility mode with version 8.3.11 is used - overall totals are not displayed vertically, if the `ResourcesAutoPosition` parameter is set to `DontUse`.
    - Compatibility mode is not used - does not affect.
- Dragging of cells and pictures of a spreadheet document displayed in web client:
    - Compatibility mode with version 8.3.11 is used - standard dragging processing is not performed.

    - Compatibility mode is not used - standard dragging processing is performed: copying of cells or pictures when dragging with pressed Ctrl, or dragging of cells and pictures.
- Behavior of a query with the MERGE sentence when data can contain NULL:
    - Compatibility mode with version 8.3.11 is used - a query containing the MERGE sentence may not remove duplicated NULL retrieved by the CAST AS Link operation.
      The ISNULL() function from the CAST AS Link expression can return an incorrect result.
    - Compatibility mode is not used - a query containing the MERGE sentence (without the ALL keyword), the CAST AS Link operation will contain one string with the NULL value.
      The ISNULL() function from the CAST AS Link expression returns the correct result.
- System behavior in case the table column associated with a dynamic list is located in the column group with the In cell grouping, a document retrieved by performing the Display list command:
    - Compatibility mode with version 8.3.11 is used:
        - In thin client a column name of a form table is used as a header for this column.
        - In web client a column header of a form table is used as a header for this column.
    - Compatibility mode is not used - a column header of a managed form table is used as a header for this column.
- Setting a string and column docking in generation of a header template of data composition based on a predefined template:
    - Compatibility mode with version 8.3.11 is used - docking is never set.
    - Compatibility mode is not used - string and column docking is set, if it is required. Column docking (the FixLeft property of the data composition template area cell) is set, if the FixLeft report parameter is set to Auto or is not set. String docking (the FixTop property) is set, if the FixTop report parameter is set to Auto or is not set.

7.2.9. With version 8.3.12

- System behavior when calling the AddRow(), the ChangeRow(), the CopyRow() and the EndEditRow() methods of the ManagedFormTable object:
    - Compatibility mode is used - behavior depended on a client application type:
        - In thin client, appropriate actions are performed:
        - In web client, no action is performed.

    - Compatibility mode is not used- no action is performed, if interactive commands corresponding these methods are unavailable.
- Behavior when calling a serialized background job depending on the parameters size:
    - Compatibility mode is used - if a size of serialized parameters of a background job exceeds 2 GB - an exception is generated.
    - Compatibility mode is not used - if a size of serialized parameters of a background job exceeds 1 GB - an exception is generated.
- The maximum size of a spreadsheet document page:
    - Compatibility mode is used - no limits. When export and printing a document, problems can arise.
    - Compatibility mode is not used - any side size (the width and the height) is limited to 5 000 mm max. An attempt to set a value greater than a specified one can cause an exception generation.
- A chart output to a spreadsheet document:
    - Compatibility mode is used - a data composition system output processor does not set an automatic column width calculation for the last column in a cell area that contains a chart.
    - Compatibility mode is not used - a data composition system output processor sets an automatic column width calculation for all cells of a spreadsheet document area that contains a chart.
- Behavior, if the `IS NULL` query language operator is applied to an expression where a combination of several virtual fields or a combination of a virtual field and an expression that does not contain virtual fields is used:
    - Compatibility mode is used - a result of function operation is always the `False` value.
    - Compatibility mode is not used - a result of function operation will be the `True` value, if otherwise, an expression result is equal to `NULL` and `False`.
- The `MarkIncomplete` property value for a managed form table, when setting the `AutoMarkIncomplete` property of a managed form table to `True`:
    - Compatibility mode is used - was updated only after a table update or the current cell editing.
    - Compatibility mode is not used - will match an actual table state after 1C:Enterprise language operating end.
- Behavior in a calculated field output to a value collection, if in a calculated expression of a data composition system the `SELECTION` construction is used, where fields with an unknown type are retrieved in sections (for example, if fields are retrieved from a value table and a field type is not specified), and the `Undefined` value is specified in the `ELSE` section:
    - Compatibility mode is used - only the `NULL` type is specified as a column type.

    - Compatibility mode is not used - the `Arbitrary` type is specified as column type (type description will not contain any type).
- Behavior of a managed form table when setting a string filter (assignment of a value to the `RowFilter` property), if there is more than one item in a set filter, and a filter value is equal to `Undefined` for the first item:
    - Compatibility mode is used - string filter is not performed.
    - Compatibility mode is not used - string filter is performed.
- System behavior, if predefined groupings templates are used for a table columns output:
    - Compatibility mode is used - resources fields headers cannot be output in an overall totals template, if the height of a predefined grouping template is less than the height of an overall totals template assumed by a template generator.
    - Compatibility mode is not used - the height of an overall totals template is calculated for displaying headers of all output resources in a template. The template generator automatically increases the predefined grouping template height If it is less than the overall totals template height.
- Behavior of a data composition system filter, if a filter with the `Filled`/`NotFilled` type of comparison is used in data composition system settings and this filter is applied to a calculated field which type can be similar to a field of the `Characteristic` type or a type defined by a user or any link - type:
    - Compatibility mode is used - a filter ignores types included in `Characteristic`, a defined type or any link.
    - Compatibility mode is not used - a filter will consider types included in `Characteristic`, a defined type or any link.
- A type of a temporary table field generated by the `PLACE` query language operator, where `NULL` can be placed:
    - Compatibility mode is used - has the `NULL` type.
    - Compatibility mode is not used - becomes composite. A composite type includes the following types: `Number` with an accuracy of 1.0, `String` of the 1 character length, `DateTime`, `Boolean` and `NULL`.
- Applying a form conditional appearance for a dynamic list appearance, if the main table containing groups and items is specified in a dynamic list, then for all columns not related to the main table fields:
    - Compatibility mode is used - a form conditional appearance is applied as follows:
        - For columns related to ordinary fields of not the main table used in a query (not fields separated by a dot, expressions etc.) - if the main table item is displayed in a string.
        - For other columns not related to the main table fields - if the main table group is displayed in a string.

    - Compatibility mode is not used - a form conditional appearance is applied regardless of displaying an item or a group of the main table in a string.
- Retrieving a value of a functional option stored in a resource of periodic information registers:
    - Compatibility mode is used - a value is calculated by the logical OR operation among all stored values not limited by slice last. This behavior is occurred, if only the `Period` parameter is specified when retrieving a functional option value.
    - Compatibility mode is not used - only a value retrieved from slice last is used. A date specified for the `Period` functional option parameter or the current date is used as slice date (if the `Period` parameter is not specified).
- Accessibility of child fields of a composite type in user settings of a data composition system.
    - Compatibility mode is used - if user has no the View right for one of a field table of a composite type, all child fields of an unavailable type are missing in user settings.
    - Compatibility mode is not used - child fields of a composite type are available in user settings, if user has the `View` right for at least one table from a composite type and a field itself. Child fields become available for selection, can be included in filter, sorting, a conditional appearance and in other places of user settings where available fields are.
- A median calculation in a data analyzing feature, if selection contains an even amount of items (including two items):
    - Compatibility mode is used - a median value depended on amount of items in selection:
        - Selection contains more than 2 values - the arithmetic mean of two middle selection items.
        - Selection contains exactly 2 value - the second item value.
    - Compatibility mode is not used - the arithmetic mean of two middle selection items is used as a median.
- System behavior on opening reports forms and dynamic lists, if user settings are missing in a storage:
    - Compatibility mode is used - the following behavior is occurred:
        - Items with unavailable and missing fields are not removed from the main settings of a dynamic list and settings of a report option that are marked as user settings.
        - Items with unavailable and missing fields are not removed from user settings generated by default.
        - `Handlers of the BeforeLoadUserSettingsAtServer` and `the OnLoadUserSettingsAtServer events` are not invoked.

When performing the Set default settings command of report forms and dynamic lists, the following behavior is occurred:

- Items with unavailable and missing fields are not removed from the main settings of a dynamic list and settings of a report option that are marked as user settings.
- Items with unavailable and missing fields are not removed from user settings generated by default.
- If the user settings of a dynamic list or a report option were not changed on the client (interactively or from 1C:Enterprise language) since the form was opened, the default settings generated based on the main dynamic list settings or on the current report option settings are saved as user settings when the form was closed.

The `DataCompositionSettingsComposer.Restore()` method invoked with the `DataCompositionSettingsRefreshMethod.Full` parameter disables the `Use` check box in the main settings items disconnected with available fields.

- Compatibility mode is not used - the following actions are performed:
  - Items with unavailable or missing fields are deleted from the main dynamic list settings and report option settings that are marked as user settings.
  - Items with unavailable or missing fields are removed from the user settings generated by default.
  - `BeforeLoadUserSettingsAtServer` and `OnLoadUserSettingsAtServer` event handlers are executed.

  `New UseDefaultSettings` parameter for `BeforeLoadUserSettingsAtServer` and `OnLoadUserSettingsAtServer` event handlers.

  The Set default settings command of report and dynamic list forms performs the following actions:

  - Items with unavailable or missing fields are deleted from the main dynamic list settings and report option settings that are marked as user settings.
  - Items with unavailable or missing fields are removed from the user settings generated by default.
  - If the user settings of a dynamic list or a report option were not changed on the client (interactively or from 1C:Enterprise language) since the form was opened, the current user settings are deleted from the system storage when the form is closed.

  Method `DataCompositionSettingsComposer.Refresh()` with parameter

`DataCompositionSettingsRefreshMethod.Full` performs the following actions: sets the `Use` flag to False for main settings items that lost their linking to available fields, and deletes items of main settings and user settings that are marked as user settings but contain fields absent from the collection of available fields or not available for the current functional options and rights.

- `InternetMail.GetMessagesCheckboxes()` method behavior if no message corresponding to the messages sent as the parameter ID is found in the mail:
  - Compatibility mode is used - an exception is generated.
    Compatibility mode is not used - an empty fit is returned.

7.2.10. With version 8.3.13

- Print of the report containing the diagram:
  - Compatibility method is used - the diagram can be "cut" by the page breaks.
  - Compatibility method is not used - the diagram is always printed whole on one page. Upon printing, the data composition system output processor sets `StayWithNext` property value to `True` for the spreadsheet document rows that follow the chart.
- Field alias formation in generation of data composition template:
  - Compatibility mode is used - default query wizard naming rules are applied to create a field alias.
  - Compatibility mode is not used - the field alias includes the source field alias and the additional information description (representation of field attribute, aggregate function, etc.).
- Use of data composition system of the DBMS query language for the data aggregation:
  - Compatibility mode is used - the DBMS query language is not used for the data generation, except when `GROUP BY` is offered in the query.
  - Compatibility mode is not used - upon generation of a data set selection query, the data composition template generator makes an attempt to aggregate the data using the DBMS query language. Query language functions applied to a single field for data aggregation: `SUM()`, `MINIMUM()`, and `MAXIMUM()`.
- The possibility to get the help information on the global commands and command interface sections:
  - Compatibility mode is used - the help is displayed as follows:
    - If a section or global command has a help topic, the tooltip for the section in the sections or global command panel contains the Help hyperlink.
    - The functions menu commands have no tooltips.

        - In the web client, section tooltips in the sections panel contains the section synonyms, not the explanations.
        - In the web client, in the current section functions panel, tooltips of lists and objects contains the command synonyms, not the explanations or extended presentations of lists and objects.
    - Compatibility mode is not used - the help is displayed as follows:
        - If a section or command interface command has a help topic, its tooltip does not contain the Help hyperlink. The help topics are still available from the help window.
        - In the current section functions panel and the functions menu, lists and configuration objects have tooltips with their explanations or extended presentations.
- Interpretation of `Auto` value in `ChoiceFoldersAndItems` properties for the form input field displaying the reference-type data:
    - Compatibility mode is used - `Auto` value is interpreted as `FoldersAndItems`.
    - Compatibility mode is not used - `Auto` value is interpreted as `Items`.
- Comparison of various values in the data composition system:
    - Compatibility mode is used - the algorithm output for objects of the same type (including arrays and value tables, excluding objects of primitive type and reference type) is comparison of the object RAM addresses.
    - Compatibility mode is not used.
- Error arising when one of the operands of «+» operation of the query language exceeds 1,024 characters:
    - Compatibility mode is used - the error wording is Incorrect parameters "+".
    - Compatibility mode is not used - the error wording is Unlimited length strings are disallowed in "+" operation.
- Dimension of the result of work of `SUBSTRING()` function of the query language:
    - Compatibility mode is used - the resulting string length was always equal to the function third parameter value. The first parameter value length was ignored.
    - Compatibility mode is not used - the resulting string length equals the third parameter value, but does not exceed the first parameter value.
- System behavior when the form input field containing the modified data is changed:
    - Compatibility mode is used - setting the `Visible` property of an input field to `False` results in the following:
        - If the field is linked to a text value, its value is generated from the field text and then the `OnChange` event handler is called.

        - If the field is linked to a reference value and a user has not finished filling it, the resulting field state might be inconsistent.
    - Compatibility mode is not used - if a form input field contains modified data and it is not the current form item, an attempt to set its `Visible`, `Enabled`, or `EditText` property to `False` or set its `ReadOnly` property to `True` results in the following:
        - `TextEditEnd` event handler is called.
        - After the `TextEditEnd` event handler is completed, if the `StandardProcessing` parameter is set to `True`:
            - If a valid value is generated from the field text, the `OnChange` event handler is called.
            - If a valid value cannot be generated from the field text, the input field value is set to the value of the attribute linked to the input field.
        - After the `TextEditEnd` event handler is completed, if the `StandardProcessing` parameter is set to `False` and a valid value cannot be generated from the field text, the input field value is set to the value of the attribute linked to the input field.

    If a form input field contains unmodified data and it is not the current form item, an attempt to set its `Visible`, `Enabled` or `EditText` property to `False` or set its `ReadOnly` property value to `True` results in closing the drop-down list (if the list was open).
- Full-text search behavior when «*» wildcard character together with a high-frequency prefix is used:
    - Compatibility mode is used - search for expressions like "word*" may fail due to exceeding search word threshold error.
    - Compatibility mode is not used - exceeding search word threshold is less likely for expressions like "word*".
- Specific features of drawings import from XLS (Microsoft Excel 97), Microsoft Office OpenXML (Microsoft Excel 2007 and older) or ODS (OpenDocument) documents:
    - Compatibility mode is used:
        - For shapes, if the text is blank, the import is not performed; if the text contains only spaces, the import is performed.
        - For pictures, if the text is blank or contains only spaces, the import is performed.
    - Compatibility mode is not used - a drawing of `Text` type is not created in the target spreadsheet if a shape or picture text in the source spreadsheet is blank or contains only spaces.
- Drawing type to be used in a spreadsheet cell:

- Compatibility mode is used: - for spreadsheet document drawings that are comments, the `DrawingType` property value is `SpreadsheetDocumentDrawingType.Text`.
- Compatibility mode is not used - for spreadsheet document drawings that are comments, the `DrawingType` property value is `SpreadsheetDocumentDrawingType.Comment`.

- Resource fields export to the grouping template in some cases:
  - Compatibility mode is used - the data composition system template generator does not output resource field groupings to templates if a predefined template describes the output of the parent grouping and resources are not specified in the selected fields of the parent grouping but are specified in the fields of the child grouping.
  - Compatibility mode is not used - the data composition system template generator outputs resource field groupings to templates even if a predefined template describes the output of the parent grouping and resources are not specified in the selected fields of the parent grouping but specified in the fields of the child grouping.
- Query behavior if a query selection list does not contain tabular section attributes, but the query contains a comparison of tabular section attributes combined with "AND" operator:
  - Compatibility mode is used - the conditions are performed in isolation which may lead to wrong results of the comparison.
  - Compatibility mode is not used - the conditions are applied together to each tabular section row.
- Use of conditional appearance of the data composition system template generator when it output fields to chart grouping templates:
  - Compatibility mode is used - it applied conditional appearance only to nonresource fields.
  - Compatibility mode is not used - it applies conditional appearance to all of the fields.
    As a result, in reports based on data composition schemas the conditional appearance applied to resource fields is also applied to charts.
    This might change the look of reports containing charts with resources that have conditional appearance.
- Text in spreadsheet document cells with "Justify" alignment:
  - Compatibility mode is used - looks as text with "Left" alignment.
  - Compatibility mode is not used - the text is displayed correctly.
- The result of the work of the query language function `ValueType()`, which is applied to an expression containing `ThisGroup` field:
  - Compatibility mode is used - field returned an incorrect value. The result is calculated as if the `IsFolder` field value contains an inverted value.
  - Compatibility mode is not used - returns an incorrect value.

- The result of the work of the query language function `ValueType()`, which is applied to an expression in which both the virtual field and a primitive type value participate:
  - Compatibility mode is used - field returns an incorrect value.
  - Compatibility mode is not used - returns a correct value.

7.2.11. With version 8.3.14

- Operation of the data composition system with additional characteristics upon getting the characteristic values (for any purpose, for example, sorting or filtering by it):
  - Compatibility mode is used - a characteristic value type is not considered upon getting a characteristic value. Even if a value, which does not match the used characteristic value type, is written in the infobase, it will be used.
  - Compatibility mode is not used - a characteristic value type is considered upon getting a characteristic value. If a value, which does not match the used characteristic value type, is written in the infobase, then `NULL` will be used instead.
- Adding the commands Generate, Print, Save, and Compact mode to the report form command bar (with enabled automatic filling) depending on whether the property of the report form Report result is set:
  - Compatibility mode is used - are always added to the report form command bar with enabled automatic filling.
  - Compatibility mode is not used - are added to the report form command panel only if Report results report form property is set. If a property of the report form Report result is not set, the commands will not be added to the report form command bar, which is filled in automatically.
- System behavior when processing a query text containing the right joints with the query constructor, query scheme, data composition system, and report builder:
  - Compatibility mode is used - right joins are replaced automatically with similar left joins. If conversion failed to be performed - the connection is removed from the query.
  - Compatibility mode is not used - right joins are not replaced automatically with similar left joins.
- Behavior of composer of data composition system template for calculated fields, data set fields, and calculated resource fields:
  - Compatibility mode is used - composer of data composition template uses an expression of the calculated field presentation being a resource in detailed, group, and total records.

- Compatibility mode is not used - composer of data composition template does not use the presentation expression specified for the calculated field upon generating an expression of group and total records for calculated fields being resources. Thus, the presentation expression for calculated fields being resources will be displayed in the report in detailed records only.

- Query language work with the expressions being the virtual tables parameters:
  - Compatibility mode is used - the query language has no optimized calculations of expressions being virtual table parameters.
  - Compatibility mode is not used - the query language now has optimized calculations of expressions being virtual table parameters. This optimization is similar to optimization of standard query language expressions.
- Determination of used fields by the composer of data composition system template:
  - Compatibility mode is used - composer of data composition system template considers to be used all fields in conditional appearance item even if this conditional appearance item is not applied to any field. All fields, which are formally used in conditional appearance, are added to the data acquisition query.
  - Compatibility mode is not used - composer of data composition system template considers to be used only those fields that are used in the conditional appearance item of data composition only when this conditional appearance item is applied at least to one of the displayed fields (also upon applying conditional appearance to all fields). If conditional appearance item is not applied to any field, used fields are considered to be unused and are not added to the resulting query of the data composition template. Fields, which are not used in conditional appearance, are not added to the data acquisition query.
- `CASE` operation use as the a virtual table constant parameter:
  - Compatibility mode is used - query language operation `CASE` is not replaced with a constant and can't be used in constant parameters of virtual tables. Upon calculating operation `CASE` in query totals, if the operation operand contains `NULL`, the operation result is `NULL`.
  - Compatibility mode is not used - query language operation `CASE` will be considered a constant and replaced with a constant if all operation branches contain only constant expressions. Therefore, operation `CASE` can be used in constant parameters of virtual tables if all branches of operation `CASE` contain only constant expressions.

