



One Identity Manager 8.2.1

Configuration Guide

Copyright 2022 One Identity LLC.

ALL RIGHTS RESERVED.

This guide contains proprietary information protected by copyright. The software described in this guide is furnished under a software license or nondisclosure agreement. This software may be used or copied only in accordance with the terms of the applicable agreement. No part of this guide may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording for any purpose other than the purchaser's personal use without the written permission of One Identity LLC .

The information in this document is provided in connection with One Identity products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of One Identity LLC products. EXCEPT AS SET FORTH IN THE TERMS AND CONDITIONS AS SPECIFIED IN THE LICENSE AGREEMENT FOR THIS PRODUCT, ONE IDENTITY ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL ONE IDENTITY BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF PROFITS, BUSINESS INTERRUPTION OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF ONE IDENTITY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. One Identity makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and product descriptions at any time without notice. One Identity does not make any commitment to update the information contained in this document.

If you have any questions regarding your potential use of this material, contact:

One Identity LLC.
Attn: LEGAL Dept
4 Polaris Way
Aliso Viejo, CA 92656

Refer to our Web site (<http://www.OneIdentity.com>) for regional and international office information.

Patents

One Identity is proud of our advanced technology. Patents and pending patents may apply to this product. For the most current information about applicable patents for this product, please visit our website at <http://www.OneIdentity.com/legal/patents.aspx>.

Trademarks

One Identity and the One Identity logo are trademarks and registered trademarks of One Identity LLC. in the U.S.A. and other countries. For a complete list of One Identity trademarks, please visit our website at www.OneIdentity.com/legal. All other trademarks are the property of their respective owners.

Legend

 **WARNING:** A WARNING icon highlights a potential risk of bodily injury or property damage, for which industry-standard safety precautions are advised. This icon is often associated with electrical hazards related to hardware.

 **CAUTION:** A CAUTION icon indicates potential damage to hardware or loss of data if instructions are not followed.

One Identity Manager Configuration Guide
Updated - 25 May 2022, 15:25
Version - 8.2.1

Contents

About this guide	17
One Identity Manager software architecture	18
Working with objects in One Identity Manager	22
Inserting, modifying, and deleting an object in One Identity Manager	25
Customizing the One Identity Manager default configuration	28
Reloading changes dynamically	28
Locking and unlocking individual properties for editing	30
System configuration reports	31
Customizing the One Identity Manager base configuration	33
Overview of the database settings	33
Changing database connection data	36
Database configuration for a test, development, or productive environment	37
Changing the database staging level	38
Language settings for displaying and maintaining the data	38
Setting login languages	39
Default country for determining working hours and public holidays	39
Configuration parameters for system configuration	40
Editing configuration parameters	40
Creating custom configuration parameters	41
Configuration parameter properties	42
Configuration parameter options	43
One Identity Manager schema basics	44
Overview of the One Identity Manager schema	45
Displaying data models in the Designer	47
Displaying the column dependencies based on templates	48
Table types and default columns in the One Identity Manager data model	49
Notes on editing table definitions and column definitions	53
Table definitions	54
Table types in One Identity Manager	54
Database views of the View type	55

Database views of the proxy type	57
Database views of the Union type	59
Database views of the Read-only type	61
Table scripts	63
Working with a globally unique identifier module	64
Defining unique columns for tables	65
Specifying deferred deletion for objects	67
Editing table definitions	68
Table definition properties	68
Displaying the table definition Customizer	74
Column definitions	74
Templates for generating values	75
Editing value templates	76
Preventing a change to a column	77
Restricting performance of value templates	78
Example of local value templates within an object	79
Example of cross-object value templates	79
Limiting column lengths	80
Defining decimal places for displaying values	80
Using predefined formatting types	81
Creating formatting scripts	83
Column dependencies for setting values	84
Permitted column values	84
Specifying requirements for MVP columns	86
Defining bitmasks	86
Configuring columns for full-text search	87
Scripts for conditionally displaying and editing columns	89
Editing column definitions	90
Column definition properties	91
Table relations	99
Dynamic foreign key	102
Supporting file groups	104
Editing the user interface	107
Object definitions for the user interface	108
Selection criteria for object definitions	109

Using the captions for object definitions	109
Creating and editing object definitions	110
Object definition properties	111
User interface navigation	112
Navigation elements	113
Recommendations for editing menu navigation	115
Tips for working with the User Interface Editor	116
Selecting the user interface navigation view for editing	116
Loading a complete user interface navigation	117
Loading menu navigation using an application	117
Direct loading of menu items	117
Loading menu items using permissions groups	118
Loading menu navigation using a where clause	119
Simulating user interface navigation during editing	119
Copying existing user interface navigation for new permissions groups	120
Creating a new user interface navigation	121
Copying menu items	122
Creating new menu items	123
Creating new menu categories	123
Assigning menu items to applications	124
Assigning menu items to permissions groups	125
General menu item properties	125
Creating database queries for data-dependent menu items	128
Recursive data-dependent menu items	130
Editing lists	130
Display template for displaying a list	132
Defining insert values	132
Using links in the navigation	133
Using variables in the navigation	135
Creating and displaying variables	137
Forms for the user interface	138
Recommendations for editing forms	139
Editing user interface forms	140
Tips for working with the Form Editor	141
Disabling user interface forms	141

Copying user interface forms	142
Creating user interface forms	142
Assigning user interface forms to applications	143
Assigning user interface forms to permissions groups	144
Assigning user interface forms to object definitions	144
Effects of object definitions when displaying interface forms	145
Assigning user interface forms to menu items	146
Displaying custom columns and tables on main data forms	146
Forms for custom extensions	147
Hierarchical display of data on assignment forms	149
Replacing default forms with custom forms	150
User interface form properties	151
Form definitions and form templates	153
Working with overview forms	162
Creating overview forms	163
Adding more form elements to overview forms	165
Special features of editing overview forms	166
Customizing the form elements layout	167
Previewing an overview form during editing	169
Disabling overview forms and form elements	169
Deleting form elements	170
Deleting overview forms	170
Statistics in One Identity Manager	171
Creating and editing statistic definitions	171
General properties statistic definitions	172
Querying statistic measurements	174
Disabling statistics definition	175
Including statistics in the user interface	176
Using reports in statistics	177
Using simple reports in statistics	178
Diagram types for visualizing statistics	179
Examples of statistic definitions	183
Extending the Launchpad	187
Recommendations for extending the Launchpad	188
Actions for the Launchpad	189

Creating new menu items and actions for the Launchpad	190
Task definitions for the user interface	191
Creating and editing task definitions	191
Disabling task definitions	192
Script for conditional displaying of tasks	193
Properties of task definitions	193
Applications for configuring the user interface	195
Program properties	196
Icons and images for configuring the user interface	198
Using predefined database queries	199
Localization in One Identity Manager	201
Working in different time zones	202
Determining working hours	202
Editing country information	203
Setting countries and states	203
Specifying the standard hours for countries and states/provinces/counties	204
Displaying public holidays for countries and states	204
Editing countries	205
Editing states	206
Country properties	207
State properties	208
Public holiday properties	209
Language-dependent data representation	210
Basic rules for using language-dependent data	211
Flagging columns for translation	212
Using the text memory for translation	214
Displaying translations in the Language Editor	214
Showing usage of a translation	215
Editing translations of a single table	216
Editing all translations	217
Changing translation keys	218
Importing translations from the language pack	219
Process orchestration in One Identity Manager	220
Mapping processes in One Identity Manager	220

Editing processes with the Process Editor	222
Defining processes	224
Creating and editing processes	224
Copying processes	225
Creating and editing process steps	226
Copying single process steps	227
Copying process steps within a process	227
Searching for entries within processes	228
Comparing processes	229
Exporting and importing processes	229
Process properties	230
Process step properties	231
Process step parameters	234
Events for processes	238
Permissions for triggering processes	240
Simulating process generation	241
Checking the validity of a process	242
Compiling processes	244
Using process-specific and global variables for the process definition	245
Thresholds for handling processes	247
Specifying the executing server	248
Selecting servers with server functions	248
Selecting servers with selection scripts	248
Notifications about process step handling	249
Running processes automatically	251
Displaying process plan status	251
Starting a process plan immediately	252
Creating and editing process plans	252
Process plan properties	253
Overview of process components	254
Displaying and editing process task exe types	256
Changing the maximum number of instances for process tasks and process components	257
Properties of process components, process tasks, and parameter templates	258
Setting up Job servers	261

Editing the Job server	262
Job server properties	263
Machine roles and server functions	265
Overview of server functions	266
Overview of machine roles	267
Job server statistics	269
Connection data for process generation	270
Entering connection data for the application server	270
Entering Job server connection data	272
Installing the One Identity Manager Service on a Job server remotely	272
Configuring the Job server for connecting to the application server	274
The One Identity Manager Service functionality	276
Handling processes with the One Identity Manager Service	277
Parallel processing of processes by the One Identity Manager Service	278
Running external processes with the StdioProcessor	279
Configuring the One Identity Manager Service	279
One Identity Manager Service configuration files	283
Customizing the One Identity Manager Service configuration for a Job server	284
Template for the configuration file	285
Selecting module types and editing parameters	286
Validating the configuration file	287
Process collection module	288
MSSQLJobProvider	288
FileJobProvider	289
FTPJobProvider	291
HTTPJobProvider	293
AppServerJobProvider	294
Job destination module	294
JobServiceDestination	295
FileJobDestination	297
FTPJobDestination	299
HTTPJobDestination	301
Configuration module	302
Logwriter module	305
EventLogLogWriter	305

FileLogWriter	306
Dispatcher module	308
Connection module	309
HTTP authentication module	310
Module plugins	310
HTTPLogPlugin	311
ScheduleCommandPlugin	311
RequestWatchDogPlugin	312
PerformanceCounterPlugin	312
DebugMailPlugin	313
ShareInfoPlugin	313
RemoteConnectPlugin	313
DatabaseAgentPlugin	314
File module with private key	314
Tracking changes with process monitoring	316
Basic rules for process monitoring	317
Logging data changes	318
Labeling columns for recording changes to data	319
Logging process information during process handling	320
Editing process information for processes	322
Editing process information for process steps	322
Editing process information for events	323
Recording messages in the process history	324
Process tracking for DBQueue Processor operations	325
Example of replacing the GenProcID	326
Archiving and deleting records	330
Deleting log entries in the One Identity Manager database without archiving	331
Specifying log retention times	332
Optimizing performance by deleting log entries	333
Conditional compilation using preprocessor conditions	335
Preprocessor-relevant configuration parameters	336
Preprocessor conditions in objects	337
Preprocessor conditions in VB.Net expressions	338
Evaluation of preprocessor conditions during compilation	339