    If an operand of operation `CASE` contains `NULL` in query totals, then the operation result will be a value specified in `ELSE` or `NULL` if it is absent.

- The field separated by dot from the temporary table field or nested query availability check by the data composition system:
  - Compatibility mode is used - availability check does not include checking that the field is not disabled with a functional option. If a temporary table field or a nested query field is available, it is considered that all fields available via "." (dot) are also available.
  - Compatibility mode is not used - availability check includes checking that the field is not disabled with a functional option.
- Use of query language functions `SUBSTRING()`, `YEAR()`, `QUARTER()`, `MONTH()`, `DAYOFYEAR()`, `DAY()`, `WEEK()`, `WEEKDAY()`, `HOUR()`, `MINUTE()`, and `SECOND()` in the constant parameters of virtual tables:
  - Compatibility mode is used - the function is disallowed to be used in the constant parameters of the virtual tables whatever the function parameter value is.
  - Compatibility mode is not used - the functions are deemed to be constants and are replaced with a constant if all function parameters contain only constant expressions. In the result, these functions can be used in the constant parameters of virtual tables if all function parameters contain only constant expressions.
- Application of conditional appearance to the columns of a data composition system report table:
  - Compatibility mode is used - is applied only to the title and column total.
  - Compatibility mode is not used - is applied to the whole column of the report.
- Calculation of the data composition system expression type for the expressions containing the parameters and fields with arbitrary type.
  - Compatibility mode is used - if a value with an arbitrary type is present in a data composition system expression, an arbitrary type is ignored upon calculating the resulting expression time.
  - Compatibility mode is not used - if one of types in the data composition system expression is arbitrary then the target expression will be of arbitrary type.
- Calculation of fields (% common, % in grouping, % in hierarchy group, % in string or series, % in column or dot, % in string or series grouping, % in column or dot grouping, % in string or series hierarchy group, % in column or dot hierarchy group) of the data composition system if the value of the resource from which this field is obtained is 0 in the current grouping:
  - Compatibility mode is used - values are always divided by a parent grouping value or grand total, except for cases when the resource value in the grouping is equal to `NULL`. As a result, an exception could be thrown due to division by 0.
  - Compatibility mode is not used - the field value is 0.

- System behavior in case a new value is attributed to `Chart.DataSource` property:
  - Compatibility mode is used - the chart is not cleared, the selection is not cleared.
  - Compatibility mode is not used - the method `Clear()` `Chart` object will be called after the attribution. Therefore, after a new value is assigned, the current selection is cleared.

7.2.12. With version 8.3.15

- Menu Additions (Search Control form item) generation:
  - Compatibility mode is enabled - menu consists of all commands available for related form table (similarly to autofilling of table command bar). Redundant commands are disabled.
  - Compatibility mode is disabled - menu consists of commands which are to be added to this menu only: search by current value, advanced search, search history and search cancellation.
- System behavior when you press Enter (or double-click a row) in a row of a table linked to a dynamic list or when you call `ChangeRow()` form table method, a form table extension:
  - Compatibility mode is enabled (version 8.3.10) - a form related to this row will be opened irrespective of whether table commands include Edit command, or not.
  - Compatibility mode is enabled (version 8.3.11 - 8.3.15) - a form related to this row will be opened, only if a table includes Edit command.
  - Compatibility mode is disabled - a form related to this row will be opened irrespective of whether table commands include Edit command, or not.
- The platform supports XSD schemas containing types, in which attributes are defined via references to global attributes, but overriding such properties as `default` or `fixed` ones:
  - Compatibility mode is enabled - when XDTO model is verified, an error is returned. Such schemas cannot be used.
  - Compatibility mode is disabled - they can be used.
- Verification of availability of filter fields in a list of available fields:
  - Compatibility mode is enabled - no verification is performed.
  - Compatibility mode is disabled - verification is performed. When the first field unavailable for filter is found, an error is generated.
- Use of fields with different expressions for selection and filter as key fields of a dynamic list.
  - Compatibility mode is enabled - they can be used. In this scenario, certain hardly found errors can be observed in the dynamic list operation.
  - Compatibility mode is disabled - they cannot be used.

- Alignment of data set query and data composition schema sorting:
  - Compatibility mode is enabled - data composition template composer adds sorting from the query after the existing data composition schema sorting.
  - Compatibility mode is disabled - data composition template composer add sorting from the query before the existing data composition schema sorting.
- Kind of a data composition field which is made available due to use of query language aggregate functions `SUM()` and `AVERAGE()`:
  - Compatibility mode is enabled - a field kind being a parameter of the said aggregate functions is used as a field kind.
  - Compatibility mode is disabled - a value of indefinite length is used as a field kind.

7.3. Specifics of the extension compatibility mode

In version 8.3.7

- Adding child objects for adopted objects:
  - Compatibility mode with version 8.3.7 is used - an action is not supported.
  - Compatibility mode is not used - an action is supported.
- Predefining properties of adopted objects in an extension:
  - Compatibility mode with version 8.3.7 is used - an action is not supported.
  - Compatibility mode is not used - an action is supported.

With version 8.3.10

- Verification of extensions applicability before applying a configuration extension:
  - Extension compatibility mode with version 8.3.10 is used - not required.
  - Extension compatibility mode is not used - is required.
- Adding custom catalogs and information registers and also adding the main configuration of attributes and table sections to catalogs:
  - Extension compatibility mode with version 8.3.10 is used - an action is not supported.
  - Extension compatibility mode is not used - an action is supported.

7.4. Specifics of web client operations

- The following is not supported in Google Chrome and Safari web browsers:
  - Programmatic insertion from the clipboard, and starting from Google Chrome 6 and Safari 5 versions - any operations of programmatic access to the clipboard. Only keyboard clipboard commands (not the context menu) are available.
  - Automatic operating system authentication.
  - A feature of changing footers when printing.
- The cryptography tools extension is not supported in Safari web browser.
- The following is not supported in Microsoft Edge web browser:
  - Add-in technology;
  - File system extension;
  - Cryptography extension.
- Operations with COM objects are supported only in Microsoft Internet Explorer.
- OS X only supports password input using Latin alphabet letters and digits. iOS up to version 5.x (including 5.x) only supports password input using the Latin alphabet letters and digits. iOS 6 version supports the use of national alphabet characters in passwords. To ensure compatibility between various OS versions or when application use is required in Safari web browser, it is recommended to set passwords using digits and letters of the Latin alphabet only.
- When a page containing a web application is added to Bookmarks in Mozilla Firefox web browser using the Add page to bookmarks command, the user has to adjust the added link manually. To do this, open the properties editing window for the added bookmark and remove text from the beginning of the line up to http:// in the address field or by drag-and-dropping the application icon (left of page address) into Bookmarks menu.
- Whenever a web client is used in Google Chrome web browser with 1C:Enterprise Extension a situation may arise when an extension remains inoperable upon web browser startup. As such, no operations with a clipboard can be available in a web client. Whenever it arises, it is recommended to restart your web browser.
- User interruption of module operation is not supported. When performing the method call module `UserInterruptProcessing()` is ignored.
- `The` LockApplication() `method is not supported` .
- The `Calculate()` function does not diagnose the Procedure called as function error.
- IDs used in the expression, transfered to the `Calculate` function, can be recorded only in Russian and English languages.

- If there is an error in the expression transferred to the `Calculate()` function, the error is identified as a method call error, and not as an error in the transferred expression.
- Depending on the web browser used, either the original SVG format image is displayed, or the image is pre-converted into PNG format. The size of the image in PNG format corresponds to the size specified in the SVG image format. The following web browsers display original SVG images:
  - Internet Explorer 11 and older.
  - Google Chrome;
  - Mozilla Firefox;
  - Safari 7 and older.

  To display a picture in SVG format correctly, setting the picture size (using the `width` and `height` properties) is required, and to display it using Microsoft Internet Explorer web browser, -the `viewBox` property must be additionally set.
- Web client does not support debugging of code in 1C:Enterprise language in the application event handlers `BeforeExit()` and `OnExit()`, as well as in the managed form handlers `BeforeClose()` and `OnClose()` during application shutdown if it is done by closing the browser window.
- When performing time-consuming operations the cursor appearance is not changed.
- Horizontal scrolling (with mouse wheel when Shift is pressed) is not supported in all form items.
- Microsoft Internet Explorer and Mozilla Firefox browsers do not process keys pressed during a server call. In Google Chrome and Safari browsers these pressings are processed when the server call is finished.
- Hotkey definition using the & character in the command text, menu items, etc is not supported. The & character, specified in text, is ignored and not displayed in the interface.
- In Safari web browser, when the server is performing a query for a long time, an error can occur.
- It is not recommended to programmatically open and close one or more forms in one handler. These actions should be performed at different times. It is recommended to close forms in idle handler.
- If a form, that is opened in an auxiliary window, is closed by clicking the system close window button ("cross" on the top right corner of the form) or by pressing Alt + F4, the `BeforeClose` handler is not called for the closed form module.
- If a form, that is opened in an auxiliary window, is closed by clicking the system close window button (the cross on the top right corner of the form) before it is fully displayed on screen, a web browser error can occur.

- When changing form attribute values outside of the module of this form, the changes made are not displayed by form items. To display changes, you need to call the `RefreshDataRepresentation()` method.
- A minimized window in Google Chrome web browser is not activated if a warning, related to the contents of this window, is displayed.
- If you minimize a form of the modified object in Google Chrome and then close the main application window, a warning about unsaved data will be issued, but the form with this data will not be activated.
- If you need to display a form element text normally, then, when specifying the font size, you should specify both the font size and type of the vector font (TrueType font).
- In web client operating in Google Chrome web browser, when using the form items of “MS Shell Dlg” font, the font “Times New Roman” will be used instead of “Microsoft Sans Serif” font.
- Formation of value presentation (for display in the input field), that is transfered in the `ValueSelected` parameter of the `ChoiceProcessing` handler, occurs after the transferred value is converted to the type of attribute value, that is displayed by the input field.
- If the button height is more than 5 units (with standard font size), the button is shown without gradient.
- If the `Question()` global context function button list does not contain a button with the `DialogReturnCode.Cancel` function, an attempt to close the dialog box using the system close window button (the cross on the top right corner of the form) will reopen the dialog box.
- Status bar functionality is limited. While performing a module, status updates depend on the Web browser.
- The indicator field display differs from the thin client. In particular, the appearance of the indicator field is different, for which the `Display` property is set to `Intermittent` or `BrokenTilt` values.
- `Dendrogram` object functionality is limited.
- `GraphicalSchema` object functionality is limited:
  - graphical schema editing is not supported;
  - an object model is available in limited amount for the following objects: `GraphicalSchemaItemActivity`,`GraphicalSchemaItemSplit`,`GraphicalSchemaItemCondition`,`GraphicalSchemaItemConnectionLine`,`GraphicalSchemaItemDecorativeLine`,`GraphicalSchemaItemDecoration`,`GraphicalSchemaItemCompletion`,`GraphicalSchemaItemStart`,`GraphicalSchemaItemJoin`,`GraphicalSchemaItemSwitch`,`GraphicalSchemaItemProcessing`,`GraphicalSchemaItemSubBusinessProcess`.
- `GeographicalSchema` object functionality is limited.

- GanttChart object has limited interactive set-up features.
- Retrieval of system information about a client computer is not supported.
- Operations with TextReader, TextWriter, ZipFileReader, ZipFileWriter, XBase objects are not supported. It is recommended to operate with these files on the server side.
- Operations with XML and XDTO are not supported. It is recommended to operate with these files on the server side.
- Internal clipboard is implemented in Google Chrome and Safari browsers:
  - Data in the internal clipboard is relevant during the session.
  - All values in the operating system clipboard are also duplicated in the internal clipboard.
  - The M+ and M- commands change the value located in the internal clipboard. The acquired result is placed into the operating system clipboard.
  - The internal clipboard is displayed in the "1" cell of the calculator. MR, M+ and M- calculator commands operate with internal clipboard values. The acquired result is placed into the operating system clipboard.
- The calculator setup dialog box is not supported.
- The rules for transferring text in a cell of a spreadsheet document depend on the web browser used.
- Viewing and printing a spreadsheet document in black and white mode is not supported.
- The spreadsheet document printed from different web browsers may have different sizes.
- For correct printing of spreadsheet documents with non-standard page size from web browsers, it is recommended to use the PrintAccuracy property of the spreadsheet document. Otherwise, before printing, the user will need to check and, if required, set the sheet size in the print dialog box of the web browser in accordance with sheet size of the printed spreadsheet document.
- When printing a spreadsheet document from Google Chrome browser with DuplexPrinting property set for two-sided printing, it is recommended to print using the system dialog box. For a spreadsheet document, it is necessary to specify in the print settings, the use of the required type of two-sided printing. You can also use the PrintAccuracy property set to Accurate to print documents that use two-sided printing.
- When displaying multiline text in cells of a spreadsheet document, line spacing may differ depending on the web browser used.
- When displaying text in the cells of a spreadsheet document, the character spacing may differ depending on the web browser used.
- Only those strings of a multiline text that entirely fit in a cell along the height are displayed in a cell of a managed form table. If a string is not fully

displayed, the "…" character is displayed at the end of a string. In Microsoft Internet Explorer 8 web browser, three dots are not displayed at the end of the line.

- If as a result of changing table data source, the current table row is changed and at the same time the form item property value changes, which rebuilds the form, the `OnRowChange` handler call and form reload moment are not defined.
- In forms that are supposed to scroll vertically for items, such as table, spreadsheet document, chart, etc, vertical stretching should be disabled.
- The `GetHTML()` method of the `FormattedDocument` object returns a blank picture list.
- Highlighting text in formatted document field, hides the text color (text color and background color) when operating in Microsoft Internet Explorer web browser and thin client.
- Opening a spreadsheet document that contains a report, decrypts of which contain reference values, is not supported.
- In a spreadsheet document, a spreadsheet document field and a form table do not support scrolling using the mouse wheel.
- If a cell of a spreadsheet document has right alignment and the text does not fit completely in the cell, then text's position is aligned with the left margin.
- In Microsoft Internet Explorer 6.0 and 7 web browsers, the display of the merged cells of a spreadsheet document with a border on the left and no border on the bottom is not supported. It is recommended to specify the bottom border for such cells.
- In Microsoft Internet Explorer 7 web browser, for a spreadsheet document with cells of different widths in one column and text and a drawing in one cell, printing is performed with distortions.
- In Google Chrome and Safari web browsers, with a certain combination of merged cells and cell borders in a spreadsheet document, the border may be displayed incorrectly.
- For two adjacent (horizontally) cells of a spreadsheet document, the contiguous border will be framed with the same line as the right border of the left cell, if a spreadsheet document is displayed in Microsoft Internet Explorer 6.0 web browser.
- In Microsoft Internet Explorer and Google Chrome web browsers, the dotted border of adjacent cells in some cases can be displayed as solid.
- To correctly print spreadsheet documents containing dotted strings from Google Chrome and Mozilla Firefox web browsers (version 20 and higher), it is recommended to set the `PrintAccuracy` property to `Exact` for these spreadsheet documents. The same recommendation applies when

printing spreadsheet documents, for which you need to ensure exact dimension of all items to be printed from Google Chrome and Mozilla Firefox web browsers (version 20 and higher).

- The file comparison mode is not supported.
- The list of last opened files is not supported.
- Drag-and-dropping files into web client forms is not supported.
- Web client uses a limited set of fonts available in the operating systems used. Other fonts cannot be selected in the font selection dialog box of the web client, and when used are replaced with supported fonts.
- To access Service and Settings (main menu), use F10 key; access by pressing Alt key once is not supported.
- Pressing the Ctrl + W combination in Google Chrome web browser will not result in word selection, but will close the auxiliary window.
- Pressing the Ctrl + F4 combination in Google Chrome web browser closes the current tab. If it is the last tab, it closes the current web browser window.
- Pressing the Ctrl + N combination in Google Chrome web browser creates a new browser window and does not open the New Document creation menu.
- Web browsers do not support scale display, other than 100%, or using non-standard DPI.
- Navigation between windows using the keyboard has a number of features (depending on the web browser type):
  - Microsoft Internet Explorer:
    - Version 6.0 - no features.
    - Version 7 & 8 - navigation using Ctrl + Tab, Ctrl + Shift + Tab is not supported if tabs are enabled in the Web browser. Navigation using F6, Ctrl + F6 is available with no limitations. In Microsoft Internet Explorer 8 web browser, in some cases when navigating between windows a Click OK to navigate to the window with the OK button warning will be displayed. Transition to the window will be performed after pressing the OK key on the keyboard, additional configuration is required, which can be done using Service and Settings - Settings -Web Browser Settings dialog box.
  - Google Chrome:
    - For any version - navigation using the Ctrl + Tab, Ctrl + Shift + Tab is unavailable. Navigation using F6, Ctrl + F6 is available with no limitations.
    - For version 7.0 and earlier - when using the (F6, Ctrl + F6) hotkeys to switch web client windows, when attempting to re-open an already opened form or using the `Activate()` method to navigate, after switching to a new window a warning is displayed, saying Navigation to window completed. Click OK to continue. After

clicking OK you can continue operations with the window, to which the switch occurred.

- The modal window in Microsoft Internet Explorer web browser locks all parent windows, in Mozilla Firefox web browser only the parent window is locked, and in Google Chrome web browser windows are not locked. All other application windows are shaded, you can switch to them, but you can't perform any actions in them.
- When a modal window is opened, refreshing of the main window stops in Microsoft Internet Explorer.
- Web browser limitations make it impossible to open a modeless form from a modal one. Therefore, all forms to be opened from modal form are also opened in the modal mode.
- Size of the web client window cannot be less than a minimum size defined by the web browser.
- When operating in multiple monitors, if they have a different resolution set, windows, that are opened not on the first display, can have different coordinates when they are re-opened.
- Displaying the command bar with vertical display of buttons is not supported.
- Displaying a missing picture in web client differs from displaying a missing picture in thin client.
- Microsoft Internet Explorer 8 incorrectly displays images containing color with transparency other than 0 and 100%.
- Displaying pages with a web server host name in the address matching web client host name in an HTML document field is not supported. It is recommended to store pages in the infobase.
- Event processing of an HTML document field for pages with web server host name in the address differing from the web client host name is not supported. Pages that require event processing should be stored in an infobase.
- When `ChooseFromList()` and `ShowChooseFromList()` methods are used, the value list selection window is opened as a separate browser window and not as a list.
- When the `ChooseFromMenu()` method is used, the value list selection menu and data composition system details menu are opened as a separate browser window and not as a menu.
- When the `PlaceFile()` method is called, there could be situations when errors related to placing files are determined incorrectly. In this case `True` value is returned and the `Address` parameter does not contain a temporary storage address.
- In Safari web browser, the calculator and calendar windows are always opened in windows that are not smaller than the minimum window size for this browser

- Scroll bar color for HTML document field corresponds to the color specified by the operating system, except the case when Microsoft Internet Explorer web browser is used and HTML document field displays HTML page set by a text string.
- Calling the `DocumentComplete` and `OnClick` event handlers and executing Print and Save commands of an HTML document field is not supported if a document is displayed in the field, where the web server host name differs from the web client host name.
- Behavior of undo and return commands in multiline input fields differs from other clients. In the web client, the browser determines the number of changes that are rolled back.
- The result of left clicking a word in input field (with Ctrl key pressed) depends on the browser and does not result in selection of the word been clicked.
- Pressing the left/right arrows in a text document field in the first/last string position will move the pointer to the previous/next string.
- In a multiline input field and in a text document field, text entry in replace mode is not supported.
- In a multiline input field and a form table cell, text is wrapped only by words. Solid text at item border is not wrapped.
- When Shift+Del is pressed in a multiline input field, the whole string is deleted and automatic line separator on input field borders is not considered.
- If an invalid XML version 1.0 character (http://www.w3.org/TR/xml/) is programmatically placed to an input field, then in Google Chrome and Safari web browsers this character is replaced with "?" and the input field becomes read-only.
- If tooltip text contains line separators, then in Mozilla Firefox web browser white spaces are shown instead.
- The web client interface language in Safari web browser matches the browser interface language, if the interface language is not explicitly set using the L start parameter.
- When a web client is debugged and local variables or method parameters are used in the Expression window, a scoreboard or breakpoint conditions, their names should be specified exactly as they were set in the definition (case sensitive).
- The following features are not supported in the web client:
  - General:
    - print on the current printer.
  - Graphical schema field:
    - search and replace.
  - Text document:
    - transition to line;

        - operations with tabs;
        - Text submenu in Service and Settings menu (main menu).
- The web client does not support calling commands using the following hot-keys:
    - Picture field:
        - zoom in (Num+);
        - zoom out (Num-);
        - picture scrolling using keyboard.
    - Graphical schema field:
        - schema scrolling.
    - Table linked to a hierarchical list:
        - minimize node (Ctrl + Num-);
        - minimize all nodes (Ctrl + Shift + Num-);
        - maximize node (Ctrl + Num+);
        - maximize all nodes (Ctrl + Shift+ Num+).
    - Spreadsheet Document:
        - minimize group (Ctrl + Num-);
        - minimize all groups (Ctrl + Shift + Num-);
        - maximize group (Ctrl + Num+);
        - maximize all groups (Ctrl + Shift + Num+);
        - scroll page left (Alt + Page Up);
        - scroll page right (Alt + Page Down);
        - selecting cells using keyboard.
    - Text document:
        - delete current line (Ctrl +L): not supported in Mozilla Firefox, Google Chrome and Safari web browsers.
- In OS X operating system, when a non-English keyboard layout is enabled, hotkeys are not supported if the following keys are used in combinations:
    - in Google Chrome web browser: Alt and Alt + Shift keys;
    - in Safari web browser: Alt, Alt + Shift, Cmd and Cmd + Shift keys.
- The video call window of the collaboration system cannot be moved beyond the borders of the web browser window.

7.5. Features of web client operations on iPad

These specifics extend the list of web client behavior features described in the previous section.
When operating on an iPad:

- Instead of double-clicking the left mouse button, a quick double touch on the same screen position is used. Instead of a clicking the right mouse button, a long touch is used (more than 1 sec.).

- Only operation in tabs and Taxi interface is supported.
- The on-screen keyboard is not displayed automatically when you are editing a table cell or a spreadsheet document cell (in the displayed input field). To display the keyboard, click the input field edited again.
- Transition to the main window using the All Windows dialog box is not supported. If you try to transition to it, a warning with the following text is displayed Unable to transition to the main window.
- It is recommended to use the close button in the top right corner of the main window to finish operations in the web client.
- Dragging is not supported.
- Transitioning to the main window when auxiliary windows are open is not supported.
- Printing is not supported.
- A formatted document field allows only viewing of the document (without the editing option).
- The status panel is displayed using the same window as user notification window.
- The `PlaceFile()` method is unavailable.
- The file system extension, cryptography tools extension, and add-ins are unavailable.

The following object and global context methods cannot be used in an application running on iPad:

- Global context:
  - `Question()`,
  - `OpenValue()`,
  - `OpenFormModal()`,
  - `Warning()`,
  - `InputDate()`,
  - `InputValue()`,
  - `InputString()`,
  - `InputNumber()`.
- Object`FontChooseDialog`:
  - `Select()`.
- Object`ColorChooseDialog`:
  - `Select()`.
- Object`StandardPeriodEditDialog`:
  - `Edit()`.
- Object`FormatStringWizard`:
  - `DoModal()`.
- Object `DataCompositionDetailsProcess`:
  - `ChooseAction()`.

- Object`ScheduledJobDialog`:
  - `DoModal()`.
- Object `ValueList`:
  - `CheckItems()`,
  - `ChooseItem()`.
- Object `ClientApplicationForm`:
  - `ChooseFromMenu()` ,
  - `ChooseFromList()`,
  - `DoModal()`.

7.6. Features of system object behavior with the enabled data separation mode

7.6.1. Changing separator values

During the session, the use of separators and separator values can be changed by changing the session parameters. Session parameter values to which a common attribute is referencing can be changed at any moment of time according to limitations on rights of the user, on whose behalf the attempt to change the session parameters is made.
Replacing the current separator values, for example, allows to perform required administrative actions when operation in the separated mode is required.
However, you should keep in mind that if an infobase is accessed via web server, then changing object values, related to a separation mechanism, programmatically can be locked with the default.vrd file.
1C:Enterprise does not guarantee the logical integrity of application data (not infobase) after session parameters are changed, similar to changing session parameters affecting access restrictions.
Changing a session parameter value referenced by at least one separator in the Independent mode in 1C:Enterprise language leads to the following actions:

- object cache is cleared;
- reused values are deleted.

After changing values and using separators in session, it is recommended to call the `RefreshInterface()` method.

7.6.2. Objects numeration

The numerating mechanism works within each unique combination of separator values. For such a combination, automatic object numeration is used and uniqueness control is implemented.
When determining the number, separator values set in the session are used. If a separator is not used in the session, the value set for the infobase object is used for the separator.

7.6.3. Real-time timestamp

Real-time timestamp is separated within every unique combination of independent separators only if there are no documents for which real-time posting is enabled and which are not included in components of these separators.
For example, there are two independent separators in an application: `Subscriber` and `Organization`, and three documents (with enabled real-time posting) `ReceiptOfGoods`, `Invoice` and `Move`. The `Organization` separator includes `ReceiptOfGoods` and `Move` documents, and the `Subscriber` separator includes all documents. The real-time timestamp is separated only for the `Subscriber` separator, since there are documents for the `Organization` separator that are not separated by it (`Invoice` document):

	`Subscriber`	`Organization`
`ReceiptOfGoods`	Included	Included
`Invoice`	Included	*Not included*
`Move`	Included	Included
Real-time timestamp separation	**Separated**	**Not separated**

7.6.4. Predefined data

Predefined data can be used for objects included in any separator components.
When you first access the object table, a predefined data is only created if all separators are used in the session (including the separators in the Independent and Shared mode) that contain an object with predefined data.
To create a predefined data in a new data area, use the `InitializePredefinedData()` method. Depending on how separators, which include a configuration object, are used in a session, the following options can be distinguished:

1. All separators that include a configuration object are used in a session - predefined items will be created.

2. All separators that include a configuration object are not used in a session- predefined items will not be created and a configuration object will be skipped.

The UUID of data items related to a predefined data do not match for a data located in different areas. In other words, if the `Goods` catalog has a predefined item named `P1` and there are two areas: with codes `165` and `567`, the UUID of the data item associated with `P1` item in area `165`will not be equal to UUID of a data item associated with `P1` in area `567`.

In a distributed and separated infobase, changes to predefined items are registered as follows:

- If the exchange plan and the configuration object, for which registration of changes to the predefined data is performed, are included in the components of the same separator, then only those predefined items are registered, in which the value of the separator matches the value of the same separator of the exchange plan node. Thus, changes of predefined items are not registered if there are no nodes in the exchange plans, whose separator value matches the separator value of the predefined item. Registration is not performed in nodes whose separator value differs from the separator value of a predefined item.
- If the exchange plan and metadata object do not have common separators, then all predefined items are registered at all nodes of this exchange plan.
- Changes are never registered for `ThisNode` node.

7.6.5. Access rights

Administrative actions that are generally performed with an infobase and a specific data area are regulated with different rights.

- Administration - this right regulates administrative actions with the whole infobase and regulates a user list editing.
- Data administration - this right regulates administrative actions with a data. These operations include:
  - viewing the event log records and getting filter values without limitations;
  - setting the infobase (data area) time zone - calling the `SetInfoBaseTimeZone()` method;
  - creating an initial image for the distributed infobase subordinate node (calling the `CreateInitialImage` method);
  - object numeration update (calling the `RefreshObjectsNumbering()` method);
  - executing the `StandardSettingsStorageManager` object methods, if non-current user settings are operated.

NOTE. The Administration right requires the Data administration right. Verification of an infobase user list for the presence of a user with the Administration right includes users without any separators set only.

For common attributes with enabled data separation, default rights are not set, even if Set rights for attributes and tabular sections by default check box is selected in a role editing form.

7.6.6. Users

When an infobase user is created (the `InfoBaseUser` object), there is a feature that allows you to specify separator values for this user.
The `DataSeparation` property of the `InfoBaseUser` object is used for this purpose. This property stores a structure, in which:

- A structure item key contains a name of a separator (as it is set in Designer).
- A structure item value contains a string representation of a separator value.

If a structure includes a record with any separator name, this means that this separator value is set for a user.
The `DataSeparation` property value determines:

- username uniqueness dimension, if a separator separates authentication;
- user list separation, if a separator separates users;
- initial separator values set-up of a user session for any separator defined in the configuration.

7.6.7. Exchange plans

From an exchange plan point of view, any separator that includes an exchange plan is considered as being in the Independent and Shared mode. The uniqueness of an exchange plan item links is ensured, including items that describe predefined nodes in each data area (the `ThisNode()` method of an exchange plan manager).
Behavior of the system when registering changes depends on the following factors:

- whether an exchange plan is separated or not;
- whether a registered object is separated by all separators of an exchange plan or not;
- whether at least one separator, which includes an exchange plan, is used in the current session or not.

A description of how the automatic registration of changes will be performed depending on the above factors is given in the table below:

Exchange Plan	A metadata object is separated by all exchange plan separators	At least one exchange plan separator is used	Automatic registration
Not separated	–	–	All nodes
Separated	No	No	All nodes
Separated	No	Yes	1
Separated	Yes	No	2
Separated	Yes	Yes	2

In the given table:

1. An exception is raised when performing an automatic completion of a list of recipient nodes. If an exchange plan includes at least one separator that does not include a metadata object with autoregistration enabled for this exchange plan, then, when a database configuration updating, a warning is displayed, saying that a configuration object is not included in components of an exchange plan separator.
2. When automatically filling a list of recipient nodes, a data area is used which is defined by a set of separators, each of which separates both an exchange plan and a metadata object. If not all separators that include both an exchange plan and an object for which changes registration is performed, then for missing separators, separator values from this object are used. If in the process of changes registration it is found that separator values are changed:
   - For data area exchange plan nodes, defined by the "old" combination of separator values, data object deletion is registered.
   - For data area exchange plan nodes, defined by the "new" combination of separator values, data object change is registered.

Configuration changes are registered for all nodes of all exchange plans regardless of these exchange plans separation by any separator (only for exchange plans with selected Distributed infobase check box).

When recording the exchange plan node, it is checked that in the data area described by values of all separators, which include a recorded object, there must be exactly one element of the exchange plan for which the value of the property `ThisNode` is set to `True`. If the condition is not met - an exception is generated.

If recording is performed in the data loading mode (the `DataExchange.Loading` property is set to `True`), then the verification is not performed.

7.6.8. Functional options

If a configuration object that is used to store a functional option value is included in separators, a functional option value is received as follows:

- If at least one independent separator that separates a configuration object used to store a functional option value is not set, then a value of a functional option that is stored in an attribute of the `Boolean` type is equal to the `True` value, and for attributes of other types an exception is invoked.
- If all independent separators are used and at least one separator with the set Independent and shared separation mode is not used, then a value of a functional option that is stored in an attribute of the `Boolean` type is defined by «by OR» addition of all values with undefined separator values, and for attributes of other types an exception is invoked.
- The analysis uses only those separators that contain a configuration object where a functional option value is stored.

7.6.9. Scheduled jobs

A scheduled job can be included in a separator.

A scheduled job starts with separator values that are set for this job using the `DataSeparation` property of the `ScheduledJob` object. Using this property, a set of separators corresponding to the session where a scheduled job starts can also be retrieved (only those separator values that are in the Independent and shared mode are available).

For the `ScheduledJob` object, the `DataSeparation` property value can be changed using general rules, except predefined scheduled jobs for which the `DataSeparation` property can not be changed. A scheduled job can be changed from the session where only those separators that separate this scheduled job are used.

When retrieving scheduled jobs (by UUID or a scheduled jobs list), implicit setting of a filter by separators according to separators values set in the current session is performed. Depending on a separator mode, a different set of scheduled jobs is available:

- A separator in the Independent mode.
  - A separator is used - only those scheduled jobs are used for which separator values are the same as separator values in the current session.
  - A separator is not used - no available scheduled jobs.
- A separator in the Independent and shared mode:
  - A separator is used - only those scheduled jobs are used for which separator values are the same as separator values in the current session.
  - A separator is not used - scheduled jobs with an arbitrary value of this separator and jobs not separated by this separator are available.

If a scheduled job is marked as predefined in Designer, then a list of scheduled jobs with the current combination of separators values and their use is created in the first logging with a new separators combination in the 1C:Enterprise mode. If the separator is not used in a session and a scheduled job is included in this separator, then using is enabled and a default value for a separator type is recorded in a corresponding property of a scheduled job.

TIP. It is mot recommended to create separated scheduled jobs that are performed quite often with a large amount of data areas. It is recommended to create a shared scheduled job that processes different values of separators itself.

Note that there are more severe record limitations on separated scheduled jobs comparing to other objects, as a scheduled job is not just a record in a data base but also a consequence of any action.
A session operating with a specific combination of separators must not have a feature to initiate any action in the areas that are unavailable for it. Therefore, if a separator value is set in a session, and a scheduled job is not separated by this separator, then recording this scheduled job is not allowed in this session, as this allows to perform a code with a separator value different from a separator value in the current session.

7.6.10. Background jobs

A background job starts with values of separators corresponding to the session, where a background job starts. When a background job operating (or after it is ended), separators values can be retrieved using the `DataSeparation` property of the `BackgroundJob` object (values of only those separators are available that are in the Independent and shared mode). Uniqueness of a background job key is checked in the context of a unique set of separators values. So, a single background job with a specific key can exist in each unique data area.
For a separator being in the Independent and shared mode, only one background job with a specific key for an unspecified and a specified value of this common attribute can exist. So, if a background job with a specified value of this separator is started, then this does not allow to start a background job in a session where a separator value is not used. The opposite is also true: a started background job in a session with an unused separator does not allow to start a background job with a specified value of this separator.
For an independent separator, an unused attribute, in terms of uniqueness check mechanism, is used as an individual value in addition to specified values of a used separator. A background job with the same key value can be started simultaneously in a session where a separator is used (and set to any value) and in a session where a separator is not used.
When retrieving background jobs (by the UUID or a list of background jobs), implicit setting of a filter by separators according to separators values set in the

current session is performed. Depending on a separator type, a different set of background jobs is available:

- A separator in the Independent mode: only background jobs started from sessions with the same using and separators values as for the current session are available.
- A separator in the Independent and shared mode: only background jobs started from sessions with the same separators values as for the current session are available. Unused separators are ignored.

7.6.11. Web services

When invoking a web service, separators values can be specified in the same way as for a web client.

7.6.12. Constants

If all separators containing constants are used in a session, then, in a query language, constants can be accessed by the `Constants` table (similar to the 1C:Enterprise version 8.2.13 and later).
Fields corresponding to separators, that are in the Independent and shared mode, exist in the `ConstantValueManager`, the `ConstantValueKey` objects and the `Constant` table.

7.6.13. Registers

A separator being in the Independent and shared mode, serves in a register as a dimension, logically preceded dimensions specified by an application developer.
In the same record set, only records with the same separator value set can be located.

7.6.13.1. Information registers

Separators are included in the main filter of an information register.
Filter items corresponding to separators, which include an information register, can be added to the `Filter` property of the information register record set.
Record uniqueness is checked within the data area corresponding to the current unique combination of separator values.
When calling the `Read()` method for a record set, keep in mind the following features:

- There is no need to set separator values in a record set filter, if the current session uses all separators that contain a register;

- If the separator values in the record set filter are set to values that do not match the current session values, the result of method execution will be a blank set of records;
- If the separators values that are not used in the current session are not set in the record set filter, the result of the method execution will be a set of records with no filtering of the values of these separators.

When calling the `Read()` method of the record manager, keep in mind the following features:

- There is no need to set separator values in the record manager, if the current session uses all separators that contain a register;
- If the separator values in the record manager are set to values that do not match the current session values, the result of method execution will be a blank set of records;
- If the separator values in the record manager are set to values that do not match the current session values, the result of method execution will be a blank record.

When calling the `Select()`, `Get()`, `GetFirst()`, `GetLast()`, `SliceFirst()` and `SliceLast()` methods of the register manager, you should keep in mind the following features:

- If the `Filter` parameter of these methods specifies a value for the separator used in the session, this value must match the separator value. Otherwise, an exception is thrown.
- When calling the `Get()` method, the `Filter` parameter should specify the values of all the separators not used in the current session.

7.6.13.2. Accumulation and accounting registers

Register parameters (totals separation flag, totals usage flag, etc.) are separated and stored for each data area. Totals settings can be received and installed only for a session that uses all separators values that include the register. When deleting separators or modifying their type, the records are merged in the register parameter tables by new combinations of defined separators during the database configuration update.

Recalculate totals operation (the `RecalcTotals()` and `RecalcTotalsForPeriod()` methods) and recalculate present totals (the `RecalcPresentTotals()` method) can be executed with any usage state of separators that include the corresponding register. Totals are recalculated within a data area defined by the current values of used separators.

7.6.13.2.1. Accounting register

When editing accounting registers and charts of accounts, keep in mind the following features:

- Components of independent separators that include charts of accounts and the related accounting register should be identical.
- The accounting register should be included in all separators being in Independent and Shared mode that include a chart of accounts related to this register.

When saving the chart of accounts item, if the accounting settings that affect accounting registers movement are changed, it is checked whether the accounting register separators that are not included in the chart of accounts in the session are used in the Independent and Shared mode.
When saving the accounting register record set, it is checked that all accounts used in the record set have separator values that are identical to the separator values in the record set records.