Scripts in One Identity Manager	341
Visual Basic .NET scripts usage	341
Notes on message output	342
Notes on using date values	342
Tips for using Windows PowerShell scripts	343
Using dollar (\$) notation	344
Accessing local object columns	345
Accessing columns of an object connected by a relation	346
Accessing the old column value	346
Accessing the display value of a column	347
Accessing references in comments	348
Accessing metavalues of the local object	349
Accessing objects' display values	349
Using base objects	350
Calling functions	350
Pre-scripts for use in processes and process steps	351
Using session services	351
Querying configuration parameters	352
Testing for the existence of certain database entries	353
Querying session object global variables	353
Using #LD-notation	354
Using #LD notation in process tracking	356
Example of specifying the language or language variant	357
Script library	358
Support for processing scripts in the Script Editor	359
Creating and editing scripts in the Script Editor	365
Copying scripts in the Script Editor	366
Testing scripts in the Script Editor	367
Testing script compilation in the Script Editor	368
Overriding scripts	368
Permissions for running scripts	369
Editing and testing script code with the System Debugger	370
Loading the script library	370
Tips for editing script code in the System Debugger	372
Logging database queries and object actions	372

Testing script code in the System Debugger	373
Testing scripts in the System Debugger	373
Testing templates and formatting scripts in the System Debugger	374
Testing methods in the System Debugger	375
Testing table scripts in the System Debugger	375
Saving changes to the database	376
Extended debugging in the Object Browser	376
Creating local debug assemblies	377
Debugging in the Object Browser	378
Troubleshooting debugging in the Object Browser	379
One Identity Manager query language	381
Language elements of the One Identity Manager query language	381
Comments	382
Identifier	382
Literal values	383
String values	383
Integer values	383
Decimal values	384
Date and time values	384
Parameter references	385
Preformatted Where clauses	385
Formulating queries in the One Identity Manager query language	386
Query header	386
Where clauses	387
Search clauses	387
Select clauses	387
Order by clauses	389
Paging clauses	389
Display value clauses	390
Query parameter clauses	391
Query hints	392
Conditions	392
Special conditions	393
Comparing columns	394
Comparison by means of IN and NOT IN clauses	395

Compare date differences	396
Compare date ranges	396
Compare fixed values	397
Comparing parameters	398
Using preformatted Where clauses	398
Reports in One Identity Manager	399
Working with the Report Editor	399
Menu items in Report Editor	400
Views in the Report Editor	401
Report Editor program settings	402
Logging database queries	403
Creating and editing reports in the Report Editor	404
Editing general report properties	406
Creating and editing data sources	407
Data retrieval using SQL queries	408
Data retrieval using database views	409
Data retrieval using an object	410
Data retrieval using single object history	411
Data retrieval using multiple object history	413
Data retrieval using historical assignments	415
Data query for simulation data	417
Report parameters	418
Editing report parameters	420
Editing general parameter settings	421
Editing parameter value definitions	422
Settings for calculating values	424
Using virtual data sources	425
Editing the report form	425
Adding data fields to report forms	426
Tips for entering dates in reports	427
Example of a simple report with data grouping	428
Translating reports	432
Embedding reports in the user interface	433
Generating and exporting reports on a cyclical basis	434

Adding custom tables or columns to the One Identity Manager schema	435
Creating new tables	436
Extending tables	438
Defining columns	438
Creating simple columns	439
Creating foreign key columns	440
Creating dynamic foreign keys	441
Creating new columns for database views with type view	442
Advanced configuration of columns	443
Creating database views with read-only type	445
Using Common Table Expressions in read-only database views	447
Creating database views with Union type	447
Creating new assignment tables	448
Creating indexes	450
Removing custom schema extensions	450
Possible error messages due to custom schema extensions	451
Permissions for schema extensions	452
Change labels for the schema extensions	452
Adding schema extensions to the database	453
Recommendations for advanced configuration of custom schema extensions	454
Managing custom database objects within the database	456
Web service integration	458
Binding a web service	459
Generic web service call	459
Direct web service call	461
Self-defined web service call	461
Creating web service solutions with the Web Service Integration Wizard	462
Modifying a web service solution	466
Deleting a web service solution	467
One Identity Manager as SCIM 2.0 service provider	468
Endpoints and base URL	468
SCIM plugin features	469
Authenticating SCIM clients	469
Authenticating SCIM plugins in One Identity Manager	470

Special features of generating the SCIM schema	470
SCIM plugin requests	472
SOAP Web Service	474
Installing and configuring the SOAP Web Service	475
Installing the SOAP Web Service	476
SOAP Web Service configuration	479
Displaying the SOAP Web Service's status	481
Uninstalling SOAP Web Service	482
Examples of calls	482
One Identity Manager as SPML provisioning service provider	488
SPML web service	488
Installing and configuring the SPML web service	489
Installing the SPML Web Service	490
Configuring the SPML web service	493
Uninstalling the SPML Web Service	496
Configuring the One Identity Manager schema	496
Preparing the One Identity Manager schema for export to the SPML schema	497
Creating schema files	498
Testing SPML web service functionality	498
Configuring the SPML test front-end	499
Using the SPML test front-end	499
Processing DBQueue tasks	501
DBQueue Processor configuration for test, development, or productive environments	502
Configuring notification behavior for DBQueue Processor initialization	503
Reinitializing the DBQueue Processor	503
Bulk processing in the DBQueue Processor	504
Processing DBQueue Processor tasks by the SQL Server Agent	505
Processing DBQueue Processor tasks by the Database Agent Service	506
Controlling processing of DBQueue Processor tasks	507
Processing DBQueue Processor tasks	508
How the central dispatcher communicates with individual slots	510
Example of communication during processing	511
Appendix: One Identity Manager Service configuration files	513

Jobservice.cfg	513
viNetworkService.exe.config	514
About us	517
Contacting us	517
Technical support resources	517
Index	518

About this guide

The *One Identity Manager Configuration Guide* gives you an overview of the One Identity Manager architecture and the basics of working with objects in One Identity Manager. It describes the structure of the One Identity Manager schema and explains how to customize and extend the One Identity Manager schema to specific requirements.

In addition, it details how to customize the user interface of the administration tools, especially Manager and Launchpad. The guide explains how to extend the user interface navigation, customize forms, create reports, or localize custom captions.

The basic rules for process orchestration are described in the One Identity Manager. It describes how to customize processes to your requirements and your own processes. An explanation of how to configure logging of data changes and information from process handling is also provided. Advanced configuration settings for the Job server One Identity Manager Service are described. Information is also provided on integrating web services, binding a SOAP Web Service and data exchange using SPML.

This guide is intended for end users, system administrators, consultants, analysts, and any other IT professionals using the product.

NOTE: This guide describes One Identity Manager functionality available to the default user. It is possible that not all the functions described here are available to you. This depends on your system configuration and permissions.

Available documentation

You can access One Identity Manager documentation in the Manager and in the Designer by selecting the **Help > Search** menu item. The online version of One Identity Manager documentation is available in the Support portal under [Technical Documentation](#). You will find videos with additional information at www.YouTube.com/OneIdentity.

One Identity Manager software architecture

The basis for the One Identity Manager structure is classic 3-tier architecture. However, in One Identity Manager the object layer (business logic) is shared. This allows high performance gain due to separate time and location processing.

Database layer

The database represents the core of One Identity Manager. It fulfills the main tasks, which are managing data and calculating inheritance. Object properties can be inherited along the hierarchical structures, such as departments, cost centers, location, or business roles. For data management, the database maps managed target systems and ERP structures as well as compliance rules and access permissions.

The database is separated into two logical parts; payload and metadata. The payload contains all the information required to maintaining data, such as information about employees, user accounts, groups, memberships, operating data, approval workflows, attestation, recertification, and compliance rules.

The metadata contains the description of the application data model and scripts for formatting roles and templates or conditional interactions. One Identity Manager's entire system configuration, all the front-end control settings, and the queues for asynchronous processing of data and processes are also part of the metadata.

Recalculation of inheritance is started by the database trigger logic. For this purpose, the triggers place processing tasks in a task list known as the DBQueue. The DBQueue Processor processes these tasks and recalculates inheritance of the respective database objects. A table labeled JobQueue is used to store processing orders that are to be run by the object layer.

A SQL Server or a managed instance in Azure SQL Database is used as the database system.

Object layer

The object layer enables object oriented access to the database data. The VI.DB.DLL generates entities for objects and collections. Entities use external session services for loading (EntitySource) and saving (UnitOfWork) data objects. Save operations are grouped

so that several data objects can be saved in bulk. The default events Insert, Update, and Delete are available for each object and can be generated after objects are saved.

One or more processing logics are assigned to each entity (EntityLogic). These combine operations that can be run for a particular entity. Separate customizers were developed for the various entities. A customizer is an EntityLogic that deliver specific behavior for an entity. Customizers run processing logic which would normally be implemented in the object code, such as mutual exclusion of properties.

A value template can be assigned to each of the generated object's properties. Templates are implemented for generating user data or for transforming values. You can use templates to fill object properties with default values or to form property values from other properties of the same or other objects.

One Identity Manager uses processes for mapping business processes. A process consists of process steps that represent processing tasks and are joined by predecessor/successor relations. This functionality allows flexibility when linking actions and sequences to object events. Processes are modeled using process templates. A process generator (Jobgenerator) is responsible for converting script templates in processes and process steps into a concrete process in the 'Job queue'.