7.6.13.2.2. Accumulation register aggregates

Aggregate settings are separable and stored for every data area.
When operating with aggregates, keep in mind the following features:

- The following methods can be called only in the session where all separators that include the register are used:
  - `GetAggregatesMode()`,
  - `GetAggregatesUsing()`,
  - `GetAgregates()`,
  - `SetAggregatesMode()`,
  - `SetAggregatesUsing()`.
- The following methods can be called in any case:
  - `DetermineOptimalAggregates()`. Executed for the data area defined by used separators values.
  - `RebuildAggregatesUsing()`. If all the separators that include a register are used in the session, the method is executed only for a specific data area. Otherwise, similar settings (components and periodicity) of aggregates usage for separators used and all value combinations of unused separators are set.
  - `UpdateAggregates()`, `ClearAggregates()`, `AggregatesIsFilled()`. If all separators that include the register are used in the session, methods are executed only for a specified area; otherwise - they are executed for every data area within separator values used.

If a separator is in the Independent and Shared mode and it is not used in the session, you can use aggregates, except for the following cases:

- If the totals mode is enabled for some separator values.
- If aggregates usage is disabled for some separator values.
- If different minimal periodicity of used aggregates is set for different separator values. If the `RebuildAggregatesUsing()` method is called

in the shared mode, the minimal periodicity of used aggregates will be the same.

- If for different separator values the sets of used aggregates for different separator values do not overlap. If the `RebuildAggregatesUsing()` method is called when the separators are not used, the sets of the used aggregates will be the same.
- For queries in a transaction - if for some separator values the aggregates update process is not completed and it is on the stage of data transfer from a buffer to an aggregate.

7.6.13.3. Calculation Registers

When receiving the schedule data virtual table, the calculation register records and information register job schedule records are mapped not only by calculation register attributes with the filled Link to schedule property, but also by Independent and shared mode separators that are common separators for the information register and the calculation register.

When a basic data virtual table is received, calculation register records are mapped not only by fields passed as virtual table parameters, but also by the Independent and shared mode separators that are common separators for the information register and the calculation register.

When calculating the actual validity period, the validity period between displacing calculation types is separated using the separator value matches (in the Independent and Shared mode).

Also keep in mind the following features:

- when saving the calculation register record set, separators values matching (in the Independent and shared mode) is not controlled in calculation register record and this record calculation type;
- when saving the calculation type, separators values matching (in the Independent and shared mode) is not controlled in the calculation type and all calculation register records with this type of calculation;
- separator value matching (in the Independent and shared mode) is not controlled in the calculation type and those calculation types that are included in the tabular section of displacing, base and leading calculation types.

7.6.13.3.1. Recalculations

Recalculations are automatically included in the same set of separators as the corresponding calculation registers. When saving recalculations, values of separators set for the corresponding calculation register records are used.

7.6.14. Document sequence

Separator in the Independent and shared mode acts as a tool of calculation for a sequence.
A document sequence is separated only in the following cases:

- all documents and registers included in the sequence are separated;
- documents and registers included in the sequence are separated with the same set of separators.

The `Filter` parameter of the `SetBound()` and `Validate()` methods is optional.
If the Sequence filling property of a document included in the sequence is set to the Fill automatically value, then the sequence record common attribute values will be set to values corresponding to the document that generated this record.

7.6.15. Infobase parameters

The infobase time zone is separated by independent separators. If not all separators are used, an attempt to set or get the infobase time zone raises an exception. Otherwise, the timezone value broken down by separator values is used.
Other sessions are notified of the infobase time zone change within 20 seconds after the changes are made. The exclusive infobase access is not required if the time zone is set in a session that uses at least one separator.
The `SetInfoBaseTimeZone()` method requires an exclusive area lock if the current session does not use a separator.

7.6.16. Session list

The session list will include sessions where separator values match the values of the current session (from which the list is received using the `GetInfoBaseSessions()` method). In this session list, the usernames (and computer names) will be unavailable if the user does not belong to the data area available for the current session. For example, the `Subscriber` separator (of the `Number` type) exists in an infobase. For `Ivanov` user, the separator value is set to 5. If, while running, the application changes the Subscriber separator value to any other value, usernames will be unavailable in the session list.

7.6.17. Connection list

The list of connections (acquired using the `GetInfoBaseConnections()` method) will be non-empty only if no separator is used in the current session.

7.6.18. Event log

7.6.18.1. Visual representation

If the event log is opened in a session where at least one unused separator exists, the Session data separation field will be added to the following forms:

- event log list form,
- event log list filter setting form,
- event log record form.

Separator names representation is defined as follows:

- For an existing separator - its representation.
- For a renamed separator, i.e. a separator which name has been changed after related records were added to the log - the current separator representation.
- For a deleted separator - the last name of the deleted separator supplemented with spaces before the non-first capital letters. In this case, upper case letters are converted to lower case letters (except for the first letter in the separator name).

7.6.18.2. Filters

When generating the event log (to view the log or export it using the `UnloadEventLog()` method), filtering by separators that are used in the current session is always implemented. This filtering (forced) cannot be disabled or changed. If the event log is generated in the session where the separator is conditionally disabled, filtering by this separator is also disabled.

For separators that are not used in the current session, filter values can be set by the user. In this case the forced filtering and user filtering by separator are united using the "AND" statement.

When filter values for the `User` and `Computer` fields are received (using the `GetEventLogFilterValues()` method), the result is generated as follows:

- Separator is used - field values are returned from log records that were docked for the current separation value.
- Separator is not used - all field values docked by the event log are returned.

When event log records are received programmatically (using the `UnloadEventLog()` method), the filter is generated by adding the `DataSessionSeparation` item of the `Structure` type to the filter structure. Every structure item contains a separator name as the key, - and a set of separator values that are used to filter event log records as the value. If the `DataSessionSeparation` structure has multiple items specified, a record will match the filter, if it matches all values set in the structure (filtering by "And").

This set can be set by either using a structure (in this case the records are filtered by one value) or by using a structure array (in this case the filter is specified by a set of values).
The structure that is used for filtering contains the following items:

- `Usage` (`Boolean` type). Default value - `True`. Sets the usage of a common attribute separating data. If `False`, the `Value` item is ignored and can be omitted. Therefore, you can filter log records that do not have this common attribute set.
- `Value` (arbitrary type). A common attribute value by default -`Undefined`.

If a structure array is specified in the `SessionDataSeparation` structure item, an event log record will match this condition, if it matches at least one value (filtering "by OR").
Consider an example.
Two separators are set in an application:

- `Subscriber`, type `Number`;
- `Branch`, type `Number`.

Example 1:

- Type `Structure`.
  - Key `Subscriber`, value Structure:
    - Item 1: key `Usage`, value: True.
    - Item 2: key `Value`, value 1.
  - Key `Branch`, value Structure:
    - Item 1: key `Usage`, value False.

In this case, the event log records that simultaneously fulfill the following conditions are included in the filter:

- The `Subscriber` separator is used and set to 1 value.
- The `Branch` separator is not used.

Example 2:

- Type `Structure`.
  - Key `Subscriber`, value Array:
    - Item 1: `Structure`:
      - Item 1: key `Usage`, value: True.
      - Item 2: key `Value`, value 1.
    - Item 2: `Structure`:
      - Item 1: key `Usage`, value: True.
      - Item 2: key `Value`, value 5.
  - Key `Branch`, value Structure:
    - Item 1: key `Usage`, value: True.
    - Item 2: key `Value`, value 8215.

In this case, the event log records that simultaneously fulfill the following conditions are included in the filter:

- The `Subscriber` separator is used and can take `1` or `5` values.
- The `Branch` separator is used and can take `8215` value.

7.6.18.3. Getting event log records

When exporting the event log (to a value table or an XML format document) the following columns are added to the corresponding data structure:

- `SessionDataSeparation`, `Structure` type, where the structure key - a separator name, a value- a separator value;
- `SessionDataSeparationRepresentation`, type `Array`, consisting of strings of `<Separator representation> type: <Separator value representation>`.

These columns are added only if there in the current session is at least one separator, that is unused in the current session. A data in the `SessionDataSeparation` and the `SessionDataSeparationRepresentation` columns contains information about unused separators only.

For example, two separators exist in an application:

- `Subscriber`,
- `Branch`.

If both separators are used in the session, then an exported data will not have the `SessionDataSeparation` and the `SessionDataSeparationRepresentation` columns. If only the `Subscriber` separator is not used in the session, the `SessionDataSeparation` and the `SessionDataSeparationRepresentation` columns data will contain information about the `Subscriber` separator only and there will be no information about the `Branch` separator.

Separator name representation is defined as follows:

- For an existing separator - its representation.
- For a renamed separator, i.e. a separator which name has been changed after related records were added to the log - the current separator representation.
- For a deleted separator - the last name of the deleted separator supplemented with spaces before the non-first capital letters. In this case, upper case letters are converted to lower case letters (except for the first letter in the separator name).

To get a set of acceptable values of common attributes separating data, the `GetEventLogFilterValues()` method is used.

Specify the `SessionDataSeparation` string as the first method parameter to get a set of common attributes separating data.

In the structure returned by a method, a set of common attributes is saved into an item with the `SessionDataSeparation` key. An item value is a match, where:

- `Key` - a common attribute name,
- `Value` - a common attribute representation.

To get a set of separator values, you need to specify the `SessionDataSeparation.<Common attribute name>` string in the first method parameter, where the `<Common attribute name>` - name of the common attribute (see above).

In a structure returned by the method, a set of acceptable values of the common attributes is saved to an item with the `SessionDataSeparation` key. An item value is a match, where:

- `Key` - common attribute name.
- `Value` - `Match` type value, where:
  - `Key` - common attribute value;
  - `Value` - common attribute value representation.

7.6.18.4. Miscellaneous

If the user is unavailable in the current session (belongs to another data area, is deleted, etc.), when viewing the log, exporting it and in the filter setting dialog box this user is shown as `Username <not found>` string.

7.6.19. Settings and favorites

Settings are stored in a unique combination of separator values and usernames. Different users with the same name are indistinguishable. A disabled separator represents a particular value, settings for which are also stored.

When separator values are changed in the process of 1C:Enterprise operation, new separator components settings are not automatically read. To do this, close and re-open the form whose settings need to be re-read.

When restructuring an infobase (after the separator components are changed), the following actions are performed:

- When adding a separator, it is considered that all settings previously set in a database, were saved in a situation when a new separator is used and its value is the default value of the added common attribute type.
- If, when deleting a separator, several settings are found that are saved for the same configuration objects and different separator values, only the settings that were saved for the minimum separator value will be saved. A warning will be displayed when this situation occurs in restructuring.

Operations with user favorites are performed according to the above rules.

7.6.20. History

User actions history is stored in a unique combination of separator values and usernames. Different users with the same name are indistinguishable. A disabled separator represents a particular value, with its history also stored.
When restructuring an infobase (after the separator components are changed), the following actions are performed:

- When adding a separator, it is considered that all actions history, that was in the database, was saved in a situation when a new separator is used and its value is the default value of the added common attribute type.
- When a separator is deleted, History is cleared in such way, that every unique value combination of the remaining separators never contains more than 200 history records.

7.6.21. Settings OData standard interface

Odata standard interface settings are stored in a unique combination of separator values. A disabled separator represents a particular value, with OData standard interface settings also stored.
When restructuring an infobase (after the separator components are changed), the OData standard interface settings are deleted and you need to configure them again.

7.6.22. Queries

If an object is included in a separator components and a data query uses the connection with its object table, you can't use the tabular section of the object in the connection condition (`UNTIL` query section).
Keep in mind that the efficiency of queries that are processed slowly (due to a large number of connections) in a system without separators will be even lower, if the system contains over two separators or a separator of the `String` type. Therefore, avoid using a large number of connections in such systems.

7.6.23. Managed transaction locks

When locking from 1C:Enterprise language without specifying any fields, the entire space is locked according to the used separators.
A lock can be set from 1C:Enterprise language with an explicitly specified separators only for the Independent and Shared mode separators, and only for dimension spaces (not record sets) registers.

If an object is recorded with separator values equal to the default values of their types and these separators are used in the current session, lock is set by the separator values specified in the session.
If a lock is executed by a reference value, the space to be locked is defined by these values and the values of independent separators used. Thus, simultaneous locking by a session reference value, where an Independent and Shared mode separator is used and from the other session, where the same separator is not used, is unavailable.
If a session sets an exclusive access to some areas, any attempt to set managed locks for these areas (performed in the same session) are ignored.

7.6.24. Exclusive mode

The `SetExclusiveMode()` method operates with an area described by the separator used in this session. So, a single infobase can simultaneously have multiple areas with an exclusive mode set (with the restrictions described below considered). If a session does not use a single separator, the exclusive mode will be set not for the area, but for the entire database.
Exclusive mode of an area is not compatible with any locks made by another session to the same data area or any of its nested areas, as well as with the exclusive mode that tries to establish a different session for any ambient area. Area `O2` is nested in area `O1` if there is a separator that is used in area `O2`and not used in area `O1`, while separators used in both area `O1` and area `O2`have the same values. Accordingly, the ambient region is defined in the same way.
When moving from area to area, the list of areas with exclusive access set by the current session does not change.
Exclusive mode set for a data area prohibits:

- Creating new sessions for an area and its nested areas;
- Changing a data area from ambient areas;
- Setting exclusive mode for ambient areas.

Herewith, (for the above restrictions) the entire infobase can be considered as an ambient area for any data area.

7.6.25. Deleting marked objects

The `FindMarkedForDeletion()` method finds all objects marked for deletion available in the current session.
When attempting objects deletion (the `DeleteObjects()` method) with reference integrity control, this control is implemented for the objects that are available in the current session according to set separators. There could be situations, when there are references to deleted objects from another objects, unavailable in

the current session (for example, from another data area). In this case, the application developer and the user deleting the objects are responsible for performing such deletion.

TIP. It is recommended to restrict object deletion in applications (for example, deleting shared objects in separated sessions) using configuration tools (access rights, etc).

7.6.26. Using full-text search

In an application using separators, a full-text search index (as a set of service files) is common for all separators values of infobase. An index is updated for all separators values at once, and update can be performed only in a session where all separators are not used. A full-text index update is not supported, if at least one separator is used in a session.
Separated constants (i.e constants included in separators) can not be selected as the Additional full-text search morphology and synonym dictionaries configuration property value.
The search uses all shared objects and separated objects, for which the following conditions are met:

- A separator in the Independent mode:
  - A separator is used - the search includes objects, for which a separator value set for a session is the same as a value set for an object.
  - A separator is not used - separated object are not included in the search.
- A separator in the Independent and shared mode:
  - A separator is used - the search includes objects, for which a separator value set for a session is the same as a value set for an object.
  - A separator is not used - all separated objects are included in the search.

If all application separators are used in a session. then information search time is equivalent to search time in an infobase without separators that is similar to a data area in characteristics. If not all application separators are used in a session, then the search takes more time than in an infobase without data separation that is similar to parameters of a data area in specified separators.

TIP. Scheduled jobs used to update and merge a full-text search index are required to be shared.

Due to the fact that a full-text search index is a single one for the whole infobase (not for each separators combination), the `IndexTrue()` method can return `False`, even if there are no changes in the current session that can be placed in an index. These changed can be performed in sessions with other separators values.
The `UpdateIndex()`, the `ClearIndex()`, the `SetFullTextSearchMode()` methods can be used, only if no separator is used in a session. Otherwise, an exception is invoked.

7.6.27. Configuration extension

Separators defining a data area, where an extension is set, automatically includes all native extension objects. The option to change an adopted separator components is not available.

An extension is attached for a data area that is defined by a unique combination of separators values for which the Configuration extensions separation property is set to the Separate value. So, a unique set of attached configuration extensions can exist in each data area. If the Configuration extensions separation property is set to Do not use for all application separators, then extensions are attached to an infobase without separation by areas. Additional data that can be brought to an infobase by an extension is created in a data area for which an extension is attached.

A set of extensions available for user is defined by a set of used separators values at the moment of operating session start. Changing a set of used separators during session operating does not cause changing a set of available extensions. After changing a set of separators, reading and recording any infobase data are unavailable. The option to read and record "returns" after restoring a set of separators values that was set on start of the current session. If during developing an application user knows that an extension:

- can be operated in a separated infobase,
- configuration extensions, that can extend components of stored data, can be attached to it,
- performing some processing in other data areas is required during operating process,

then, for these operatings, it is recommended to use background jobs that are started using the `ConfigurationExtensions method. ExecuteBackgroundJobWithDatabaseExtensions()`.

7.6.28. Data history

Data history versions are separated according to the same principles that objects storing their history. Data history is accessed in the similar way. In other words, if a data object is unavailable in the current session due to incorrect setting of separators (including the flag of use), then data history of this object is also unavailable in the current session. But, if separators values are changed for a data object (and separators values contain this value) and, thereby, a data area, where an object is located, is changed, then the whole data history also "migrates" to a new data area. It can be said that data history is "bound" to "its" object and always follows it.

Data history settings and metadata history are separated only by independent separators (within each configuration object). Setting data history is allowed only if in the current session all independent separators containing a customizable

configuration object are used. For a configuration object separated only by shared separators, data history settings are the same for all data areas.
A version of metadata is stored in the specific way:

1. History of versioned configuration objects metadata is stored for an infobase as a whole without independent separators. This history is generated on upgrade of infobase configuration.
2. Metadata history is stored for each data area defined by independent separators. This history is generated on each invoking of a settings setting method and on changing infobase configuration, if, before this, settings have been changed in this area.

7.6.29. Collaboration system

If a data area a session operates with is changed during session operating, then:

- A collaboration server connection is reset on the client application side when invoking the `RefreshInterface()` method. If another registration was performed in a collaboration system for a new data area, then a registration of a new data area is used on reconnection.
- On a server side, existing collaboration server connections are closed on changing values of session parameters describing a data area. A new collaboration system server connection is set for a new data area.

7.7. Specifics of client application operating on Linux OS

A client application operating on Linux OS has the following specifics and restrictions:

- The COM technology and the whole list of features related to this technology are not supported:
  - Operating with the `COMObject` object is not supported;
  - The 1C:Enterprise client start in the Automation server mode;
  - Add-ins created by using the COM technology.
- The administration console of the 1C:Enterprise server cluster is not supported. Administration is available by using an administration server (ras) and an administration utility (rac).
- User authentication with OS tools is not supported.
- Operating with the `Mail` object is not supported. Instead of this object, you can use the default mail application, using the `RunApp` function, passing as a parameter the URL generated according to the description of the scheme mailto: (RFC 2368, http://tools.ietf.org/html/rfc2368).

- Placing several application versions on the same computer is not supported as well as the option of automatic "selection" of a system version for operating with an infobase.
- Use of Window metafiles (WMF, EMF) is not supported.
- When developing applications, keep in mind that the `HTMLDocumentField` control (in ordinary mode) and the form field of HTML document field type (in managed mode) are implemented on the basis of the WebKit library, instead of Internet Explorer. Therefore, you should keep in mind that the appearance and the object model of the HTML document (DOM model) that can be accessed via the `HTMLDocumentField.Document` property may differ from these in the client application operating on Windows.