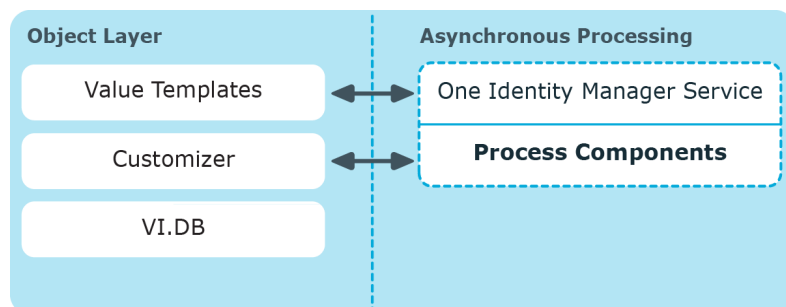
The One Identity Manager Service enables the distribution throughout the network of information that is administrated in the One Identity Manager database. The One Identity Manager Service performs data synchronization between the database and any connected target systems and runs actions at the database and file level.

The One Identity Manager Service retrieves process steps from the Job queue. Process steps are run by process components. The One Identity Manager Service also creates an instance of the required process component and transfers the process step parameters. Decision logic monitors the performance of the process steps and determines how processing should continue depending on the results of the run process components. The One Identity Manager Service enables parallel processing of process steps because it can create several instances of process components.

The One Identity Manager Service is the only One Identity Manager component authorized to make changes in the target system.

Strictly speaking, the One Identity Manager Service is part of the object layer because it does not contain any business logic. The One Identity Manager Service provides help for realizing asynchronous processing.

Figure 1: One Identity Manager object layer



Presentation layer

The presentation layer comprises front-ends that are used to input and output data. There are different front-ends for different tasks. For example, a different front-end is used to configure One Identity Manager than that for managing employee data. The contents to be displayed and the extent to which it can be altered is determined in conjunction with the access permissions of the respective user through the object layer. Available front-end solutions are both client and browser-based.

Clients connect to an application server storing business logic. The application server provides a connection pool for accessing the database and ensures a secure connection to the database. Clients send their queries to the application server, which processes the objects, for example, by determining values using templates and sending the results back to the clients. The data from the application is sent to the database when an object is saved.

Clients can alternatively work without external application servers by retaining the object layer themselves and accessing the database layer directly. In this case, only the part of the object layer that is required for the acquisition process is mapped in the clients.

To implement browser-based user interfaces, there is an application running on a web server that is based on a website render engine. Users use a web browser to access the website that has been dynamically set up and customized for them. Data exchange between database and web server can take place either directly or through the application server.

Figure 2: Layer distribution with application server

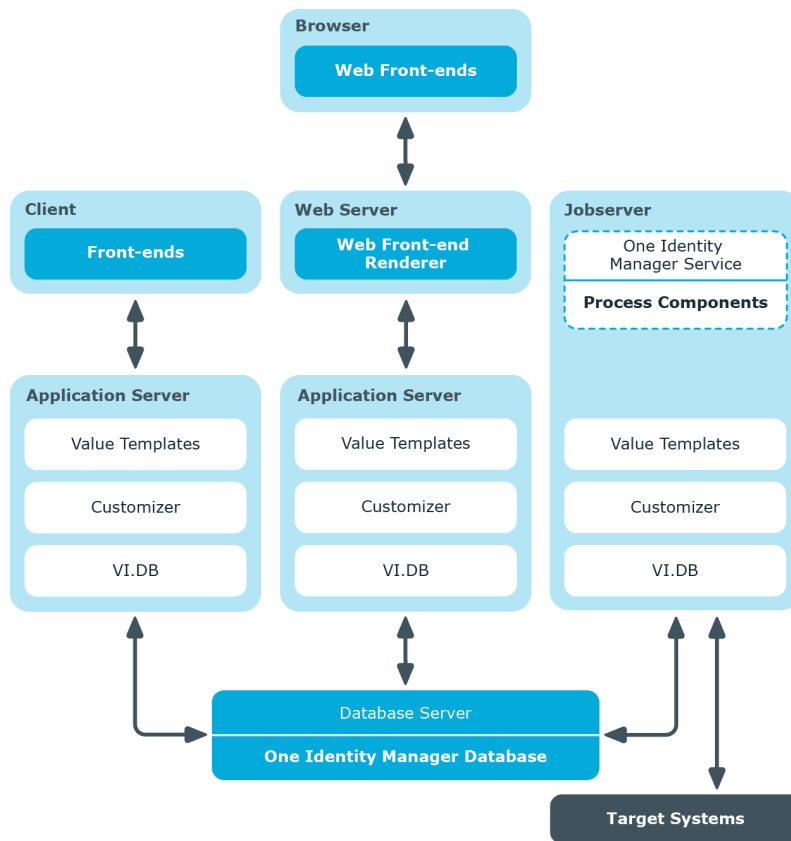
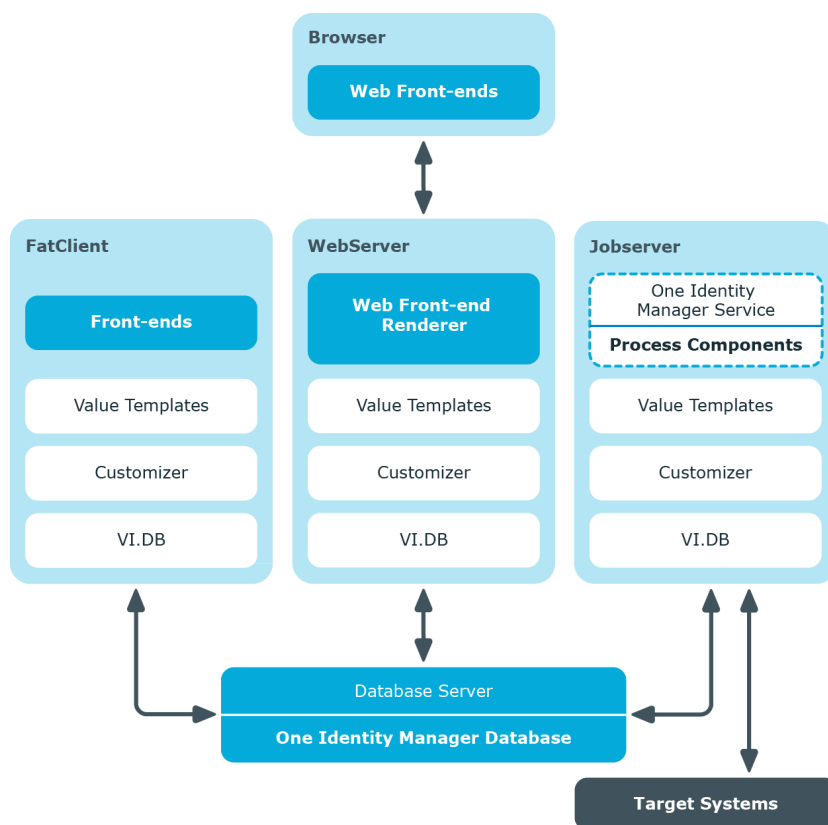


Figure 3: Layer distribution without application server



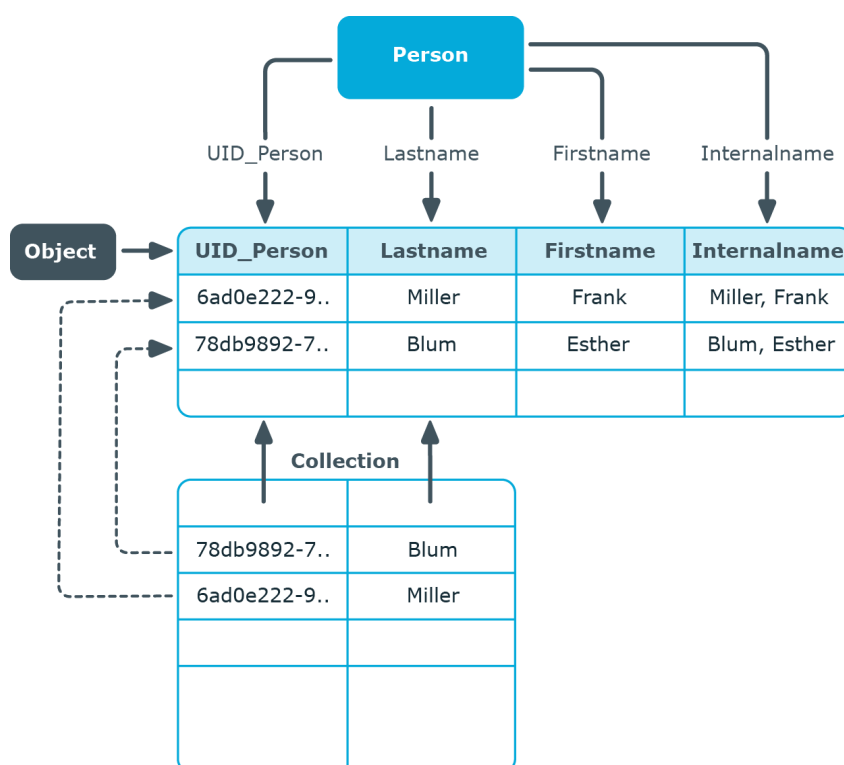
Related topics

- [Working with objects in One Identity Manager](#) on page 22
- [Inserting, modifying, and deleting an object in One Identity Manager](#) on page 25

Working with objects in One Identity Manager

Object oriented access to tables and data sets is done through the One Identity Manager object layer.

Figure 4: Access to tables and data sets



The following applies to this:

- Object class - table
- Properties - columns
- Object - target
- Collection - number (1-n) of columns in a table with several lines.

Objects and collections are mapped using entities. Entities are those data units that can be called from the database and saved to the database. An entity corresponds to a row in a table in the database. It contains data columns and some metavalues such as display values and permissions.

Entities can contain either some or all columns in a table. In the first case, these are flagged by the `IsPartial` property and cannot be changed.

There are three types of entities:

- Read-only
Data values can only be read. You cannot save the entities.
- Delayed logic
You can change and store the entities. The delayed logic mode runs all business logic rules and methods when saving the entity. If the entity runs with an application server, it exists on the client side and does not use server resources.
- Interactive

You can change and store the entities. The underlying logic is applied immediately after a value is changed. The entities' primary application is in user interfaces, where users want to see the business logic directly. For an entity to be able to run the logic without restriction with the user's permissions, it must exist on the application server if it is not run directly with a database.