7.8. Features of the client application on macOS

The 1C:Enterprise version for macOS has the following distinctive features and limitations:

- The 32-bit version is missing.
- Operations with file infobase are not supported if it is located on a network resource.
- Operation of 1C:Enterprise server cluster on a computer running macOS is not supported.
- Publishing on a web server is not supported.
- The COM technology and the whole list of features related to this technology are not supported:
  - Operation with the `COMObject` object is not supported;
  - The 1C:Enterprise client start in the Automation server mode;
- Add-ins created by using the COM technology.
- Operations with external data sources on the client application side and in Designer are not supported.
- User authentication with OS tools is not supported.
- The administration console of the 1C:Enterprise server cluster is not supported.
- Operating with the `Mail` object is not supported. Instead of this object, you can use the default mail application, using the `RunApp` function, passing as a parameter the URL generated according to the description of the scheme mailto: (RFC 2368, http://tools.ietf.org/html/rfc2368).
- Use of Window metafiles (WMF, EMF) is not supported.
- Operations with the macOS panel are not supported.

- In key combinations that use the Control key, the Cmd key should be used instead.

7.9. Specifics mobile version of 1C:Enterprise

This section does not fully describe the features and limitations of the mobile version of 1C:Enterprise (both of the mobile platform and the mobile client). Features and changes implemented in the personal computer platform are not automatically transferred to the mobile version of 1C:Enterprise. The new features of the mobile version are reflected in the corresponding records in V8Update.htm file, in 1C:Enterprise mobile version section.
When developing a mobile application, the following configuration objects and system mechanisms are supported:

- Subsystems
- Constants;
- Catalogues;
- Documents;
- Document journal;
- Accumulation registers (except totals split and aggregate mode);
- Information register;
- Data processors;
- Enumerations;
- Access rights (with limitations), roles;
- Infobase users (with limitations);
- Functional options;
- Session parameters;
- Exchange plans (except exchange plans with a Distributed infobase flag set);
- Managed locks;
- Event subscriptions;
- Queries;
- Dynamic list (with limitations);
- Usage of Web services (without the feature of creating a Web service in a mobile application);
- Common pictures;
- Common commands and command groups;
- Common templates;
- Common modules;
- Common forms;
- Data composition system (including the setting dialogs);

- Conditional appearance;
- XDTO mechanism (without the feature of XDTO package creation);
- Full-text search;
- Languages;
- Background jobs;
- Mechanisms for debugging applications.

These mechanisms and configuration objects are not supported:

- Switching interfaces (and the interfaces compatibility mode)
- Accounting mechanisms;
- Periodic calculation mechanisms;
- Mechanisms of business processes and tasks;
- Common attribute mechanism;
- Data access restrictions;
- Settings storages;
- Operations with external data sources;
- Automated testing mechanism;
- Dragging mechanism;
- Standard functions;
- Some managed form items;
- Extended editing in form items;
- User form settings;
- Saving and restoring form data in settings;
- Information panel, user operations history and notifications (`ShowUserNotification()` method);
- Display of the long-term process state (`State()` method);
- Help in the mobile platform;
- Scheduled job mechanism.

When importing a configuration for a mobile application without sources, you should keep in mind the following 1C:Enterprise feature: a module supplied without source text is compiled with the assumption that it will be executed only on a personal computer. If such a module (supplied without source texts) uses methods that have a different number of parameters for a mobile platform and a personal computer platform, then when checking such a module when exporting to a mobile application, an error will be generated.

Some configuration objects and mechanisms have certain operation features:

- IOS does not support the `Execute()` operator and the `Calculate()` function;
- IOS , up to version 5.x (inclusively) only supports password input using the Latin alphabet letters and numbers. iOS 6 version supports the use of national alphabet characters in passwords. To ensure compatibility between different OS versions, it is recommended to set user passwords using numbers and letters of the Latin alphabet only.

- When running a dynamic list, interactive control of dynamic list settings is not supported. Settings management is available only from 1C:Enterprise language.
- On mobile devices running Android OS, the animation in a mobile application may be disabled if the platform evaluates this device as not sufficiently powerful.

When operating running Windows OS (both the phones version and touch devices version) the following is not supported:

- Scanning barcodes using the device camera (not supported only when running Windows OS for personal computers);
- Operations with calendars;
- Operations with contacts;
- Access to the call log;
- Printing from application;
- Receiving and interactive sending of SMS messages;
- Non-interactive phone number dial;
- Operations with client application certificates.

You should also keep in mind, that supported objects do not have all the features that are available in applications for personal computers implemented. In this regard, you should pay attention to the Availability section of the syntax assistant to determine the availability of a method or property in the mobile platform.

7.10. Protection mechanism against dangerous actions

7.10.1. General information

In addition to the program code, that is located in the configuration, third-party program code can be also executed in the application. Third-party program code can get into the system with external reports and processing and extensions. In general, the administrator of an application cannot guarantee that the third-party code does not perform any destructive actions (both such code itself and indirectly, through running external applications, add-ins, COM objects).

To assist the administrator of an application solution, a protection mechanism against dangerous actions is provided. When performing a potentially dangerous action, the system issues a warning containing information about the action being performed and the potential danger of this action. Potentially dangerous actions include:

- Downloading external data processor (report) or configuration extension.
- Downloading or updating a configuration/extension.

- Accessing the following features from an external data processor (report) or extension:
  - Downloading another external data processor (report).
  - Executing an operating system command.
  - User management.
  - Downloading add-ins.

The user can allow or reject a potentially dangerous action. If the action is allowed, in some cases it may be necessary to re-perform the permitted action.

In general, operating with an enabled protection mechanism against dangerous actions looks as follows (for example, with an extension):

- When attempting to add an extension to the infobase, 1C:Enterprise will request the user's permission. If the user allows this action, then the extension is added to the infobase. If the user rejects the action - the extension is not added.
- If protection against dangerous actions is enabled for an extension, then if the extension is performing one of the potentially dangerous actions (the list of actions is given above), 1C:Enterprise prompts the user.
- If the user allows a potentially dangerous action, 1C:Enterprise stores this permission in the context of the user, the external report itself, the processing or extension, and the permitted action. Upon further use, the request is not repeated if the current user has already authorized any actions for this object (external data processor (report) or extension). Permission is stored in the infobase.

  If the user rejects an action, this prohibition is not stored, and the next time the request is executed, the request for permission will be asked again.

  In case a potentially dangerous action is performed in the place of 1C:Enterprise where interactive operations are unavailable (for example, in a background job) and the action being performed has not previously been allowed interactively, then it will be blocked and a record will be made in the event log.

1C:Enterprise identifies the object (external data processor (report) or extension) by the hash of this object. Therefore, 1C:Enterprise will request permission for potentially dangerous actions again for a changed object.

The mechanism of protection from dangerous actions is manifested:

- During interactive downloading of external data processor (a report).
- When interactively attaching an extension.
- When updating a configuration using the configuration update file.
- When calling the `Create()` and `Attach()` methods of the external data processor (reports) manager.
- When attempting to execute an operating system command: `RunApplication()`,`BeginRunningApplication()`,`System()`.

- When attempting to record or delete information about the infobase user: `Record()` or `Delete()` the `InfoBaseUser` object.
- When attempting to download an add-in: `AttachAddIn()`, `BeginAttachingAddIn()`.
- When operating with COM objects: `GetCOMObject()`, `COMObject()` object wizard.

7.10.2. Disabling the mechanism of protection against dangerous actions

In some cases, it is required to disable the protection mechanism against dangerous actions. To do this, you can use the following features:

1. Deselect the Protection from dangerous actions (`UnsafeActionProtection`) check box in the properties of a specific user. This will disable the protection for this user.
2. Use the `UnsafeActionProtection` parameter of the `Create()` and `Attach()` methods of the external data processor (reports) managers. In this case, you can load an external data processor (report) without user requests.
3. Use the `DisableUnsafeActionProtection` parameter of the conf.cfg file. In this case, the protection mechanism against dangerous actions will be disabled for all users of the infobases, the connection strings of which satisfy the specified template.

Disabling protection against dangerous actions is performed according to the following rules (in the specified order):

1. Protection is considered disabled, if the current user has a cleared UnsafeActionProtection check box.
2. Protection is considered disabled, if an infobase connection string matches one of templates specified in the `DisableUnsafeActionProtection` parameter of the conf.cfg file.
3. If external processing (a report) is attached with explicitly disabled protection using the `UnsafeActionProtection` parameter.
4. If protection is explicitly disabled using the `UnsafeActionProtection` property of an extension.

Appendix 8. Features of operating with different DBMS

8.1. General features

- An index can include no more than 16 database fields except a file mode where a limit is 256 fields.
- Operations of a query language strings comparison ignore trailing spaces, unlike the 1C:Enterprise language strings comparison where trailing spaces are included in a comparing operation. For example, comparison of the «bb» and «bb » returns `False` in the 1C:Enterprise language (strings are not equal) and `True` in a query language (strings are equal).

8.2. File database

- A database file (.1CD) inside is organized as a set of so called internal files. Each database table corresponds up to four internal files:
  - Table description file - it contains a table description.
  - Data record file - contains data of all table records except data contained in fields of unlimited length.
  - Index file - contains all indexes defined for a table. If no index is defined, then this file is missing.
  - File of values of the unlimited length - stores values of unlimited length contained in table fields.
- Size of each above internal file can not exceed:
  - for a format of version 8.2.14 - 4 GB.
  - for a format of version 8.3.8 with a page of 4 096 bytes size - 4 GB.
  - for a format of version 8.3.8 with a page of 8 192, 16 384, 32 768 and 65 536 bytes size - 6 GB.
- Key length in an index can not exceed 1 920 bytes.

8.3. IBM DB2

- Only a literal (a parameter) or an expression over literals can be the right operand of the `LIKE` (`LIKE`) comparison operation. Mask characters are the «_» - any character and the «%» - a sequence of any characters only.

- The untypified NULL. A type of a column that is a result of the SELECT NULL query - the most general composite type, not a simple type compatible with any another one. That is why this column can not be used in operations where fields of composite types can not be used. For example, SELECT ISNULL(f1 + 1, 1) FROM (SELECT NULL f1) t1 causes an error when operating with DB2, because an operand of the <+> operation can not be a field of a composite type.
- The maximum length of number data - 31 characters (not 38).
- The maximum length of accumulation and accounting registers resources - 25 characters (not 32).
- The maximum size of data of unlimited length is 1 GB.
- The maximum amount of columns in a list of operator selection can not exceed 1012. When defining amount of columns, note that 1C:Enterprise creates several columns in a DBMS table for fields of a composite type.
- The different from other DBMS rules of defining a result accuracy, when performing arithmetical operations.
- Use of a subquery in a query language in the BY section can slow down a query performing. When repeated applying a subquery in the BY section (for example, joining several tables by conditions containing a subquery), there can be the situation when a query is not performed.
- If a query includes a concatenation operation of 10 and more string values including fields and literals, and a result of a concatenation is included in comparison, then there can be the error DBMS Error: SQL0401N. Operands specified for the '...' operation have incompatible data types.
- Joining in a query is not allowed, if two tables are joining with a condition including comparison of a tabular section fields.
- If the COLLATION_SEQUENCE parameter in the DBMS settings is not equal the UCA500R1_LROOT_AN_CX_EX_FX_HX_NX_S2 value , then there are the following features in an application operating:
  - Operating of the database configuration background update is not supported;
  - Letter case sensitivity in strings comparison. Uppercase and lowercase letters are different in implicit strings comparison performed when processing DISTINCT (DISTINCT), GROUP BY (GROUP BY), UNION (UNION) of a query language. Therefore, query performing has the following features:
    - When using the DISTINCT and the UNION (without ALL) constructions, query result field values of the String type are considered distinct, if they are different in a letter case (when using other DBMS, they are considered the same).
    - When using the ORDER BY construction, values of the String type, that are different in a letter case only, are considered distinct:

    - if a query includes the UNION, the UNION ALL, the DISTINCT constructions;
    - a list of query selection includes the SELECTION expression where nested queries are;
    - the current user has data access restriction, and a query has not the PERMITTED keyword.
  - In other cases, comparison of string data is case-insensitive (values of the String type, that are different in a letter case only, are considered the same):
    - in explicit comparison of fields of the String type;
    - when using the GROUP BY construction;
    - in the GROUP BY construction except the listed cases.

  The COLLATION_SEQUENCE DBMS parameter is set when creating a database. If a database is created by 1C:Enterprise of version 8.1.12 and earlier, then this parameter **is not** equal to the UCA500R1_LROOT_AN_CX_EX_FX_HX_NX_S2 value and there are the features described above. A database created by 1C:Enterprise of version 8.1.13 and later sets this parameter to the UCA500R1_LROOT_AN_CX_EX_FX_HX_NX_S2 value and there are no described restrictions.
- If one of the following situations is detected:
  - The long query performing time;
  - The large amount of hard drive operations when performing a query;
  - The HASH JOIN construction is in a query plan;
  - A query performing gets significantly faster when changing the current query optimization from 5 to 0,

  It is recommended to perform the db2set DB2_OVERRIDE_BPF=5000 command and restart DBMS IBM DB2. This recommendation is not required in DBMS versions IBM DB2 9.7 FixPack 5 and later.
- If a query includes tables of configuration objects contained in a common attribute that is a separator, then use of subqueries in joining tables conditions is not allowed.

8.4. Microsoft SQL Server

- No more than 256 tables can be used in a query (for versions Microsoft SQL Server 2000 and Microsoft SQL Server 2005).
- One record in a table can not be more than 8 060 bytes;
- One index record can not be more than 900 bytes. If an index record is more than 900 bytes, then an error is occurred not in the moment of a database

configuration update, but during an attempt to enter a value that will cause an index to contain a record exceeding the limit.

- When using the Microsoft SQL Server 2000, when grouping by an expression (not by a field), when using an aggregate function, not included in the `SELECT` section, in the `HAVING` section, the DBMS error can occur. When creating applications that are oriented to the Microsoft SQL Server 2000, it is recommended to include expressions used in the `HAVING` section to the `SELECT` section. There is no this problem in DBMS Microsoft SQL Server 2005 and later.
- When an application is intended for operating in the Microsoft SQL Server 2000, it is not recommended to use data operating functions in the `Group By` query section, because this can lead to incorrect results. Instead of using expressions containing functions, you can use grouping by all table fields an expression depends on.
- If the `Kazakhstan` country is specified in regional settings of an infobase, then Kazakh sorting order is not used in a database. The `Cyrillic_Generic_CI_AS` sorting order is used. Sorting is similar to the one that is performed when specifying the `Russia` country in regional settings of an infobase.

8.5. Oracle Database

- One index record can not exceed approximately 80% of a block size:
  - For a block of 4K size - 3 318 bytes;
  - For a block of 8K size - 6 398 bytes;
  - For a block of 16K size - 12 958 bytes.
- In the automatic mode, Oracle Database uses table locks. This means that a transaction, that locks one record in a table, blocks this entire table, and this can prevent competing transactions from operating with data of this table. It is recommended that the application development is carried out in a managed lock mode in order to maximize the use of the DBMS capabilities. Use of an application in the automatic mode is recommended for compatibility with the previous versions of applications and is not recommended for actual operation.
- The feature of using a nested query with the `TOP` modifier inside the `IN` operator is missing, if there are accesses to external query fields inside a nested query. This restriction does not apply to cases, when the `IN` operator is used to check presence of strings, i.e. there is `TOP 1` in a subquery, and expressions to the left of `IN` and in a selection list are equal constants. For example, in a query:

```
SELECT
  Field
  FROM Table AS T
  WHERE 1 IN (
    SELECT FIRST FIVE
    1
    FROM SubqueryTable AS ST
    WHERE ST.VerificationField = T.Attribute
  )
```

The above restriction is not relevant as expressions to the left of the IN and in a selection list are equal constants.

- When sorting ascending, fields with the NULL values are the last in selection.
- Use of a subquery in a query language in the BY section can slow down a query performing. When repeated applying a subquery in the BY section (for example, joining several tables by conditions containing a subquery), there can be the situation when a query is not performed.
- A record is optimized from the first (in the table order) attribute of the ValuesStorage type, and if there is no such ones, then from the first (in the table order) attribute of the String of unlimited length type.
- Note that when operating with Oracle Database, statistics relevance affects performance. It is recommended to keep DBMS statistics relevant (DBMS does it itself by default, but you can start statistics collecting manually using the dbms_stats.gather_schema_stats procedure). Also, it makes sense to perform statistics collecting after loading an infobase from an infobase dumping file (*.dt files). After a database server restart, statistics collecting also can affect performance.
- If registers have more than three dimensions of the String type, then a tablespace for indexes with large size of a key can be required during a database configuration update and also during loading an infobase. 1C:Enterprise does not create such tablespace, but it can use the tablespace named V81C_INDEX_BIG.

  Tablespace creation must be performed by a database administrator with the CREATE TABLESPACE command specifying a block length in the BLOCKSIZE parameter. To create a tablespace with a block length larger than a database block length, additional setting of a database instance might be required. For example, a tablespace with a block size of 16384 bytes can be created by setting the DB_16K_CACHE_SIZE parameter using the ALTER SYSTEM command.
- In queries using the LIKE comparison operation, brackets in a template are interpreted as specific characters only if they are used in a text literal, not in an expression.

In the next example brackets have a specific meaning - represent a lot of characters:

```
SELECT * Catalog.Goods. WHERE Article LIKE "123[АБ]%"
```

In the next example brackets that can be located in the `Template` variable do not have a specific character:

```
SELECT * Catalog.Goods. WHERE Article LIKE Template+"%"
```

8.6. PostgreSQL

- In the automatic mode, the PostgreSQL uses table docks. This means that a transaction, that locks one record in a table, blocks this entire table, and this can prevent competing transactions from operating with data of this table. It is recommended that the application development is carried out in a managed lock mode in order to maximize the use of the DBMS capabilities. Use of an application in the automatic mode is recommended for compatibility with the previous versions of applications and is not recommended for actual operation.
- When sorting ascending, fields with the `NULL` values are the last in selection, and when sorting descending - the first.
- When performing operations related to intensive deleting and adding records to database tables (for example, documents reposting), performance degradation of a performed operation can occur. To restore original performance, it is recommended to perform the `REINDEX` (or the `VACUUM`) operation for intensively modified database tables regularly. Frequency of performing specified operations depends on intensity of operating with these tables.
- The PostgreSQL performance is significantly affected by performance of hard drive system, because the `fsync` parameter is enabled by default. This means that data is immediately replaced from OS cash to the hard drive when performing the `COMMIT` operation; this guarantees consistency on a possible hardware failure. The reverse side of this is reduction of hard drive recording operation performance, because in this case, features of deferred OS data recording are not used.
- Performance can be increased by using multi-hard-drive RAID arrays based on cashed RAID controllers with non-volatile cash-memory, and by using uninterruptible power supply (UPS). In this case, the devices listed abode are responsible for providing a data consistency on a hardware failure, therefore, disabling the `fsync` parameter and increasing hard drive recording operation performance are available. Note that increasing

amount of hard drives in the RAID array and amount of the RAID controller cash allows to compensate performance reduction caused by enabling the `fsync` parameter.