The entities have the following default methods for performing the database operations.

Table 1: Default methods

Method	Description
EntitySource	Creating new objects and collections or loading objects and collections
UnitOfWork	Grouping together save operations of multiple objects and collections
discard	Discarding of objects
MarkForDeletion	Marking objects to be deleted Not deleted until saved.

When an object is loaded, all the columns are loaded. When a collection is loaded, not all the columns are loaded for performance reasons. The primary key columns are loaded and all columns that are in the display template and those where an object is marked for deletion. Defined display templates specify how each collection object is displayed in the front-end. Defaults for each table's display template are stored in the One Identity Manager schema and can be customized.

Objects recognize the following default events, which can be generated as a result of saving.

Table 2: Default events of the objects

Event	Description
Insert	Insert an object.
Update	Change an object.
Delete	Delete an object.
Assign	Add M:N assignments.
Remove	Remove M:N assignments.

Processes can be linked to these events that run actions in different target systems, for example, to add user accounts, add a home directory on a server, or write data to the One Identity Manager database.

Table 3: Lifecycle of an object

Front-end action	Object state	Event on saving	Database action
Insert an object.	Object does not exist.	Insert	UID is created and the object is added to the database.
Change properties.	Object exists in the database and is loaded.	Update	Object properties are changed.
Delete object.	Object exists in the database and is loaded.	Delete	<p>For objects that have the Marked for deletion (XMarkedForDeletion) property:</p> <ul style="list-style-type: none">• The MarkForDeletion method is run. Objects are locked and cannot be modified.• If deferred deletion > 0 days is configured, a deferred operation is created for deletion. The objects are initially disabled. During the retention period, you have the option to restore the objects. If a deleted object is restored, the object properties are reset to their state before deletion. The objects are finally deleted when the deferred deletion time period has expired.• Object with deferred deletion on 0 days are deleted immediately. <p>Objects that do not have the Marked for deletion property are immediately deleted.</p>

Related topics

- [Inserting, modifying, and deleting an object in One Identity Manager](#) on page 25

Inserting, modifying, and deleting an object in One Identity Manager

All actions in One Identity Manager are run over the object layer and saved in the One Identity Manager database. Each change to an object (insert, change, delete) is run within a transaction. Another fixed item in a transaction of this type creates the processes itself.

The transaction can only be successfully completed if the changes are saved and the processes have been successfully generated. If errors occur within the transaction, the entire transaction is rolled back.

The following is an example of how to insert an object in One Identity Manager.

Run the following steps in the front-end:

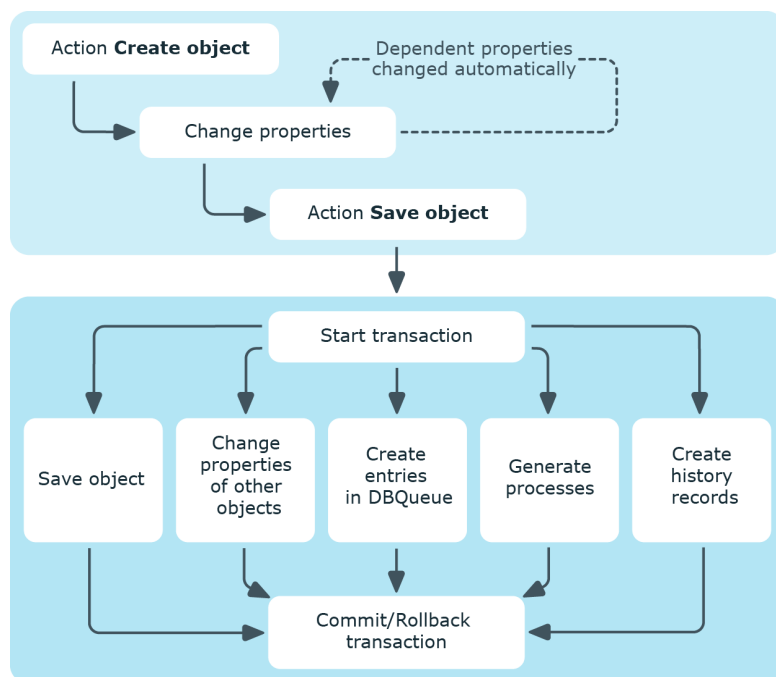
- Insert a new object.
- Enter the object properties.
Dependent properties within this object are created with templates. Side effects implemented in the Customizer, such as mutual exclusion of certain properties, are applied.
- Save the object.

After saving the object in the front-end, run the following steps in the object layer:

- Start a transaction (Begin Transaction).
- The following steps are processed in parallel:
 - Save the object in the database.
 - Apply the templates and formatting scripts to dependent objects.
 - Generate processing tasks for the One Identity Manager Service in the Job queue.
 - Generate processing tasks for the DBQueue Processor in the DBQueue.
 - Generate a record of changes in the history.
- The transaction ends with success (Commit Transaction) or changes are rolled back if an error occurs (Rollback Transaction).

The following figure shows the flow of data when an object is inserted.



Figure 5: Dataflow inserting an object



Customizing the One Identity Manager default configuration

You can customize large parts of the One Identity Manager default configuration. For example, you can specify your own display names for columns or menu items or define your own templates and formatting rule for column values.

If you customize a default configuration, the change is captured by a trigger and the default configuration is copied into a configuration buffer. You can retrieve changes from the configuration buffer and restore the default configuration in this way.

- Changes to data are labeled with the  icon in front of the modified value. As long as the changes have not been saved, you can restore them by clicking the icon.
- Changes to the default configuration are labeled with the Designer icon in the . To restore the default configuration, click the icon.

If the default configuration is changed by a service pack, a complete version upgrade or by loading a hotfix package during a One Identity Manager version upgrade, a check is made to see if it has already been customized. In this case, the modified default configuration is copied to the configuration buffer. This ensures that customizations do not go missing.


Related topics

- [Reloading changes dynamically](#) on page 28
- [Locking and unlocking individual properties for editing](#) on page 30
- [System configuration reports](#) on page 31

Reloading changes dynamically

Cached system data can be dynamically reloaded if it has changed. The changes are reloaded automatically in background.

An exception to this are changes that effect the character of the user interface. These changes are only reloaded after requesting confirmation from the user. The user can decide

when to accept these changes. In the status bar of the Manager, the  icon indicates that the user interface was modified.

The semaphore is incremented when changes are made. The semaphore is calculated when the DBQueue Processor is run.

To configure the reloading of changes

1. In the Designer, check if the **Common | CacheReload** configuration parameter is set. Otherwise, set the configuration parameter and compile the database.
2. Use the **Common | CacheReload | Type** configuration parameter to specify the method for checking the validity of cached information. Permitted values are:
 - **ALWAYS**: The validity of the cached information is checked during every access.
 - **NEVER**: The validity of the cached information is never checked.
 - **TIMER**: The validity of the cached information is checked on expiry of the interval.
3. If you use the **TIMER** method, specify the time in seconds in the **Common | CacheReload | Interval** configuration parameter after which the values are to be checked when they are accessed.

Which columns are reloaded is defined in the data model. In the Designer, you can find an overview of the semaphore in the category **Base data > Advanced > Semaphore**.

- To reload data after changes to a column, the column must be assigned to the semaphore.
- To reload data after inserting or deleting in a table, the primary column key must be assigned to the semaphore.

Table 4: Changes to reload

Changes	Semaphore
Script assembly and Customizer	Assembly
Calculate column dependencies	BulkdDependencies
Names, such as column headings or display text	Caption
Configuration parameter	Config
Countries and time zones	Country
Parts of user interface	Dialog
Use of special program functions	Feature
Icons	Image
Tables, columns, table relations, column relations, objects, tasks	Model
Notification	Notification

Changes	Semaphore
Permissions and permissions groups	Right
Software revisions status (for software update)	SoftwareRevision
Statistic definitions	DashBoardDef
Statistical content	DashBoardContent
Module dependencies	ModuleDepend
User data stored in memory.	UserDataResident
Changes to synchronization configuration	DPRConfiguration
Changes to module dependencies	ModuleDepend
Changes to the Web Portal configuration	AEDS
Changes to predefined SQL queries	LimitedSQL
Changes to permissions for Web API methods	AEDSGROUP
Changes to password policies	PasswordPolicy

Locking and unlocking individual properties for editing

You can prevent individual properties from being overwritten by transports or normal editing using a lock.

For example, you may want to block processing, as follows:

- Configuration parameters and their values should not be overwritten when a test environment is transported to a productive system.
- Server configurations should neither be overwritten in the test environment nor the productive system during a transport.

NOTE: To lock properties for editing, users require the **Allow setting a change lock for specific properties of individual objects** program function (Common_AlowPropertyLocks).

If certain users are allowed to lock properties for editing, you can assign the permissions to the users through permissions groups.

- The **QBM_PropertyLock** permissions group is provided for non role-based login. This group owns the program function. Add the system users to the permissions groups. Administrative system users automatically obtain these permissions groups.

- The **QER_4_PropertyLock** permissions group is provided for non role-based login. This group owns the program function. The permissions group is linked to the **Base roles | Lock single properties** application role. Add the employees to the application role.

To unlock and unlock a single property

1. Open the object in the Designer or the Manager.
2. Click the property name and select one of the following options from the context menu:
 - **Prohibit modification:** The property is locked for editing. The input field is locked and grayed-out.
 - **Permit modification:** The property is unlocked and available for editing.

System configuration reports

In the Designer, different reports about the system configuration and customizations are available. When you select an entry in this category the corresponding report is generated. Generating the report may take some time depending on its size.

To display a report about the system configuration

- In the Designer, select the **Documentation** category.

Table 5: System configuration reports

Report	Contents
System configuration	This report contains the description and settings of enabled configuration parameters.
Processes	This report contains the description of all enabled default processes. The process steps and their parameters as well as the scripts used and configuration parameters for a process are listed.
Process Components	The report contains the description of all process components with their tasks and parameters.
Templates	This report contains the descriptions of all default templates including affected columns, scripts used and configuration parameters.
Formatting rules	This report contains the description of all default formatting rules including scripts used and configuration parameters.
Scripts	This report contains the description of all default scripts including configuration parameters used. The usage in processes, process steps, templates, formatting rules and scripts is listed for each script.