- There are some differences in operating of the date and time functions. In Microsoft SQL, the `select datediff(hour, datetime(2006, 10, 29, 0, 0, 0), datetime(2006, 10, 30, 0, 0, 0))` returns 24, but in the PostgreSQL - 25 because there was the daylight saving transition between these two dates. The similar situation about the daylight saving time occurs for the `DATEADD` function.
- It is not recommended to use the `FULL EXTERNAL CONNECTION` in queries as this construction is implemented not efficient enough in DBMS. In the most cases, using this construction is not required, you can rewrite an original query.
- Using both the `FULL EXTERNAL CONNECTION` and an access to tabular sections in a list of selection fields at the same time is not allowed.
- In queries using the `LIKE` comparison operation, brackets in a template are interpreted as specific characters only if they are used in a text literal, not in an expression.
  In the next example brackets have a specific meaning - represent a lot of characters:

```
SELECT * Catalog.Goods. WHERE Article LIKE "123[АБ]%"
```

  In the next example, brackets that can be located in the `Template` column have no specific meaning:

```
SELECT * Catalog.Goods. WHERE Article LIKE Template+"%"
```

Appendix 9. XBase operations

This article explains terms used in the description of the 1C:Enterprise language tools for operating with databases of - the XBAse object.

9.1. Fields and records

The good example of a database is a telephone directory of an organization. It contains surnames, telephone numbers and room numbers of each organization employee. Each directory string matches a record, and each column - a field. Each field has a name and information characteristics for which storing it is intended: a type, a length, an accuracy. A field content for a specific record is named a field value. For example, a telephone directory can be organized as a table with the Surname Name Middle Name, Room number, Telephone number columns (fields). Each string (a record) contains information about a subscriber.

9.2. A table, a table structure, a database file

The entire directory in database terms is named a table. Components of fields included in a table define a table structure, and components of records included in a table - its content. Each record in a table contains the same field set as the entire table, therefore, sometimes the **record structure** term is used. This concept is adequate to a table structure, although use of the first term seems to be more correct as a table has a structure regardless of whether it has at least one record.
Implementation of databases of the BDF format implies that each table is stored in a separate file. Therefore, in the future we will use the «database file» or the «DB file» term meaning a database table.

9.3. Indexes, expressions of an index and a filter, an index file

To organize ordering of a database file content and the search in it by a value of a field or several ones, the index feature is used. Its use can be compared with sorting of card files by a specific flag (set of flags). However, unlike card files, a database file can have several indexes at once and, respectively, be ordered by several flags at the same time. Each index has a name, a uniqueness flag, an index expression and a filter. An index name is used to identify an index. An index expression and a filter are expressions written in a specific language, which value calculation for each record allows to define its location when ordering and whether it is required to place it to an ordered list (an index can contain a reference only to table records matching a filter expression, not all table records). A unique index (with enabled uniqueness flag) allows to have links to records only with a different index expression value in an index.
Indexes are stored in an index file. An index file can contain information about no more than one index.

9.4. Recording of changes to a database

Each object is a data structure located in a computer memory. And changing an object properties content does not cause an instant change in a database files. In the enabled autosaving mode, recording of an object content to a database files is performed on positioning changing (switching to the next record, the search by a key etc.). In the disabled autosaving mode, changes recording is performed only on invoking the corresponding object method.

9.5. Operating with index files

Note that the `XBase` object can be associated with no more than one index file. All database changes made in a session of operating with one index file do not affect the rest. Therefore, it is not recommended to have more than one index file for a database. Otherwise, it is required to perform reindexing (an index file content update) after each opening a database with an index file different from the one that was opened in the previous session of operating with a base.

9.6. Record removing

Removing a record from a database does not lead to its physical destruction in a hard drive. In this case, the deletion mark is performed in a specific service record field unavailable for general tools. Positioning is not performed in a record marked as deleted, if a specific view deleted records mode is not enabled. There are a property controlling a specific view mode, and also a set of methods for defining whether a positioned mark is deleted and restoring a deleted mark.
A base compression method causes physical destruction of records marked as deleted. A base clearing method causes physical destruction of all records. Restoring deleted records becomes impossible after using these methods.

9.7. Creating a database, index, an index file

Besides operating with existing databases, the XBase object has a set of methods that allow to create a new database of arbitrary structure, new indexes and a new index file. Note that if use of methods changing a database structure is available only for objects not related to the existing database (i.e for databases being created), then creating new indexes and an index file is available for databases being created as well as for existing and open ones.

9.8. Restrictions

The main purpose of the `XBase` objects - organization of information export-import in/from external files of the DBF format.
The `XBase` objects do not support fields of the `memo` type. The `XBase` objects support an exclusive access to files only. The XBase objects support index files of the `CDX` format. However, using external programs (for example, `FoxBase`) of index files created by the XBase objects as well as using objects of index files created by external programs is not recommended due to possible incompatibility of versions.

Appendix 10. Specifics of using some features

HTML document operations

- The <TABLE> item can not be added to the <P> item as a child item. So, the <P><TABLE></TABLE></P> construction becomes the <P></P><TABLE></TABLE> after processing by 1C:Enterprise. The rules of the <P> item processing are specified in a HTML standard (http://www.w3.org/TR/html401/struct/text.html), in the article 9.3.1 Paragraphs: the P element.

Forms

- When executing a program code located in an idle handler, the Ctrl + Break combination is not processed, and the screen redrawing is not performed. Also, it is not recommended to place a program code, that will prevent a user from comfortable operating, in an idle handler.
- The server is notified about the closure of the form (as well as unlocking the object if it is locked) in the following cases:
  - closing a form if an object was locked in an editing form;
  - receiving a form from a server using the `OpenForm()`, `GetForm()` methods;
  - calling server (context and out-of-context) form methods;
  - closing 20 forms;
  - 1C:Enterprise is in waiting status for more than 20 seconds.
  - When the following dynamic list commands are executed:
    - deleting an item;
    - marking an item for deletion;
    - posting or canceling of posting of a document;
    - moving a catalog item to another group.
- When using the `OpenValue()` method or clicking the `Open` button (a magnifier) of the input field, if an application is performed in thick client operating in the managed mode, a list form is opened when the following conditions are met at the same time:
  - For a configuration object, the In list editing mode is selected (for directories, exchange plans, charts of characteristic types, charts of

accounts, charts of calculation types, independent information registers, business processes and tasks);

- There in an application is an ordinary form of a list selected as the main or an auxiliary form;
- Use of ordinary forms in a managed application is available in an application;
- A managed form of a list or an object selected as the main one or a auxiliary one is missing.

File operations

- When operating with files in the local file system on Windows OS, note that the length of the full path to a file can not exceed 260 characters.

Spreadsheet documents

- If a multiline text aligned to horizontal and to the center and going beyond the cell borders is located in a spreadsheet document cell, then this text is cut by the left and the right borders of a cell when exporting this spreadsheet document to the Microsoft Excel format. To avoid this behavior, it is required either to create a merged cell so that the entire text fits it or to place each multiline text string to its own spreadsheet document string. Herewith, it is not recommended to make a separate format for a string containing a cell with a multiline text.
- If the `Placement` property is set to the `Auto` value in a not blank cell of a spreadsheet document, then cells are merged to the next filled cell or to the end of a spreadsheet document by horizontal.
- If a spreadsheet document cell contains a text going beyond the cell border by horizontal, then cells are merged to the next filled cell or to the end of a spreadsheet document by horizontal when exporting this spreadsheet document to the HTML format.
- When exporting a spreadsheet document to the PDF format, a font of characters, missing in the current font of a spreadsheet document, is selected.
  In the client application for Windows OS, the Sans Serif font is used. To do this, the HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\FontLink\SystemLink 1C:Enterprise registry record, where the record of the Microsoft Sans Serif font is searched, is analyzed. If the required characters are missing in the specified font, then a font is selected according to the UNICODE character range:

Character range	**Used font**
LATIN	Microsoft Sans Serif

Character range	Used font
ARABIC	Microsoft Sans Serif
HEBREW	Microsoft Sans Serif
SYRIAC	Estrangelo Edessa
THAANA	MV Boli
GREEK	Microsoft Sans Serif
CYRILLIC	Microsoft Sans Serif
ARMENIAN	Sylfaen
GEORGIAN	Sylfaen
SINHALA	Iskoola Pota
TIBETAN	Microsoft Himalaya
PHAGS_PA	Microsoft Phags Pa
THAI	Microsoft Sans Serif
LAO	DokChampa
DEVANAGARI	Mangal
BENGALI	Vrinda
GURMUKHI	Raavi
GUJARATI	Shruti
ORIYA	Kalinga
TAMIL	Latha
TELUGU	Gautami
KANNADA	Tunga
MALAYALAM	Kartika
TAI_LE	Microsoft Tai Le
NEW_TAI_LUE	Microsoft New Tai Lue
KHMER	DaunPenh
YI	Microsoft Yi Baiti
ETHIOPIC	Nyala
MONGOLIAN	Mongolian Baiti
TIFINAGH	TIFINAGH
NKO	TIFINAGH
VAI	TIFINAGH
CHEROKEE	Plantagenet Cherokee
CANADIAN_ABORIGINAL	Euphemia
OGHAM	Segoe UI Symbol
RUNIC	Segoe UI Symbol
BRAILLE	Segoe UI Symbol
DESERET,	Segoe UI Symbol
OSMANYA	kEbrima

In a client application for Linux OS as well as for a web client, a tone matching the required characteristics is selected from a list of fonts available in 1C:Enterprise.

- Exporting a spreadsheet document to the XLS (Microsoft Excel 97), the Microsoft Office OpenXML (Microsoft Excel 2007- 2010) and the ODS (OpenDocument) formats has the following restrictions on exporting pictures located in a spreadsheet document cells:
  - If a spreadsheet document cell contains a text and a picture, and the Text position relative to the picture property is set to the To the left or to the To the right value, then text in other cells of the column is cut to the cell size.
  - If a spreadsheet document cell contains a text and a picture, and the Text position relative to the picture property is set to the At the top or to the At the bottom value, then text in other cells of the string is cut to the cell size.
  - If the Text position relative to the picture cell property is set to the To the left or to the To the right value and a cell text «goes over» a cell picture, then, in a resulting document, a picture of the «text» type is used to display a text over a picture.
  - If the Text position relative to the picture cell property is set to the Over value, then a picture of the «text» type is used to display a text over a picture, in this case, rotation of a picture of the «text» type is used to specify the text orientation.
  - In the situations where a text is horizontally covered by a picture (for example, a picture is aligned to the left, a text is placed to the left of a picture), the text orientation is not saved.
- When using several string formats in a spreadsheet document, page splitting is performed by columns from the first string of a spreadsheet document. For these columns, splitting is performed so that the entire column fits a page. Columns of other string formats can fit a page not fully.
- When exporting a spreadsheet document to the XLS (Microsoft Excel 97) and the Microsoft Office OpenXML (Microsoft Excel 2007- 2010) formats, additional cell merges are created for the cells containing a multiline text aligned to the cell center and not fitting a cell. Merging is performed horizontally, to both sides of the original cell, on equal distances sufficient to place all the text of the original cell. Columns containing left and right borders of a rectangle filled with text are split in two.

  At the same time, if a spreadsheet document cell contains a multiline text aligned to the center and not fitting a cell, but there is not enough

space to the left and to the right of a cell to display a cell text fully, then an appearance of these cells in a resulting document may not match an appearance of the same cells in a spreadsheet document.

- Specifics of exporting a spreadsheet document containing the SVG:
  - When exporting a spreadsheet document to the HTML, the HTML4 formats and to the documents of the Microsoft Office OpenXML format, a picture is converted to a point image of the size depending on the sizes in a spreadsheet document, and saved as PNG.
  - When exporting to the OpenDocument Spreadsheet format, if a picture contains filters, then an image is saved the same way as the Microsoft Office OpenXML format, otherwise, an image is saved as SVG.
  - When exporting to the HTML5, a SVG image is pasted directly into a document.
  - When exporting to the PDF, a SVG picture is exported in the vector format using the PDF format features. An item with filters is converted to a point picture and saved in this type. By default, a point picture is generated for 600 dpi resolution. Exporting large pictures with filters can be time-consuming.
- Distinctive features of export of a spreadsheet document with formatted strings in its headers:
  - When exporting to Word:
    - Line breaks are ignored.
    - Images are ignored.
  - When exporting to Excel:
    - Text and header background colors are ignored.
    - The aggregate length (all three headers in a spreadsheet document are taken into consideration) of a header cannot exceed 255 characters, including special information as to the text format (font names, colors etc). If the said limit is exceeded, - data in excess thereof will be deleted.
    - So far as XLS format is concerned, no images can be exported.
    - So far as XLSX format is concerned, only the first image available in a header will be exported. The remaining images are replaced with a space character.
  - When you export into Open Document Spreadsheet format, images and background colors are ignored.

Mail operations

- When operating on Windows OS, disabling the use of the OpenSSL protocol in operating with the InternetMail object in 1C:Enterprise versions 8.3.3-8.3.5, and for later versions in the compatibility mode with version 8.3.5 and earlier, is performed with 1C:Enterprise registry editing. To do this, the `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\SSL 3.0\Client\Enabled` key should be set to the `0` value. If the key is missing, it should be created. The key type - `DWORD`.

Appendix 11. The rules of automatic form items names generation

When accessing a managed form items the context menu and the automatic command bar as the named `Items` property items, you can use the `CommandBar`, the `ContextMenu` and the `Form` standard prefixes in both 1C:Enterprise language options, regardless of what the selected option was at the moment of a form creation.

TIP. If it is required to generate names of a form items manually (for example, when programmatic creation), it is recommended to do this according to the same rules as 1C:Enterprise does this automatically. Also, it is not recommended to change a form items names that are generated by 1C:Enterprise when developing a form (for example, when creating a form item with an attribute dragging).

In the cases when an item name is generated from the path to the related data, generation is performed as follows: a name is generated as the path to the related data with the «.» excluded characters.

1. Items related to data (fields and tables), also related to the main form attribute. A name is generated from the full path to data. If related data is attached to the main attribute, then the main attribute name is removed from a name.

Example:

The `Goods` tabular section of the main attribute. A table name: `Goods`. One more object located in the `AdditionalObject` attribute is edited in a form. The table displaying the `Goods` tabular section of this object: `GoodsAdditionalObject`.

2. Items related to data, located in a table (columns) or related to the current tables data. Except tables related to the main attribute. A name is generated according to the following rule `<ItemNameTable><TableAttributeDataPath - TableDataPath>`.

Example:

The `Quantity` attribute of the `Goods` tabular section of the main form attribute: `GoodsQuantity`.

The `Quantity` attribute of the `Goods` tabular section of not the main `AdditionalObject` form attribute: `AdditionalObjectGoodsQuantity`.

3. Items related to data, located in a table (columns) or related to the current tables data. For tables related to the main attribute. If a table displays the main attribute directly, then names are generated similar to the described in point 1. If a table is related to an attribute of the main form attribute, then names are generated similar to the described in point 2.

Example:

There in the `List` table related to a dynamic list that is the main form attribute, is a column displaying an attribute of the `Date` dynamic list. A column name: `ListDate`.

A table related to a tabular section of objects displayed by a dynamic list is placed on a form. A tabular section is named as `Goods`, and a tabular section attribute is named as `Quantity`. A name of a table form item: `Goods`. A name of this form item's column: `GoodsQuantity`.

4. Predefined items (context menus and automatic command bars): `<ParentItemName>ContextMenu` and `<ParentItemName>CommandBar`.

Example:

Command bar form: `FormCommandBar`.

The context menu of the `Quantity` column related to the `Goods` tabular section of the main form attribute: `GoodsQuanityContextMenu`.

5. A name of a group related to an item that is a source of commands: `<SourceItemName>Actions`.
6. A name of a button located in a form (not in command bars or context menus): `<CommandName>`. A prefix as an item name is not added for commands generated by an item.

Example:

The `Write` form command: `Write`.

The `Change` command of the `Goods` tabular section: `Change`.

7. A name of a button located in an automatic command bar with the enabled AutoFill property: `<CommandBarOwnerItemName><CommandName>`. A prefix as an item name is not added for commands generated by an item.

Example:

The form command `Write`: `FormWrite`.

8. A button name of the `CreateByParameter` and the `ShowInList` commands (if there are more than one of these commands) that is placed in the command bar with the enabled AutoFill property: `<Form><CommandName><ConfigurationObjectName>`.
9. A name of a button that is placed in the context menu with the enabled AutoFill property: `<ContextMenuOwnerItemName>ContextMenu<CommandName>`. A prefix as an item name is not added for commands generated by an item.

Example:

The `Select` command of the field context menu `Date`: `DateContextMenuSelect`.

10. A name of a group that is placed in the command bar or the context menu without a native command source: `<ParentItemCommandSourceName><GroupName>`. Group names do not affect child item names.
11. A name of a group that is placed in a table not related to the main form attribute: `<FormTableName><GroupName>`. Group names do not affect child item names.

Example:

The `PriceAndQuantity` group in the table `Table`: `GoodsPriceAndQuantity`.

12. Names of buttons and groups that are placed in the non automatic command bars with a source - similar to point 7. Names of buttons and groups that are placed in the non automatic command bars without a command source: names are not modified.
13. A name of a group of buttons that are placed in the command bars or the context menus with a command source: `<CommandBarParentItemName><GroupName>`. Each name is generated according to the rules described above. A command source does not affect a name generation. They become parent groups for child items according to the rules described above. The `CommandBar` prefix is not added.

Example:

The `Form` button group with a data source is added into the context menu of the `ServiceQuantity` column. Standard group name: `ServiceQuantityContextMenuActionForm`.

Appendix 12. A description of entities that are provided via the standard OData interface

In a property description, providing description is not additionally given.

12.1. Constants

Entity name	Constant_<Name>
Entities set name	Constant_<Name>

Properties components:

Name	Type	Comment
Value	Matches a value type	A constant value.
<Common attributes>	Matched a separator type	Except for common attributes that are independent separators.
SurrogateKey	Edm.Int16	A surrogate key if there are no separators. Is always equal to 0.

Behavior particularities:

- Deleting is not supported (the DELETE query);
- Creating is supported (the POST query) with specifics.
  If a constant value has the Undefined value in an infobase, then setting a constant value is performed. If a constant value has a value different from Undefined in an infobase - an error is generated.
- Accessing «via a point» is not supported.

12.2. Exchange plans

Entity name	ExchangePlan_<Name>
Entities set name	ExchangePlan_<Name>

Properties components:

Name	Type	Comment
Code	Edm.String Edm.Int32	Code.
Description	Edm.String	Name.
Ref	Reference type	Reference.
DeletionMark	Edm.Boolean	Deletion mark.
Version	Edm.String	Data verison.
ReceivedNo	Edm.Int32	Number of the last received message.
SentNo	Edm.Int32	Number of the last sent message.
<Attributes>	Corresponds to attribute type	
<Common attributes>	Corresponds to attribute type	Except for common attributes that are independent separators.
<Tabular sections>	Collection (CompositeType)	Each tabular section acts as an independent entity with the ExchangePlan_<ObjectName>_<TabularSectionName> name.

12.2.1. Tabular section of exchange plan

Entity name	ExchangePlan_<Object name>_<Tabular section name>_RecordType
Entities set name	ExchangePlan_<Object name>_<Tabular section name>_RecordType

Properties components:

Name	Type	Comment
Ref	Reference type	Reference.
LineNumber	Edm.Int64	Line number.