Report	Contents
TimeTrace	The report shows the configuration of the TimeTrace.
Full report	Full report about system configuration. The report summarizes the information contained in the partial reports.

Table 6: Reports available for customizing

Report	Contents
System configuration	This report contains the description and settings of enabled configuration parameters.
Processes	This report contains the description of all enabled default processes. The process steps and their parameters as well as the scripts used and configuration parameters for a process are listed.
Templates	This report contains the descriptions of all default templates including affected columns, scripts used and configuration parameters.
Formatting rules	This report contains the description of all default formatting rules including scripts used and configuration parameters.
Scripts	This report contains the description of all default scripts including the configuration parameters used. Process usage, process steps, templates, formatting rules and scripts are listed for each script.
One Identity Manager schema	This report contains the description of custom One Identity Manager schema extensions (tables and columns). In addition, information about customized database objects is also listed, such as database procedures, functions, triggers, or view definitions.
Full report	Full report about system configuration. The report summarizes the information contained in the partial reports.

Customizing the One Identity Manager base configuration

The base data includes the main settings for configuring One Identity Manager. They are usually checked and customized on a one-off basis before the system goes into operation. The base data contains the database connection data, authentication module usage, languages used or the configuration parameter settings.

Related topics

- [Overview of the database settings](#) on page 33
- [Changing database connection data](#) on page 36
- [Database configuration for a test, development, or productive environment](#) on page 37
- [Changing the database staging level](#) on page 38
- [Language settings for displaying and maintaining the data](#) on page 38
- [Setting login languages](#) on page 39
- [Default country for determining working hours and public holidays](#) on page 39
- [Configuration parameters for system configuration](#) on page 40
- [Editing configuration parameters](#) on page 40
- [Creating custom configuration parameters](#) on page 41
- For more information about the authentication modules, see the *One Identity Manager Authorization and Authentication Guide*.

Overview of the database settings

NOTE: Changes to the data are not usually necessary and should only be made by advanced users.

To display database information

1. In the Designer, select the **Base Data > General > Databases** category.
2. Select the database in the List Editor.
3. The following information appears:

Table 7: Database information

Property	Meaning
Main database	Identifies the database as the main database. The One Identity Manager database is marked with this option when the schema is installed the first time.
Customer	Name of the customer.
Description	Description of database.
Customer prefix	Customer ID for prefix. The customer prefix is used to create and transfer customized scripts, processes, and extensions to the One Identity Manager schema.
Module owner	Module owner ID for prefix. The prefix is used to create and transfer customized scripts, processes, and extensions to the One Identity Manager schema.
Staging level	Specifies whether the database is a test, development, or production database. The permitted values are Development system , Test environment , and Production system .
Custom staging level	Detailed information about staging levels. This information is shown in the status bar of the programs in the database connection tooltip and in the installation overview in the Launchpad.
Status bar color	The color of the status bar can be displayed in a different color to the layout depending on the staging level. The color can be defined by template and customized. The following colors are defined as default: <ul style="list-style-type: none">• None - development system database is connected.• Green - test environment database is connected.• Yellow - production system database is connected.
Last compiler relevant configuration date	Date and time of the last compiler relevant modification. If the value is changed the database has to be recompiled.
Simulation started	Time at which the last front-end simulation was started.
Stop DBQueue Processor	If this option is set for the main database, the DBQueue Processor does not process any more tasks. You can stop and start the DBQueue Processor with the appropriate administrative permissions in Job Queue Info.

Property	Meaning
	For more information, see the <i>One Identity Manager Process Monitoring and Troubleshooting Guide</i> .
Stop One Identity Manager Service	<p>If this option is set for the main database, the One Identity Manager Service does not process any more tasks. You stop and start the service with the appropriate administrative permissions in Job Queue Info.</p> <p>For more information, see the <i>One Identity Manager Process Monitoring and Troubleshooting Guide</i>.</p>
provider	VI.DB.ViSqlFactory,VI.DB is entered for the connection to the SQL server.
Connection parameter	Login data for the database user, database server, and the database. The data is entered into the database during schema installation.
Authentication module	<p>The default authentication for logging in to the database.</p> <p>For more information about One Identity Manager authentication modules, see the <i>One Identity Manager Authorization and Authentication Guide</i>.</p>
Language (default)	The default language. Fallback alternative for displaying language-dependent text.
Country (default)	The default country. The country that is taken into account when determining working hours and public holidays.
Edition	Name of the installed edition.
Edition version	Version number of the edition.
Edition description	Detailed description of the edition.
Database ID	Identifier for the database. The database ID is taken from the original database server and database data. The database ID has to be recalculated if a database is created from a database backup on another server. When a database is compiled, the database ID is checked and changed if necessary.
Single-user mode process	<p>Process requiring single-user mode. If the value 0, a single-user mode is not required.</p> <p>NOTE: The DBQueue Processor checks at regular intervals whether the single user mode is still required and resets the setting if necessary.</p>
Single-user mode start time	<p>Time of the request for single-user mode.</p> <p>NOTE: The DBQueue Processor checks at regular intervals whether the single user mode is still required and resets the</p>

Property	Meaning
	setting if necessary.
Public key for encryption	The public key is entered by the Crypto Configuration program and is needed for database encryption. For more information about database encryption, see the <i>One Identity Manager Installation Guide</i> .
Preparation phase for migration	<p>Progress display for step-by-step preparation of migration. Possible phases are:</p> <ul style="list-style-type: none"> • Green: The database is in normal operating mode. • Yellow: All database users will be informed about the upcoming update. The system does not accept anymore processes. The preparation phase is displayed in the program's status bar. • Orange: New users cannot log in to the database anymore. All running processes will still be processed. The preparation phase is displayed in the program's status bar. • Red: The database is ready for updating. The preparation phase is displayed in the program's status bar. • Red/green: The processes queued by migration are in the final stage of processing.

Related topics

- [Changing database connection data on page 36](#)
- [Database configuration for a test, development, or productive environment on page 37](#)
- [Language settings for displaying and maintaining the data on page 38](#)
- [Default country for determining working hours and public holidays on page 39](#)

Changing database connection data

The One Identity Manager database connection data is set up by the initial schema installation. This information is also accessed when tasks are generated for the One Identity Manager Service.

NOTE: Changes to the data are not usually necessary and should only be made by advanced users.

To change the connection parameter

1. In the Designer, select the **Base Data > General > Databases** category.
2. In the List Editor, select the database.
3. Select the **Define connection string for database** task.
4. Enter the connection data for the database.
 - **Server:** Database server.
 - (Optional) **Windows Authentication:** Specifies whether the integrated Windows authentication is used. This type of authentication is not recommended. If you decide to use it anyway, ensure that your environment supports Windows authentication.
 - **User:** The user's SQL Server login name.
 - **Password:** Password for the user's SQL Server login.
 - **Database:** Select the database.
5. Click **OK**.
6. Select the **Database > Save to database** and click **Save**.

Database configuration for a test, development, or productive environment

You use the staging level of the One Identity Manager database to specify whether the database is a test database, development database, or a live database. A number of database settings are controlled by the staging level.

If you change the database's staging level, the following settings are configured.

Table 8: Default settings for development, test, and live environments

Setting	Development environment	Test environment	Live environment
Color of the One Identity Manager tools status bar	None	Green	Yellow
Maximum DBQueue Processor runtime	20 minutes	40 minutes	120 minutes
Maximum number of slots for the DBQueue Processor	5	7	Maximum number of slots according to the hardware configuration

Related topics

- [Changing the database staging level](#) on page 38
- [DBQueue Processor configuration for test, development, or productive environments](#) on page 502

Changing the database staging level

To modify a database staging level

1. In the Designer, select the **Base Data > General > Databases** category.
2. In the List Editor, select the database.
3. In the edit view, select the **General** tab.
4. Change the value of the **Staging level** property to **Test environment**, **Development system**, or **Production system**.
5. Confirm the security prompt with **Yes**.
6. Select the **Database > Save to database** and click **Save**.

Related topics

- [Database configuration for a test, development, or productive environment](#) on page 37
- [DBQueue Processor configuration for test, development, or productive environments](#) on page 502

Language settings for displaying and maintaining the data

The default One Identity Manager installation is supplied in the **English - United States [en-US]** and **German - Germany [de-DE]** language. You can add other languages to the user interface and display text if required. In this instance, you must translate the text before One Identity Manager goes live. There is a Language Editor in the Designer to help you do this. A special control is provided in the One Identity Manager tools that aids multi-language input.

One Identity Manager default language

Maintenance of default data takes place in the default language. The default language for an installation of One Identity Manager is **English - United States [en-US]**. The default language is valid across the system. It is not recommended to change the default language during working hours.

In the ideal case, the One Identity Manager language matches the user's administration tool login language. If these two settings are different, then the default language is used if no captions are found in the requested login language for a set of language-dependent data.

User login language

The language used in the user interface is the same as the language used when logging in to the administration tools. When you log in for the first time, the system language is used for displaying the user interface. Users can change their login language in the program settings in all administration tools. This sets the language globally for all the user's tools. Therefore, the user does not have to set the login language in every tool separately. Changes to the login language take effect after the tool is restarted.

Any language for which the **Select in front-end** option is activated can be used as a login language.

Related topics

- [Setting login languages](#) on page 39
- [Language-dependent data representation](#) on page 210

Setting login languages

Any language for which the **Select in front-end** option is activated can be used as a login language.

To enable an additional login language

1. In the Designer, select the **Base data > Localization > Languages** category.
2. In the List Editor, select the language.
3. In the **Properties** view, set the **Select in front-end** property to **True**.
4. Save the changes.
5. Select the **Database > Save to database** and click **Save**.

Default country for determining working hours and public holidays

An employee's working hours need to be made public in order to determine the reaction times of approvers or attestors to request processes in the IT Shop or during attestation. For more information, see the *One Identity Manager Identity Management Base Module Administration Guide*.

If the system cannot determine an employee's country, the default country entered in the database is used as the default.