Name	Type	Comment
<Attributes>	Corresponds to attribute type	
<Common attributes>	Corresponds to attribute type	Except for common attributes that are independent separators.

12.2.2. Tabular section row

Composite type name: ExchangePlan_<Object name>_<Tabular section name>_RowType.

Properties components:

Name	Type	Comment
LineNumber	Edm.Int64	Number of tabular section string
<Attributes>	Corresponds to attribute type	

12.3. Catalogs

Entity name	Catalog_<Name>
Entities set name	Catalog_<Name>

Properties components:

Name	Type	Comment
Ref	Reference type	Reference.
Code	Edm.String, Edm.Int32	If there is a directory code.
Description	Edm.String	If there is a directory name.
IsFolder	Edm.Boolean	For an hierarchical directory.
Owner	Link or composite type	Owner for a subordinate directory.
Parent	Reference type	Parent for an hierarchical directory.
Version	Edm.String	Data verison.
Predefined	Edm.Boolean	Predefined item flag. Read-only.
Predefined-DataName	Edm.String	Predefined item name.

Name	Type	Comment
<Attributes>	Corresponds to attribute type	
<Common attributes>	Corresponds to attribute type	Except for common attributes that are independent separators.
<Tabular sections>	Collection (CompositeType)	Each tabular section acts as an independent entity with the Catalog_<ObjectName>_<TabularSectionName> name.

12.3.1. Directory tabular section

Entity name	Catalog_<Object name>_<Tabular section name>_RecordType
Entities set name	Catalog_<Object name>_<Tabular section name>_RecordType

Properties components:

Name	Type	Comment
Ref	Reference type	Reference.
LineNumber	Edm.Int64	Line number.
<Attributes>	Corresponds to attribute type	
<Common attributes>	Corresponds to attribute type	Except for common attributes that are independent separators.

12.3.2. Tabular section row

Composite type name: Catalog_<Object name>_<Tabular section name>_RowType.

Properties components:

Name	Type	Comment
LineNumber	Edm.Int64	Row number in the tabular section.
<Attributes>	Corresponds to attribute type	

12.4. Documents

Entity name	Document_<Name>
Entities set name	Document_<Name>

Properties components:

Name	Type	Comment
Ref	Reference type	Reference.
Number	`Edm.String`, `Edm.Int32`	Number of document depending on set number type.
Date	`Edm.Date`	If enabled.
DeletionMark	`Edm.Boolean`	Deletion mark.
Version	`Edm.String`	Data verison.
Posted	`Edm.Boolean`	Document posting flag.
<Attributes>	Corresponds to attribute type	
<Common attributes>	Corresponds to attribute type	Except for common attributes that are independent separators.
<Tabular sections>	`Collection (CompositeType)`	Each tabular section acts as an independent entity with the Document_<ObjectName>_<TabularSectionName> name.

12.4.1. Methods

Post

Description:

Performs document posting.

Parameters:

PostingModeOperational	*Edm.Boolean*

Defines whether it is required to post the document in real-time mode. If such posting is required, the property value should take the true value.
Default value: false.

Unpost

Description:

Cancels document posting.

Parameters:

`PostingModeOperational`	`Edm.Boolean`
Defines whether it is required to post the document in real-time mode. If such posting is required, the property value should take the true value. Default value: false.	

12.4.2. Directory tabular section

Entity name	Document_<Object name>_<tabular section name>_ RecordType
Entities set name	Document_<Object name>_<tabular section name>_ RecordType

Properties components:

Name	Type	Comment
Ref	Reference type	Reference.
LineNumber	`Edm.Int64`	Line number.
<Attributes>	Corresponds to attribute type	
<Common attributes>	Corresponds to attribute type	Except for common attributes that are independent separators.

12.4.3. Tabular section row

Composite type name: Document_<Object name>_<tabular section name>_ RowType.

Properties components:

Name	Type	Comment
LineNumber	`Edm.Int64`	Number of tabular section string
<Attributes>	Corresponds to attribute type	

12.5. Documents journals

Entity name	DocumentJournal_<name>
Entities set name	DocumentJournal_<name>

Properties components:

Name	Type	Comment
Ref	Reference type	

Name	Type	Comment
Number	Edm.String, Edm.Int32	Number of document depending on set number type.
Date	Edm.Date	If enabled.
DeletionMark	Edm.Boolean	Deletion mark.
Type	Edm.String	Posting document type.
Posted	Edm.Boolean	Document posting flag.
<Graphs>	Corresponds to graph type	
<Attributes>	Corresponds to attribute type	
<Common attributes>	Corresponds to attribute type	Except for common attributes that are independent separators.

Behavior particularities:

- Only receiving a list of strings is supported
- It also does not support:
  - Document creation (POST query);
  - Changing the document (PATCH query);
  - Deleting a document (DELETE query);
  - Call "through a dot".

12.6. Charts of characteristic types

Entity name	ChartOfCharacteristicTypes_ <Name>
Entities set name	ChartOfCharacteristicTypes_ <Name>

Properties components:

Name	Type	Comment
Ref	Reference type	Reference.
Code	Edm.String, Edm.Int32	If the chart of characteristic types has code.
Description	Edm.String	If the chart of characteristic types has a name.

Name	Type	Comment
IsFolder	Edm.Boolean	For a hierarchical chart of characteristic types.
Owner	Link or composite type	Owner (for a subordinate chart of characteristic types)
Parent	Reference type	Parent (for a hierarchical chart of characteristic types).
DeletionMark	Edm.Boolean	Deletion mark.
Version	Edm.String	Data verison.
Predefined	Edm.Boolean	Predefined item flag. Read-only.
PredefinedDataName	Edm.String	Predefined item name.
ValueType	EnterpriseV8.TypeDescription	Characteristic value type.
<Attributes>	Corresponds to attribute type	
<Common attributes>	Corresponds to attribute type	Except for common attributes that are independent separators.
<Tabular sections>	Collection (CompositeType)	Each tabular section acts as an independent entity with a ChartOfCharacteristicTypes_<ObjectName>_<TabularSectionName>.

12.6.1. Tabular section of the chart of characteristic types

Entity name	ChartOfCharacteristicTypes_<Object name>_<Tabular section name>_ RecordType
Entities set name	ChartOfCharacteristicTypes_<Object name>_<Tabular section name>_ RecordType

Properties components:

Name	Type	Comment
Ref	Reference type	Reference.
LineNumber	Edm.Int64	Line number.
<Attributes>	Corresponds to attribute type	
<Common attributes>	Corresponds to attribute type	Except for common attributes that are independent separators.

12.6.2. Tabular section row

Composite type name: ChartOfCharacteristicTypes_<Object name>_<Tabular section name>_ RowType.

Properties components:

Name	Type	Comment
LineNumber	Edm.Int64	Row number in the tabular section.
<Attributes>	Corresponds to attribute type	

12.7. Charts of accounts

Entity name	ChartOfAccounts_<Name>
Entities set name	ChartOfAccounts_<Name>

Properties components:

Name	Type	Comment
Ref	Reference type	Reference.
Code	Edm.String,	If the chart of accounts has code.

Name	Type	Comment
	Edm.Int32	
Description	Edm.String	If the chart of accounts has a name.
OffBalance	Edm.Boolean	Flag of off-balance account.
Order	Edm.String	Sorting order.
Type	Edm.String	Account type: active, passive, active / passive.
Parent	Reference type	Parent (for hierarchical chart of accounts).
DeletionMark	Edm.Boolean	Deletion mark.
Version	Edm.String	Data verison.
Predefined	Edm.Boolean	Predefined item flag. Read-only.
Predefined-DataName	Edm.String	Predefined item name.
<AccountingFlag>	Edm.Boolean	Accounting flag.
<Attributes>	Corresponds to attribute type	
<Common attributes>	Corresponds to attribute type	Except for common attributes that are independent separators.
<Tabular sections>	Collection (CompositeType)	Each tabular section acts as an independent entity with a ChartOfAccounts_<Object-Name>_<TabularSection-Name>.
ExtDimensionTypes	Collection (CompositeType)	Standard tabular section of extra dimension type.

12.7.1. Tabular section of chart of accounts

Entity name	ChartOfAccounts_<Object name>_<Tabular section name>_RecordType
Entities set name	ChartOfAccounts_<Object name>_<Tabular section name>_RecordType

Properties components:

Name	Type	Comment
Ref	Reference type	Reference.

Name	Type	Comment
LineNumber	Edm.Int64	Line number.
<Attributes>	Corresponds to attribute type	
<Common attributes>	Corresponds to attribute type	Except for common attributes that are independent separators.

12.7.2. Tabular section row

Composite type name: ChartOfAccounts_<Object name>_<Tabular section name>_RowType.

Properties components:

Name	Type	Comment
LineNumber	Edm.Int64	Row number in the tabular section.
<Attributes>	Corresponds to attribute type	

12.7.3. Standard tabular section of extra dimension type string

Composite type name: ChartOfAccounts_<Object name>_ExtDimensions.

Properties components:

Name	Type	Comment
LineNumber	Edm.Int64	Number of tabular section string
ExtDimension-Type_Key	Edm.Guid	ID of chart of characteristic types containing extra dimension types
Predefined	Edm.Boolean	Flag of predefined extra dimension type
<Attributes>	Corresponds to attribute type	
<AccountingFlag>	Edm.Boolean	Accounting flag.

12.8. Charts of calculation types

Entity name	ChartOfCalculationTypes_<Name>
Entities set name	ChartOfCalculationTypes_<Name>

Properties components:

Name	Type	Comment
Ref	Reference type	Reference.
Code	Edm.String, Edm.Int32	If the chart of calculation types has code.
Description	Edm.String	If the chart of calculation types has a name.
ActionPeriodIsBasic	Edm.Boolean	Flag that the base period is the validity period (only for charts of calculation types with support for the validity period).
DeletionMark	Edm.Boolean	Deletion mark.
Version	Edm.String	Data verison.
Predefined	Edm.Boolean	Predefined item flag. Read-only.
PredefinedDataName	Edm.String	Predefined item name.
<Attributes>	Corresponds to attribute type	
<Common attributes>	Corresponds to attribute type	Except for common attributes that are independent separators.
<Tabular sections>	Collection (CompositeType)	Each tabular section acts as an independent entity with a ChartOfCalculationTypes_<ObjectName>_<TabularSectionName>.
LeadingCalculationTypes	Collection (CompositeType)	Standard tabular section "Leading calculation types".
DisplacingCalculationTypes	Collection (CompositeType)	Standard tabular section “Displacing calculation types” (only for chart of calculation types with the support of the validity period).

Name	Type	Comment
BaseCalculationTypes	Collection (CompositeType)	Standard tabular section “Basic calculation types” (only for chart of calculation types with the support of the validity period).

12.8.1. Tabular section of chart of calculations

Entity name	ChartOfCalculationTypes_<Object name>_<tabular section name>_RecordType
Entities set name	ChartOfCalculationTypes_<Object name>_<tabular section name>_RecordType

Properties components:

Name	Type	Comment
Ref	Reference type	Reference.
LineNumber	Edm.Int64	Line number.
<Attributes>	Corresponds to attribute type	
<Common attributes>	Corresponds to attribute type	Except for common attributes that are independent separators.

12.8.2. Tabular section row

Composite type name: ChartOfCalculationTypes_<Object name>_<tabular section name>_ RowType.

Properties components:

Name	Type	Comment
LineNumber	Edm.Int64	Row number in the tabular section.
<Attributes>	Corresponds to attribute type	

12.8.3. String of the standard tabular section “Leading calculation types”

Composite type name: ChartOfCalculationTypes_<Object name>_LeadingCalculationTypes.

Properties components:

Name	Type	Comment
LineNumber	Edm.Int64	Row number in the tabular section.
CalculationType	Edm.Guid	ID of the chart of calculation types containing calculation types.
Predefined	Edm.Boolean	Flag of predefined extra dimension type.

12.8.4. String of the standard tabular section “Displacing calculation types”

Composite type name: ChartOfCalculationTypes_<Object name>_BaseCalculationTypes.

Properties components:

Name	Type	Comment
LineNumber	Edm.Int64	Row number in the tabular section.
CalculationType	Edm.Guid	ID of the chart of calculation types containing calculation types.
Predefined	Edm.Boolean	Flag of predefined extra dimension type.

12.8.5. String of the standard tabular section “Basic calculation types”

Composite type name: ChartOfCalculationTypes_<Object name>_DisplacingCalculationTypes.

Properties components:

Name	Type	Comment
LineNumber	Edm.Int64	Row number in the tabular section.
CalculationType_Key	Edm.Guid	ID of the chart of calculation types containing calculation types.
Predefined	Edm.Boolean	Flag of predefined extra dimension type.

12.9. Information registers

12.9.1. Independent

Entity name	InformationRegister_<Name>
Entities set name	InformationRegister_<Name>

Properties components:

Name in the schema	Type	Comment
Period	Edm.Date	Record period Only for periodic information registers.
SurrogateKey	Edm.Int16	Surrogate key in case there are no dimensions and separators. Is always equal to 0.
<Dimensions>	Corresponds to dimension type	Dimensions are included in the entity key.
<Resources>	Corresponds to resource type	
<Attributes>	Corresponds to attribute type	
<Common attributes>	Corresponds to attribute type	Except for common attributes that are independent separators.

Behavior particularities:

Dimensions of the Information register form the record key. If you try to change the record key, an exception with HTTP status 404 will be generated. In this case, its recommended to delete the existing record first and then add a new one.

12.9.1.1. Functions

SliceLast

Description:

Entities set- slice of the last periodic information registers.

Parameters:

`Period`	`Edm.Date`
Period of slice acquisition.	

`Condition`	`Edm.String`
Filter of records in OData format.	

SliceFirst

Description:

Entities set - slice of the first periodic information registers.

Parameters:

`Period`	`Edm.Date`
Period of slice acquisition.	

`Condition`	`Edm.String`
Filter of records in OData format.	

12.9.2. Subordinate to the recorder

Entity name	InformationRegister_<RegisterName>
Entities set name	InformationRegister_<RegisterName>

Properties components:

Name	Type	Comment
Recorder	`Edm.String`	Recorder.
Re-corder_Type	`Edm.String`	Link to recorder.
RecordSet	`Collection (CompositeType)`	Record set.

Behavior particularities:

Dimensions of the Information register form the record key. If you try to change the record key, an exception with HTTP status 404 will be generated. In this case, its recommended to delete the existing record first and then add a new one.

12.9.2.1. Functions

SliceLast

Description:

Entities set- slice of the last periodic information registers.

Parameters:

`Period`	`Edm.Date`
Period of slice acquisition.	

`Condition`	`Edm.String`
Filter of records in OData format.	

SliceFirst

Description:

Entities set - slice of the first periodic information registers.

Parameters:

`Period`	`Edm.Date`
Period of slice acquisition.	

`Condition`	`Edm.String`
Filter of records in OData format..	

12.9.2.2. Register record set

Composite type name: InformationRegister_<Name>_RowType

Properties components:

Name	Type	Comment
LineNumber	`Edm.Int64`	Line number in the record set
Active	`Edm.Boolean`	Flag of string activeness
SurrogateKey	`Edm.Int16`	Surrogate key in case there are no dimensions, separators and the recording period is different from the Recorder. Is always equal to 0.

Name	Type	Comment
<Dimensions>	Corresponds to dimension type	
<Resources>	Corresponds to resource type	
<Attributes>	Corresponds to attribute type	
<Common attributes>	Corresponds to attribute type	Except for common attributes that are independent separators.

12.9.2.3. String of the information register record set

Entity name	InformationRegister_<Name>_RecordType
Entities set name	InformationRegister_<Name>_RecordType

Properties components:

Name	Type	Comment
LineNumber	`Edm.Int64`	Line number in record set.
Active	`Edm.Boolean`	Active string flag.
SurrogateKey	`Edm.Int16`	Surrogate key in case there are no dimensions, separators and the recording period is different from the Recorder. Is always equal to 0.
<Dimensions>	Corresponds to dimension type	
<Resources>	Corresponds to resource type	
<Attributes>	Corresponds to attribute type	
<Common attributes>	Corresponds to attribute type	Except for common attributes that are independent separators.

Behavior particularities:

Read-only entity (GET query).

12.10. Accumulation register

Entity name	AccumulationRegister_<RegisterName>
Entities set name	AccumulationRegister_<RegisterName>

Properties components:

Name	Type	Comment
Recorder	Edm.String	Recorder.
Recorder_Type	Edm.String	Link to recorder.
RecordSet	Collection (CompositeType)	Record set.

Behavior particularities:

Dimensions of the Information register form the record key. If you try to change the record key, an exception with HTTP status 404 will be generated. In this case, its recommended to delete the existing record first and then add a new one.

12.10.1. Functions

Balance

Description:

Provides access to the virtual table of accumulation register balance.

Parameters:

Period	Edm.Date

The date on which the accumulation register balances are required.

Condition	Edm.String

Filter of records in OData format.

Dimensions	Edm.String

Changes list separated by comma.

Turnovers

Description:

Provides access to the virtual turnover table of the accumulation register. Expansion of totals by period is not supported.

Parameters:

Condition	Edm.String

Filter of records in OData format..

EndPeriod	Edm.Date

The end of the period for which the turnovers are required.

`StartPeriod` `Edm.Date`

The begginng of the period for which the turnovers are required.

BalanceAndTurnovers

Description:

Provides access to the virtual table of balancse and turnovers of the accumulation register. Expansion of totals by period is not supported.

Parameters:

`Condition` `Edm.String`

Filter of records in OData format.

`EndPeriod` `Edm.Date`

The end of the period for which the turnovers are required.

`StartPeriod` `Edm.Date`

The begginng of the period for which the turnovers are required.

12.10.2. Register record set

Composite type name: AccumulationRegister_<RegisterName>_RowType.

Properties components:

Name	Type	Comment
LineNumber	`Edm.Int64`	Line number in record set.
Active	`Edm.Boolean`	Active string flag.
<Dimensions>	Corresponds to dimension type	
<Resources>	Corresponds to resource type	
<Attributes>	Corresponds to attribute type	
<Common attributes>	Corresponds to attribute type	Except for common attributes that are independent separators.

Behavior particularities:

Dimensions of the Information register form the record key. When attempting to change the record key, an exception with HTTP 404 code will be generated. The correct way - delete a record and create a new one.

12.10.3. String of accumulation register record set

Entity name	AccumulationRegis-ter_<Name>_RecordType
Entities set name	AccumulationRegis-ter_<Name>_RecordType

Properties components:

Name	Type	Comment
LineNumber	Edm.Int64	Line number in record set.
Active	Edm.Boolean	Active string flag.
<Dimensions>	Corresponds to dimension type	
<Resources>	Corresponds to resource type	
<Attributes>	Corresponds to attribute type	
<Common attrib-utes>	Corresponds to attribute type	Except for common attributes that are independent separators.

Behavior particularities:

Read-only entity (GET query).

12.11. Accounting registers

Entity name	AccountingRegister_<RegisterName>
Entities set name	AccountingRegister_<RegisterName>

Properties components:

Name	Type	Comment
Recorder	Edm.String	Recorder.
Recorder_Type	Edm.String	Link to recorder.
RecordSet	Collection (CompositeType)	Record set.