To specify a default country

1. In the Designer, select the **Base Data > General > Databases** category.
2. In the List Editor, select the main database.
3. In the edit view, select the **Settings** tab.
4. In the **Country (default)**, select the country.
5. Select the **Database > Save to database** and click **Save**.

Configuration parameters for system configuration

Use configuration parameters to configure the behavior of the system's basic settings. One Identity Manager provides default settings for different configuration parameters. Check the configuration parameters and modify them as necessary to suit your requirements.


Configuration parameters are defined in the One Identity Manager modules. Each One Identity Manager module can also install configuration parameters. In the Designer, you can find an overview of all configuration parameters in the **Base data > General > Configuration parameters** category.

Detailed information about this topic

- [Editing configuration parameters](#) on page 40

Editing configuration parameters

The configuration parameters supplied and their permitted values are maintained by the schema installation. You cannot edit the properties of these parameters. You can set or unset them and specify the actual value for the parameter. Other properties of predefined configuration parameters cannot be edited. Changing a configuration parameter can result in calculations for the DBQueue Processor.

TIP: In the Designer, in the hierarchical view of the Configuration Parameter Editor, modified configuration parameters are framed in yellow .

To edit configuration parameters

1. In the Designer, select the **Base data > General > Configuration parameters** category.
2. Select the configuration parameter in the Configuration Parameter Editor.

3. In the **Configuration parameter** view, select the **Properties** tab.
4. Customize the following settings.
 - **Enabled:** Specifies whether the configuration parameter is set. To set the configuration parameter, select the check box. To unset the configuration parameter, deselect the check box.
 - **Value:** Value of the configuration parameter.

TIP: You can edit the values of some configuration parameters in the Configuration Parameter Editor in a separate wizard. Click ... next to the input field to start the wizard.
5. Select the **Database > Save to database** and click **Save**.

IMPORTANT: Compile the database if the configuration parameter is preprocessor relevant.

Related topics

- [Creating custom configuration parameters](#) on page 41
- [Preprocessor-relevant configuration parameters](#) on page 336

Creating custom configuration parameters

If it is necessary to define other custom configuration parameters, you can add them below the **Custom** configuration parameter.

To set up a new configuration parameter

1. In the Designer, select the **Base data > General > Configuration parameters** category.
2. Select the **Custom** configuration parameter and use the **Insert** context menu to insert a new configuration parameter.
3. In the **Configuration parameter** view on the **Properties** tab, edit the main data of the configuration parameter.
4. (Optional) If a configuration parameter only permits certain values, specify the permitted values on the **Options** tab.
 - To create a new option, click **Insert**.
 - To delete an option, click **Delete**.
5. Select the **Database > Save to database** and click **Save**.

Related topics

- [Editing configuration parameters](#) on page 40
- [Configuration parameter properties](#) on page 42

- [Configuration parameter options](#) on page 43
- [Preprocessor-relevant configuration parameters](#) on page 336

Configuration parameter properties

Table 9: Configuration parameter properties

Property	Description
Full name	Full name of the configuration parameter. This consists of the name of the parameter and the name of the parent parameter.
Parameters	Technical name of the configuration parameter.
Display name	<p>The display name supplies the caption for the configuration parameter. The display names can be stored as language-dependent.</p> <p>NOTE: To show the display names in Configuration Parameter Editor, select the Configuration parameter > Show captions menu item.</p> <p>Configuration parameters that do not have a display name are displayed in brackets (<<>>) in this mode. In addition, a tooltip with the technical name is displayed.</p>
Sort order	<p>The sort order affects how the configuration parameters are ordered in the Configuration Parameter Editor.</p> <p>NOTE: The sort order is only effective if the display names are displayed in Configuration Parameter Editor.</p>
Value	<p>Value of the configuration parameter. You must enter a value for every configuration parameter. Even parent configuration parameters that serve no purpose other than providing a structure must not be empty, otherwise the child configuration parameters cannot be accessed. Some configuration parameters have several permitted values. These are specified using the configuration parameter options and can be selected here. A description of the selected option is also shown.</p> <p>TIP: You can edit the values of some configuration parameters in the Configuration Parameter Editor in a separate wizard. Click ... next to the input field to start the wizard.</p>
Description	Description of the configuration parameter. In the Configuration Parameter Editor, click Edit to edit the description.
Enabled	Specifies whether the configuration parameter is set. If this option is set, the configuration parameter is set. If this option is not set, then the whole tree from this point on is considered disabled and the configuration parameter and its child parameters are considered not to exist.
Encrypted	Configuration parameters are marked with this option when they contain encrypted data, for example, passwords. When a new value is entered it

Property	Description
	is therefore encrypted immediately.
Preprocessor relevant parameter	Specifies whether this is a preprocessor relevant configuration parameter. A preprocessor statement is entered in the associated option field that is used for conditional compiling. NOTE: When a preprocessor relevant configuration parameter is set it is valid globally across the system. The preprocessor condition does not come into effect until the database has been compiled. Every time a preprocessor relevant configuration parameter or its option is changed the database needs to be recompiled.

Related topics

- [Editing configuration parameters](#) on page 40
- [Creating custom configuration parameters](#) on page 41
- [Configuration parameter options](#) on page 43
- [Preprocessor-relevant configuration parameters](#) on page 336

Configuration parameter options

If a configuration parameter only permits certain values, these values are defined in the configuration parameter options.

Table 10: Option properties

Property	Description
Value	Value permitted for the configuration parameter.
Description	Description of the configuration parameter option.
Preprocessor expression	Preprocessor relevant configuration parameters as assigned a valid preprocessor expression in the options. This can be used as a preprocessor condition for conditional compiling.

Related topics

- [Editing configuration parameters](#) on page 40
- [Creating custom configuration parameters](#) on page 41
- [Configuration parameter properties](#) on page 42
- [Preprocessor-relevant configuration parameters](#) on page 336

One Identity Manager schema basics

The One Identity Manager data model is grouped logically into modules. The modules are linked through predecessor relationships. A module can have one or more predecessors. Each module extends the schema by its own tables and scripts and installs its own default objects, such as its own templates, scripts, or processes.

The functions of a module do not become available until the module is present in the database. For example, attestation functions do not become available until the Attestation Module is present. If the One Identity Manager report functions are going to be used, the report subscriptions module must be present in the database.

The Configuration Module is always present. The Configuration Module contains metadata that describes the application data model and scripts for formatting rules and templates or conditional interactions. One Identity Manager's system configuration, all the front-end control settings, and the queues for asynchronous processing of data and processes are also part of the metadata. The metadata is described by the system data model.

The other modules contain all the information required to maintaining data (payload), such as information about employees, user accounts, groups, memberships, and operating data, approval workflows, attestation, recertification, and compliance rules. The user data is described by the application data model.

The table definitions are stored in the `DialogTable` table. The column definition of all the tables are stored in the `DialogColumn` table. The tables relations and column relations are stored in the `QBMRRelation` and `DialogValidDynamicRef` tables.

Related topics

- [Overview of the One Identity Manager schema](#) on page 45
- [Table types and default columns in the One Identity Manager data model](#) on page 49
- [Notes on editing table definitions and column definitions](#) on page 53
- [Table definitions](#) on page 54
- [Column definitions](#) on page 74
- [Table relations](#) on page 99
- [Dynamic foreign key](#) on page 102

- [Adding custom tables or columns to the One Identity Manager schema](#) on page 435
- [One Identity Manager software architecture](#) on page 18

Overview of the One Identity Manager schema

The data model is mapped and edited in the Designer in the **One Identity Manager Schema** category. This category displays the One Identity Manager default tables and the custom tables including their properties. It gives you an overview of customizations to the default configuration, the value templates and formatting rules of the database columns.

To display the schema overview

1. In the Designer, select the **One Identity Manager Schema** category.
2. Open the schema overview with the **One Identity Manager Schema** task.

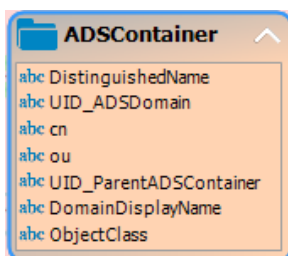
TIP: When you select a table or column in the Designer, you can open the schema overview using the **Show table <table name> in schema** and **Show column <column name> in schema** tasks.

The schema overview has two modes for displaying the One Identity Manager schema.

- Displaying the data model
This mode gives you an overview of all tables including their columns and the table relations.
- Displaying dependencies
This mode only displays those tables that have columns with dependencies due to value templates. Tables and columns without dependencies are not shown.

Tables and their columns are displayed using a special control element. The name of the database table is shown in the header of the control element. All other entries represent columns in the table. Each control element entry has a tooltip. The tooltip content depends on the display mode selected. The column entries are labeled with icons that mark particular properties of the columns depending on the display mode.

Figure 6: Control elements for displaying database tables and their columns



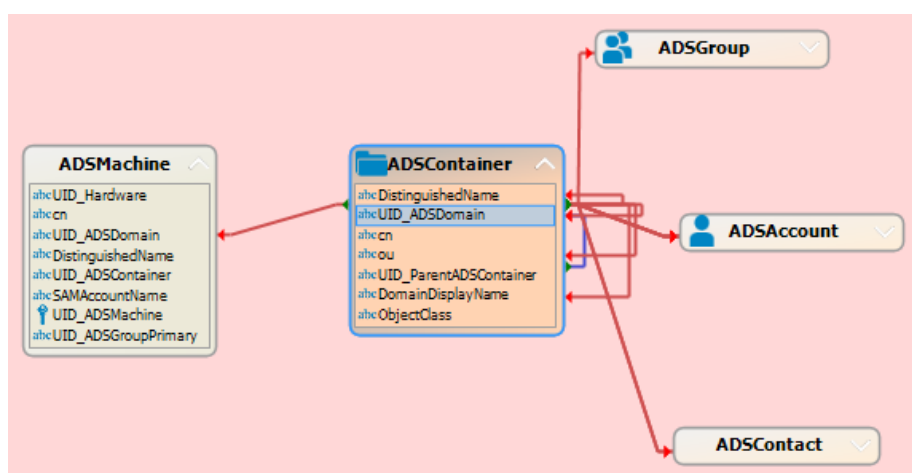
Use the **Options > Show all columns** and **Options > Hide all columns** menu items or the icons in the control element header menu items to control how you display the column entries. Use the **Options > Hide small tables** menu item to only show the name of the table for tables with less than 20 columns.

To display tables and columns that are disabled by preprocessor conditions, use the **Options > Show disabled columns** menu item.

You can change the layout of the control elements in the schema overview with the mouse. Using the **Options > Save table layout** menu item the changes made to the schema layout are saved in the internal database and in the Designer's change log.

Relations between tables or columns are represented by connecting lines. You can control how these are displayed using the **Options > Hide table relations** menu item. If the menu item is disabled all the connectors are shown. If the menu item is enabled then none of the connectors are shown. If a control element is selected the connectors are highlighted anyway without regard to the menu setting.

Figure 7: Using connectors to illustrate relations



A connector points to column entries that are related to it. You can navigate between the connection points using the connector. When you select a connector the cursor changes to an arrow icon. Double-click on the connector to move the view to other end of the connector. The direction is indicated by the arrow icon. Movement is controlled with the **Options > Animate movements** menu item. When the cursor passes over a connector a tool tip, whose contents depends on the display mode, is shown.

You can use the quick overview to navigate faster around the schema view. On the lower right edge of the schema overview there is a button which you use to open the quick overview. The area of the schema overview that is currently shown in the window is marked with a frame in the quick overview. Using the mouse you can move this frame around in the view. The corresponding area of the schema overview is then shown in the window.

Figure 8: Opening the quick overview



Related topics

- [Displaying data models in the Designer](#) on page 47
- [Displaying the column dependencies based on templates](#) on page 48

Displaying data models in the Designer

This mode gives you an overview of all tables including their columns and the table relations.

To display the data model









1. In the Designer, select the **One Identity Manager Schema** category.
2. Open the schema overview with the **One Identity Manager Schema** task.
3. Select the **Options > Data model** menu item.

A table entry's tooltip contains the name of the table and the table's preprocessor conditions. A column entry's tooltip contains the name of the column, description, data type and the minimum and maximum length of the column.

A connector's tooltip shows the table relations. This tooltip contains the name of the tables that are related to it and the table relation properties. A single mouse click on the connector opens the table relation properties in the edit view.

Column entries are marked in the control with icons representing special properties, for example the column's .Net data type.

Table 11: Meaning of the icons

Icon	Meaning
	The column is a foreign key column (FK).
	The column is a primary key column (PK).
	The column has the string or text data type.
	The column has the binary data type.
	The column has the bool data type.
	The column has the int, byte, or short data type.
	The column has the double or decimal data type.
	The column has the date data type.

Related topics

- [Displaying the column dependencies based on templates](#) on page 48

Displaying the column dependencies based on templates

This mode only displays those tables that have columns with dependencies due to value templates. Tables and columns without dependencies are not shown.

To display the column dependencies

1. In the Designer, select the **One Identity Manager Schema** category.
2. Open the schema overview with the **One Identity Manager Schema** task.
3. Select the **Options > Dependencies** menu item.

The tooltip for a table entry contains the name of the table. The tooltip for the column entries contains the name of the column. If a column has a value template it is shown in the tooltip. If the column does not have a value template itself but is referenced by value templates belong to other columns then those columns are named in the tooltip.

When you select a column, the connections to other columns are highlighted in color. A tooltip shows the sender and subscriber relationship of the column dependencies. The tooltip contains the names of tables that refer to each other. The sender, subscriber, and the part of the value template that gives the reason for the dependency are also shown.

Table 12: Meaning of colors for sender subscriber relations

Color	Meaning
Green	Column is sender.
Red	Column is subscriber.

Related topics

- [Displaying data models in the Designer](#) on page 47

Table types and default columns in the One Identity Manager data model

Different types of tables can be used at database level in One Identity Manager.

Table 13: Table types

Table type	Description
Simple table	<p>Simple tables are the most common form for storing data.</p> <p>The following columns are defined for simple tables:</p> <ul style="list-style-type: none">• Primary key, consisting of one column• Object key (XObjectKey)
Many-to-many table	<p>Many-to-many or M:N tables contain the relationships between two other tables.</p> <p>The following columns are defined for many-to-many tables:</p> <ul style="list-style-type: none">• A two column primary key <p>Both columns are defined as foreign key columns on the referenced table.</p> <ul style="list-style-type: none">• Object key (XObjectKey) <p>Many-to-many tables are also called assignment tables in this documentation.</p>
Many-to-all table	<p>Many-to-all or M:all tables are a special type of assignment tables that were developed for One Identity Manager.</p> <p>M:all tables are implemented if part of an assignment (all) can reference different tables, meaning dynamically determined. Valid tables can be limited in this way. For example, the owner of a group can be a user account or a group.</p>

Table type	Description
	<p>Furthermore M:all tables are used if additional information about an assignment is mapped, for example, an assignment's validity period.</p> <p>The following columns are defined for M:all tables:</p> <ul style="list-style-type: none"> • Primary key • Foreign key defined as NOT NULL that references the primary key of another table. • Dynamic foreign key defined as NOT NULL that reference the object key (XObjectKey) of the valid tables. • Object key (XObjectKey) <p>You can define more foreign keys and dynamic foreign keys. These columns must be defined as NULL.</p>
Work tables	Work tables are used to store data for which objects cannot be created. No primary key is required for work tables. However, you can define up to two primary keys.

Table 14: Default columns



Column	Description
Primary key	<ul style="list-style-type: none"> • If objects are generated from the table through the object layer, the table requires a primary key. • If a table represents a many-to-many mapping, a two column primary key is defined. Both primary key columns are defined as foreign key columns in the referenced tables. • No primary key is required for work tables. • Primary key columns must be defined in Globally Unique Identifier (GUID) format. <p>Default GUID's are created in the [0-9,a-f](8-4-4-4-12) format.</p> <p>Predefined module GUID's are mapped in the <MMM>-[0-9,a-f](32) format, where <MMM> corresponds to the module prefix. Custom module GUID's are created in the <CCC>-[0-9,a-f](32) format. For more information, see Working with a globally unique identifier module on page 64.</p>
XObjectKey	If objects are generated from the table through the object layer, the table must have an object key column. The object key (XObjectKey) is a unique key, which is capable of referencing every object in the database.

Column	Description
	<p>XObjectKey syntax:</p> <pre><Key><T>TableName</T><P>PrimaryKeyOfRow</P></Key></pre> <p>with:</p> <ul style="list-style-type: none"> • TableName: table name • PrimaryKeyOfRow: primary key column's GUID <p>An additional <P>SecondPrimaryKeyOfRow</P> is used for two column primary keys. The order in which columns used in the XObjectKey are sorted depends on the foreign key columns identifiers (alphabetical order).</p> <p>Example:</p> <p>PersonInProfitcenter table</p> <pre><Key><T>PersonInProfitCenter</T><P><UID_Person></P><P><UID_Profitcenter</P></Key></pre> <p>PersonInDepartment table</p> <pre><Key><T>PersonInDepartment</T><P><UID_Department></P><P><UID_Person</P></Key></pre>
Foreign key	<ul style="list-style-type: none"> • The name of the foreign key column corresponds, as far as possible, to the name of the references table's primary key. • Foreign key columns are defined in GUID format. • A table is reference through the referenced table's primary key. • If the foreign key column is part of a many-to-all table, the column in the One Identity Manager schema is labeled with the Part of key of many-to-all table option (DialogColumn.IsAllKeyMember).
Dynamic foreign key	<ul style="list-style-type: none"> • Dynamic foreign keys are used if a reference can point to different tables. For example, the manager of a user account (<MMM>Account.ObjectKeyManagertable) can be another user account (<MMM>Account table) or a group (<MMM>Group table). • Dynamic foreign keys reference the (XObjectKey) object key of the permitted tables. • Permitted tables can be limited. All tables are permitted, if there are no restrictions. • A dynamic foreign key is flagged in the One Identity Manager schema with the Dynamic foreign key option (DialogColumn.IsDynamicFK).

Column	Description
	<ul style="list-style-type: none"> If the dynamic foreign key is part of a many-to-all table, the column in the One Identity Manager schema is labeled with the Part of key of many-to-all table option (<code>DialogColumn.IsSmallKeyMember</code>).
XDateInserted	The columns contain information about which users made changes at what times. The columns must always exist together.
XDateUpdated	
XUserInserted	
XUserUpdated	
XTouched	This column contains an element's processing status. The processing status is used for creating custom configuration packages.
XMarkedForDeletion	<p>This column defines whether the object is marked for deletion. The columns exists when:</p> <ul style="list-style-type: none"> The deferred deletion function can be applied to the table. The table is synchronized again a target system and outstanding objects can be handled.
XOrigin	<p>In order to determine the origin of an assignment, a <code>XOrigin</code> column is defined in a many-to-many or a many-to-all table. The individual bit positions provide the origin of a membership.</p> <p>For more information about calculation of assignments, see the <i>One Identity Manager Identity Management Base Module Administration Guide</i>.</p>
XIsInEffect	<ul style="list-style-type: none"> To discover whether an assignment is in effect, a <code>XIsInEffect</code> column is defined on an assignment table. The column only exists if the number of assignments differs from the number of effective assignments. <p>For example, if an employee is deactivated, marked for deletion, or classified as a security risk, inheritance of company resources can be prohibited for this employee. The group assignment is maintained but the assignment has no effect.</p> <ul style="list-style-type: none"> If the <code>XIsInEffect</code> column is used, a <code>XOrigin</code> column must exist. <p>For more information about calculation of assignments, see the <i>One Identity Manager Identity Management Base Module Administration Guide</i>.</p>
XDateSubItem	This column contains the change date for dependencies and is

Column	Description
	<p>required in order to take membership changes in a target system into account during synchronization and provisioning.</p> <p>For more information about synchronizing and provisioning memberships, see the <i>One Identity Manager Target System Synchronization Reference Guide</i>.</p>

Notes on editing table definitions and column definitions

- You can largely customize the tables and schemas from One Identity Manager to your own requirements. In the Designer, edit the tables and columns in the Schema Editor.
 - The default configuration is moved to a configuration buffer during handling. You can retrieve changes from the configuration buffer and restore the default configuration in this way.
 - Changes to data are labeled with the  icon in front of the modified value. As long as the changes have not been saved, you can restore them by clicking the icon.
 - Changes to the default configuration are labeled with the Designer icon in the . To restore the default configuration, click the icon.
 - In the Designer, customized default tables and columns are displayed in the **One Identity Manager Schema > Customized tables** category. The table definitions and column definitions are labeled with an asterisk (*) in the Schema Editor schema. More information about the customizations is shown in a tooltip.
 - The database must be compiled for some changes to tables and columns.
 - Use the One Identity Manager program to add custom tables or columns to the Schema Extension schema. The Schema Extension program creates the schema extensions in the database and ensures that the necessary extensions are made in the One Identity Manager schema.
- You can then make further adjustments to the table definitions and column definitions in the Designer.
- In the Designer, customized tables are displayed in the **One Identity Manager Schema > Customized tables** category.
 - In the Designer, you can get an overview of existing columns with value templates in the **One Identity Manager Schema > Templates** category. Column dependencies due to value templates are mapped in the schema overview in the Schema Editor.