Behavior particularities:

Dimensions of the Information register form the record key. If you try to change the record key, an exception with HTTP status 404 will be generated. In this case, its recommended to delete the existing record first and then add a new one.

12.11.1. Functions

Balance

Description:

Provides access to the virtual table of accounting register balance.

Parameters:

`Period`	`Edm.Date`

The date on which the accounting register balances are required.

`Condition`	`Edm.String`

Filter of records in OData format.

`Dimensions`	`Edm.String`

Changes list separated by comma.

`AccountCondition`	`Edm.String`

Filter condition for accounts of OData format

`ExtraDimensions`	`Edm.String`

The list of UUID of extra dimension type for which you want to set the filtering.

Turnovers

Description:

Provides access to the virtual turnover table of the accounting register. Expansion of totals by period is not supported.

Parameters:

`Condition`	`Edm.String`

Filter of records in OData format.

`EndPeriod`	`Edm.Date`

The end of the period for which the turnovers are required.

`StartPeriod`	`Edm.Date`

The begginng of the period for which the turnovers are required.

`AccountCondition`	`Edm.String`

Filter condition for accounts of OData format

`BalancedAccountCondition` `Edm.String`

Filter condition for offset accounts of OData format

`Dimensions` `Edm.String`

Changes list separated by comma.

`ExtraDimensions` `Edm.String`

The list of UUID of extra dimension type for which you want to set the filtering.

`BalancedExtraDimensions` `Edm.String`

The list of UUID of offset extra dimension types for which you want to set the filtering.

BalanceAndTurnovers

Description:

Provides access to the virtual table of balances and turnovers of the accounting register. Expansion of totals by period is not supported.

Parameters:

`Condition` `Edm.String`

Filter of records in OData format.

`EndPeriod` `Edm.Date`

The end of the period for which the turnovers are required.

`StartPeriod` `Edm.Date`

The begginng of the period for which the turnovers are required.

ExtDimensions

Description:

Provides access to the virtual table of accounting register extra dimension.

RecordsWithExtDimensions

Description:

Provides access to the virtual table of movements with extra dimensions of the accounting register.

Parameters:

`Condition` `Edm.String`

Filter of records in OData format.

`EndPeriod` `Edm.Date`

The end of the period for which the turnovers are required.

`StartPeriod` `Edm.Date`

The begginng of the period for which the turnovers are required.

`Top` `Edm.Int32`

Limiting the maximum amount of records.

`OrderBy` `Edm.String`

A string containing column names (separated by commas) by which the transactions will be sorted.

DrCrTurnover

Description:

Provides access to the virtual table to get turnovers on the accounting register for an arbitrary period with a certain aggregation. Expansion of totals by period is not supported.

Parameters:

`Condition` `Edm.String`

Filter of records in OData format.

`EndPeriod` `Edm.Date`

The end of the period for which the turnovers are required.

`StartPeriod` `Edm.Date`

The begginng of the period for which the turnovers are required.

`AccountCondition` `Edm.String`

Filter condition for accounts of OData format.

`BalancedAccountCondition` `Edm.String`

Filter condition for offset accounts of OData format.

`Dimensions` `Edm.String`

Changes list separated by comma.

`ExtraDimensions` `Edm.String`

The list of UUID of extra dimension type for which you want to set the filtering.

BalancedExtraDimensions	*Edm.String*
The list of UUID of offset extra dimension types for which you want to set the filtering.	

12.11.2. Register record set

Composite type name: AccountingRegister_<RegisterName>_RowType.

Properties components:

Name	Type	Comment
LineNumber	Edm.Int64	Line number in record set.
Active	Edm.Boolean	Active string flag.
Period	Edm.Date	Time when record is created.
RecordType	Edm.String	Movement type of accounting register record: Debit or Credit.
AccountDr_Key	Edm.Guid	Debit account (only for register with correspondence).
AccountCr_Key	Edm.Guid	Credit account (only for register with correspondence).
Account_Key	Edm.Guid	Account (only for register with correspondence).
<Dimensions>	Corresponds to dimension type	For unbalanced dimensions, two properties: • <DimensionName>Dr • <DimensionName>Cr.
<Resources>	Corresponds to resource type	For unbalanced dimensions, two properties: • <ResourceName>Dr • <ResourceName>Cr.
<Attributes>	Corresponds to attribute type	•
<Common attributes>	Corresponds to attribute type	Except for common attributes that are independent separators.

12.11.3. String of the information register record set

Entity name	AccountingRegister_<Name>_RecordType
Entities set name	AccountingRegister_<Name>_RecordType

Properties components:

Name	Type	Comment
LineNumber	Edm.Int64	Line number in record set.
Active	Edm.Boolean	Active string flag.
Period	Edm.Date	Time when record is created.
RecordType	Edm.String	Movement type of accounting register record: Debit or Credit.
AccountDr_Key	Edm.Guid	Debit account (only for register with correspondence).
AccountCr_Key	Edm.Guid	Credit account (only for register with correspondence).
Account_Key	Edm.Guid	Account (only for register with correspondence).
<Dimensions>	Corresponds to dimension type	For unbalanced dimensions, two properties: • <DimensionName>Dr • <DimensionName>Cr.
<Resources>	Corresponds to resource type	For unbalanced dimensions, two properties: • <ResourceName>Dr • <ResourceName>Cr.
<Attributes>	Corresponds to attribute type	•
<Common attributes>	Corresponds to attribute type	Except for common attributes that are independent separators.

Behavior particularities:

Read-only entity (GET query).

12.12. Calculation Registers

Entity name	CalculationRegister_<RegisterName>
Entities set name	CalculationRegister_<RegisterName>

Properties components:

Name	Type	Comment
Recorder	Edm.String	Recorder.
Recorder_Type	Edm.String	Link to recorder.
RecordSet	Collection (CompositeType)	Record set.

Behavior particularities:

Dimensions of the Information register form the record key. If you try to change the record key, an exception with HTTP status 404 will be generated. In this case, its recommended to delete the existing record first and then add a new one.

12.12.1. Functions

ScheduledData

Description:

Provides access to the virtual table data schedule calculation register. Only for calculation register with a set schedule

Parameters:

`Condition`	`Edm.String`

Filter of records in OData format.

ActualActionPeriod

Description:

Provides access to the virtual table of actual action period. Only for the register with action period support.

Parameters:

`Condition`	`Edm.String`

Filter of records in OData format.

<RecalculationName>

Description:

Provides access to the virtual table of recalculation records.

Base<BaseCalculationRegisterName>

Description:

Provides access to the virtual table of calculation register base data.

Parameters:

`Condition`	`Edm.String`

Filter of records in OData format.

MainRegisterDimensions *Edm.String*

Names of main calculation register that are used to create a base data table, comma separated.

BaseRegisterDimensions *Edm.String*

Names of base calculation register dimensions that are used to create a base data table, comma separated.

ViewPoints *Edm.String*

Names of base calculation register fields that are used to sum the base data, comma separated

12.12.2. Register record set

Composite type name: CalculationRegister_<RegisterName>_RowType.

Properties components:

Name	Type	Comment
LineNumber	Edm.Int64	Line number in record set.
Active	Edm.Boolean	Active string flag.
Period	Edm.Date	Time when record is created.
BegOfBasePeriod	Edm.Date	Account's base period beginning date. Only for registers with base period.
EndOfBasePeriod	Edm.Date	Base period end date. Only for calculation registers with base period.
BegOfActionPeriod	Edm.Date	Action period beginning date. Only for calculation registers with action period.
EndOfActionPeriod	Edm.Date	Action period end date. Only for calculation registers with action period.
CalculationType_Key	Edm.Guid	Link to account's calculation type.
ReversingEntry	Edm.Boolean	Reversing entry record flag
<Dimensions>	Corresponds to dimension type	For unbalanced dimensions, two properties: • <DimensionName>Dr • <DimensionName>Cr.

Name	Type	Comment
<Resources>	Corresponds to resource type	For unbalanced dimensions, two properties: • <ResourceName>Dr • <ResourceName>Cr.
<Attributes>	Corresponds to attribute type	
<Common attributes>	Corresponds to attribute type	Except for common attributes that are independent separators.

12.12.3. String of the information register record set

Entity name	CalculationRegister_<Name>_RecordType
Entities set name	CalculationRegister_<Name>_RecordType

Properties components:

Name	Type	Comment
LineNumber	Edm.Int64	Line number in record set.
Active	Edm.Boolean	Active string flag.
Period	Edm.Date	Time when record is created.
BegOfBasePeriod	Edm.Date	Account's base period beginning date. Only for registers with base period.
EndOfBasePeriod	Edm.Date	Base period end date. Only for calculation registers with base period.
BegOfActionPeriod	Edm.Date	Action period beginning date. Only for calculation registers with action period.
EndOfActionPeriod	Edm.Date	Action period end date. Only for calculation registers with action period.
CalculationType_Key	Edm.Guid	Link to account's calculation type.
ReversingEntry	Edm.Boolean	Reversing entry record flag
<Dimensions>	Corresponds to dimension type	For unbalanced dimensions, two properties: • <DimensionName>Dr • <DimensionName>Cr.

Name	Type	Comment
<Resources>	Corresponds to resource type	For unbalanced dimensions, two properties: • <ResourceName>Dr • <ResourceName>Cr.
<Attributes>	Corresponds to attribute type	•
<Common attributes>	Corresponds to attribute type	Except for common attributes that are independent separators.

Behavior particularities:

Read-only entity (GET query).

12.13. Business processes

Entity name	BusinessProcess_<Name>
Entities set name	BusinessProcess_<Name>

Properties components:

Name	Type	Comment
Ref	Reference type	Reference.
Code	Edm.String, Edm.Int32	If business process has a code.
Description	Edm.String	If business process has a name.
DeletionMark	Edm.Boolean	Deletion mark.
Version	Edm.String	Data verison.
Started	Edm.Boolean	Business process started.
Competed	Edm.Boolean	Business process completed.
HeadTask		Two fields: • <Attribute name>- contains link value as a string; • <Attribute name>_Type -link to a task, that is processing this business process.
<Attributes>	Corresponds to attribute type	

Name	Type	Comment
<Common attributes>	Corresponds to attribute type	Except for common attributes that are independent separators.
<Tabular sections>	Collection (CompositeType)	Each tabular section acts as an independent entity with a Task_<Object name>_<TabularSectionName> name.

12.13.1. Functions

Start

Description:

Business process is started.

Parameters:

RoutePoint
Route point name, from where the business process is started. Optional parameter.

12.13.2. Tabular section of business process

Entity name	BusinessProcess_<Object name>_<Tabular section name>_RecordType
Entities set name	BusinessProcess_<Object name>_<Tabular section name>_RecordType

Properties components:

Name	Type	Comment
Ref	Reference type	Reference.
LineNumber	Edm.Int64	Line number.
<Attributes>	Corresponds to attribute type	•
<Common attributes>	Corresponds to attribute type	Except for common attributes that are independent separators.

12.13.3. Tabular section row

Composite type name: BusinessProcess_<Object name>_<Tabular section name>_RowType.

Properties components:

Name	Type	Comment
LineNumber	Edm.Int64	Row number in the tabular section.
<Attributes>	Corresponds to attribute type	

12.14. Tasks

Entity name	Task_<Name>
Entities set name	Task_<Name>

Properties components:

Name	Type	Comment
Ref	Reference type	Reference.
Number	Edm.String, Edm.Int32	If the task has a number.
Description	Edm.String	If the task has a description
DeletionMark	Edm.Boolean	Deletion mark.
Version	Edm.String	Data verison.
Executed	Edm.Boolean	Task executed.
BusinessProcess		Describes composite type, if the task is related to several business processes: • <Attribute name>- contains link value as a string; • <Attribute name>_Type- link to a task, that is processing this business process. If the task is related to one business process, the attribute is described by two properties: • Link;

Name	Type	Comment
		• BusinessProcess_Key-link value.
Route-Point	Edm.String	Route point name, for execution of which a task has been created. Only for task related to a business process.
<Attrib-utes>	Corresponds to attribute type	
<Com-mon at-tributes>	Corresponds to attribute type	Except for common attributes that are independent separators.
<Tabular sections>	Collection (CompositeType)	Each tabular section acts as an independent entity with a Task_<Object name>_<TabularSection-Name> name.

12.14.1. Functions

ExecuteTask

Description:

Executes task.

12.14.2. Tabular section of business process

Entity name	Task_<Object name>_<Tabular section name>_ RecordType
Entities set name	Task_<Object name>_<Tabular section name>_RecordType

Properties components:

Name	Type	Comment
Ref	Reference type	
LineNumber	Edm.Int64	Line number.
<Attributes>	Corresponds to attribute type	
<Common attrib-utes>	Corresponds to attribute type	Except for common attributes that are independent separators.

12.14.3. Tabular section row

Composite type name: Task_<Object name>_<tabular section name>_RowType.

Properties components:

Name	Type	Comment
LineNumber	Edm.Int64	Row number in the tabular section.
<Attributes>	Corresponds to attribute type	

Appendix 13. Namespace prefixes in JSON serialization

This appendix illustrates the correspondence of type prefixes of XDTO objects in JSON serialization and namespaces of XDTO objects in XML serialization.

Prefix	URI
jxs	http://www.w3.org/2001/XMLSchema
jxsi	http://www.w3.org/2001/XMLSchema-instance
jv8	http://v8.1c.ru/8.1/data/core
jv8ui	http://v8.1c.ru/8.1/data/ui
jstyle	http://v8.1c.ru/8.1/data/ui/style
jwin	http://v8.1c.ru/8.1/data/ui/colors/windows
jweb	http://v8.1c.ru/8.1/data/ui/colors/web
jsys	http://v8.1c.ru/8.1/data/ui/fonts/system
jcfg	http://v8.1c.ru/8.1/data/enterprise/current-config
jxcf	http://v8.1c.ru/8.3/MDClasses
jxr	http://v8.1c.ru/8.3/xcf/readable
jxep	http://v8.1c.ru/8.3/xcf/extrnprops
jxpr	http://v8.1c.ru/8.3/xcf/predef
jxen	http://v8.1c.ru/8.3/xcf/enums
jrole	http://v8.1c.ru/8.2/roles
jchm	http://v8.1c.ru/8.2/data/chart
japp	http://v8.1c.ru/8.2/managed-application/core
jcmi	http://v8.1c.ru/8.2/managed-application/cmi
jdl	http://v8.1c.ru/8.2/managed-application/dynamic-list
jlf	http://v8.1c.ru/8.2/managed-application/logform
jlfl	http://v8.1c.ru/8.2/managed-application/logform/layouter
jm	http://v8.1c.ru/8.2/managed-application/modules
jmng	http://v8.1c.ru/8.2/mngsrv/ws
jus	http://v8.1c.ru/8.2/managed-application/user-settings
juo	http://v8.1c.ru/8.2/uobjects
jdcsdet	http://v8.1c.ru/8.1/data-composition-system/details
jdcscor	http://v8.1c.ru/8.1/data-composition-system/core

Prefix	URI
jdcsset	http://v8.1c.ru/8.1/data-composition-system/settings
jdcsapt	http://v8.1c.ru/8.1/data-composition-system/appearance-tem-plate
jdcsat	http://v8.1c.ru/8.1/data-composition-system/area-template
jdcscom	http://v8.1c.ru/8.1/data-composition-system/common
jdcsct	http://v8.1c.ru/8.1/data-composition-system/composition-tem-plate
jdcsres	http://v8.1c.ru/8.1/data-composition-system/result
jdcssch	http://v8.1c.ru/8.1/data-composition-system/schema
jda	http://v8.1c.ru/8.2/data/data-analysis
jent	http://v8.1c.ru/8.1/data/enterprise
jgeo	http://v8.1c.ru/8.2/data/geo
jmxl	http://v8.1c.ru/8.2/data/spreadsheet
jsch	http://v8.1c.ru/8.2/data/graphscheme
jxsch	http://v8.1c.ru/8.3/xcf/scheme
jtxt	http://v8.1c.ru/8.1/data/txtedt
jedbexc	http://v8.1c.ru/8.2/managed-application/edbexception
jscexc	http://v8.1c.ru/8.2/managed-application/seancecontextexcep-tion
jdeploy-ment	http://v8.1c.ru/8.2/managed-application/deployment
jlfexc	http://v8.1c.ru/8.2/managed-application/logformexception
jvrs	http://v8.1c.ru/8.2/virtual-resource-system
jv8	http://v8.1c.ru/8.1/data/core
jv8ui	http://v8.1c.ru/8.1/data/ui
jxdm	http://v8.1c.ru/8.1/xdto
jscript	http://v8.1c.ru/8.2/bsl
jwsm	http://v8.1c.ru/8.1/ws/wsdefinitions-model
jbsch	http://v8.1c.ru/8.2/data/bsl
jextm	http://v8.1c.ru/8.3/data/ext

Appendix 14. New and modified sections

Deleted section:

- OC Windows Phone.
- OS Windows on touch devices.

Added section:

- Section 5.1.4. Properties related to development for mobile devices.
- Interactive modification of chart values.
- Secondary value scale.
- Chart integration.
- Series accumulation.
- Using a single color for a series.
- Section 7.7.4.3. Data representation.
- Section 7.7.6.6. Search in tables.
- Section 8.4.11.2. Grouping data sets.
- Section 10.10.4. Using a single color for a seri.
- Section 16.4. PDF files.
- Chapter 18. Web client and web site integration.
- Section 27.5. Copy content management.
- Section 29.3.5.15. Backup by mobile operating system.
- Section 29.6. Offline mobile client.
- Section 29.7.3.2. Publication details.
- Debugging offline mobile application.
- Section 36.4.19. Functional options and functional option parameters.
- Section 36.4.20. Filter criteria.
- Appendix 7.2.12. With version 8.3.15.

Modified section:

- Section 1.4.4. Types of client applications.
- Section 1.4.7. Summary.
- Section 5.1.2.1. The interface parameters.
- Section 5.1.6. "Compatibility" properties category.
- Section 7.3.1.6. Search in dynamic list.
- Section 7.7.9. Adding a form item.
- Section 8.4.5. Bilingual presentation of keywords.
- Section 8.4.16.4. Built-in functions of query language.

- Section 10.3.6.2. Data composition settings properties.
- Section 20.1. Overall information.
- Section 25.6. Deleting a data area.
- Section 27.1. General information.
- Section 27.2. Mechanism structure.
- Section Command description.
- Section QR-code content.
- Section 29.1. General information.
- Section 29.5.3. Digital signature and configuration digest.
- Section 29.3.5.16. Other mobile device features.
- Section 29.7.2.1. General information.
- Section 29.8.1. General information.
- Section 29.8.3.1.1. Mobile versions.
- Section 29.8.3.1.4. Graphic information..
- Section 29.8.4.2.1. General information.
- Section 30.1. General information.
- Section 30.3.3.1. Text messages.
- Section 31.22. Centralized Configuration Verification .
- Section 32.2.1. General information.
- Section 36.1.3. Specifics and limitations.
- Section 36.4.3.2. Working with the form editor.
- Section 37.2.5.1. General parameters.
- Section 39.6. HTMLDocumentField.
- Appendix 4.5. 1C:Enterprise storage.
- Appendix 5. Full-text search expressions.
- Appendix 6. Access rights description.