- In the Designer, you can get an overview of the existing columns in the system with predefined formatting types or formatting scripts in the **One Identity Manager Schema > Formatting rules** category.
- In the Designer, reports on system configuration and customizations of tables and columns are provided in the **Documentation** category.

Related topics

- [Customizing the One Identity Manager default configuration](#) on page 28

Table definitions

The One Identity Manager module table definitions are stored in the DialogTable table. Predefined One Identity Manager schema table definitions are maintained through schema installation and only a few properties can be modified.

Use the Designer's Schema Editor to edit One Identity Manager schema table definitions.

Detailed information about this topic

- [Notes on editing table definitions and column definitions](#) on page 53
- [Table types in One Identity Manager](#) on page 54
- [Table scripts](#) on page 63
- [Working with a globally unique identifier module](#) on page 64
- [Specifying deferred deletion for objects](#) on page 67
- [Editing table definitions](#) on page 68
- [Table definition properties](#) on page 68
- [Supporting file groups](#) on page 104

Table types in One Identity Manager

For access through the object layer, the tables in the One Identity Manager schema are labeled with a particular table type. Additional properties are required for the table definition, depending on the table type.

Table 15: Table types in the One Identity Manager schema

Table types	Meaning
Table	The Table table type is used for simple tables, many-to-many tables, M:all

Table types	Meaning
	tables, and work tables.
Base table	The Base table table type is used for simple tables, many-to-many tables, M:all tables, and work tables in order to define database views with the View type. Examples of base tables include the BaseTree table for mapping roles and organizations, and the BasetreeHas* assignment tables for assigning company resources to organizations and roles.
View	The View table type is used for database views on tables with the Base table type. Database views with the View type represent subsets of the underlying tables. Database views with the View type are mainly used to map roles. For example, the database views Department, Locality and Profitcenter are subsets of the Basetree base table.
Proxy	The Proxy table type is used for database views on tables with the Table type or on database views with the View type. Database views with the Proxy type are union views of different tables. Columns are mapped between a database view of the Proxy type and the underlying tables by means of the column definitions and proxy view extensions. Database views with the Proxy type are mainly used for mapping in the Unified Namespace.
Union	The Union table type is used for database views on tables with the Table type or on database views with the View , or Proxy type. Database views with the Union type are union views of different tables and are used to group together different object types with the same context. For example, the QERAccProductUsage database view identifies which service items are used in which IT Shop products. Database views with the Union type are mainly used for editing the user interface and creating reports.
Read only	The Read only table type is used for database views on tables with the Table type or on database views with the View , Proxy , or Union type. Database views with the Read only table type may be subsets or unions of the underlying tables. Database views with the Read only type are for display only and are mainly used for editing the user interface and creating reports.

Related topics

- [Database views of the View type](#) on page 55
- [Database views of the proxy type](#) on page 57
- [Database views of the Union type](#) on page 59
- [Database views of the Read-only type](#) on page 61

Database views of the View type

Database views with the **View** type represent subsets of the underlying tables. Database views with the **View** type are mainly used to map roles. For example, the Department,

Locality, and Profitcenter database views are subsets of the Basetree base table.

Database views with the **View** type are predefined database views. Templates and formatting rules can be defined for columns in these views.

The following information is used to define a database view of the **View** type.

Table 16: Properties for defining a database view of the View type

Property	Meaning
Table	Name of the table in the data model.
Type	View type of table.
Base table	Base table that the view is based on.
Condition for view definition	Restricting condition for creating the database view as a WHERE clause for database queries. The condition relates to the underlying base table.
Columns	A reference is required for each column of the database view to a column in the underlying base column. Make the assignment in the column definition.
Insert values	Default settings for individual columns that are assigned when a new data set is added. The values are entered in VB.Net syntax.
Selection script	Selection script as a VB.Net term, to determine during runtime whether the object passed belongs to the view.

Example:

The Department table is defined as a database view of the **View** type. When you enter data in the Department table, the UID_OrgRoot column should be populated with the **QER-V-Department** value.

Table 17: Example of defining a database view of type "View"

Property	Value
Table	Department
Type	View
Base table	BaseTree
Condition for view definition	UID_OrgRoot = 'QER-V-Department'
Insert values	base.putvalue("UID_OrgRoot", "QER-V-Depart-

Property	Value
	ment")
Selection script	Value = (String.Equals(\$UID_OrgRoot\$, "QER-V-Department", StringComparison.OrdinalIgnoreCase))
Columns --> base columns (excerpt from column definition)	Department.DepartmentName-->BaseTree.Ident_Org Department.Description-->BaseTree.Description
Resulting view definition	create view dbo.Department as select Ident_Org as DepartmentName, Description as Description, ... from BaseTree where UID_OrgRoot = 'QER-V-Department'

Related topics

- [Table definition properties](#) on page 68
- [Column definition properties](#) on page 91
- [Defining insert values](#) on page 132
- [Creating new columns for database views with type view](#) on page 442
- [Database views of the proxy type](#) on page 57
- [Database views of the Union type](#) on page 59
- [Database views of the Read-only type](#) on page 61

Database views of the proxy type

Database views with the **Proxy** table type are union views of different tables. Columns are mapped between a database view of the **Proxy** type and the underlying tables by means of the column definitions and proxy view extensions. The DBQueue Processor calculates the actual view definition from the column mapping. This only takes into account tables that are not disabled by a preprocessor condition. Templates and formatting rules cannot be defined for columns in these views.

Database views of the **Proxy** type are mainly used for mapping the Unified Namespace. For example, the UNSRoot database view is used for mapping of the ADSDomain or LDAPDomain tables in the Unified Namespace.

The following information is used to define a database view of the **Proxy** type.

Table 18: Properties for defining a database view of the proxy type

Property	Meaning
Table	Name of the table in the data model.
Type	Type of Proxy table
Additional view definition	<p>Database query generated as a SELECT statement for setting up the database view. View definition extensions are generated by the DBQueue Processor. The following are taken into account when generating:</p> <ul style="list-style-type: none">• Tables in which the database view is entered as the proxy view• Columns that have a reference to a proxy view column• Columns that are defined as extensions to the proxy view <p>The extensions are linked to each other internally with the Union operator.</p>
Condition for view definition	Restricting condition for creating the database view as a WHERE clause for database queries.
Columns	Database view columns.

Example:

The following mappings are required to map the ADSDomain table in the Unified Namespace to the USRoot database view.

- The UNSRoot database view is entered as a proxy view in the ADSDomain table.
- The columns of the ADSDomain table to be mapped in the Unified Namespace are given a reference to the corresponding columns in the proxy view.
For example, the Ident_Domain column in the ADSDomain table is mapped to the Ident_root column of the UNSRoot proxy view.
- Columns that are expected in the UNSRoot database view but are not contained in the ADSDomain table must be entered in the ADSDomain table as extensions to the proxy view.
For example, the UNSRoot view expects input of the target system type in the UID_DPRNameSpace column. This column is not in the ADSDomain tables. Therefore, 'ADS-DPRNameSpace-ADS' as UID_DPRNameSpace is entered as an extension to the proxy view in the ADSDomain table.

The DBQueue Processor generates the extended view definition from the data. The following statement is a excerpt from the generated extension.

```
select ... Ident_Domain as Ident_UNSRoot..., 'ADS-DPRNameSpace-ADS' as UID_
DPRNameSpace from ADSDomain
```

Related topics

- [Table definition properties](#) on page 68
- [Column definition properties](#) on page 91
- [Database views of the View type](#) on page 55
- [Database views of the Union type](#) on page 59
- [Database views of the Read-only type](#) on page 61

Database views of the Union type

Database views with the **Union** table type are union views of various tables and are mainly used to group various object types with the same context. In the QERAccProductUsage union view, for example, you determine which service items are used in which IT Shop products.

Database views with the **Union** type are predefined database views. Templates and formatting rules cannot be defined for columns in these views. In the view definition, the object key column (XObjectKey) must be referenced. This makes it possible to create a single object with its permitted permissions.

Database views of the **Union** type are mainly used for editing the user interface and creating reports.

The following information is used to define a database view of the **Union** type.

Table 19: Properties for defining a database view of the Union type

Property	Meaning
Table	Name of the table in the data model.
Type	Type of table Union .
Additional view definition	<p>Database query as a SELECT statement for setting up the database view.</p> <p>NOTE: Never select NULL as <Column>. Instead, convert this explicitly to the requested value type.</p> <p>Example:</p> <pre>convert(nvarchar(max), NULL) as <column> convert(varchar(38), NULL) as UID_<column> convert(varchar(138), NULL) as ObjectKey<column></pre> <p>Several extensions for the view definition can be defined. The extensions are linked to each other internally with the Union operator.</p>

Property	Meaning
	When you add a column to a custom table, an entry is created in the DialogColumn table. When you delete a column, the entry is removed from the DialogColumn table. Changes to the schema of default database views are not permitted.
Condition for view definition	Restricting condition for creating the database view as a WHERE clause for database queries.
Columns	Database view columns.

Example:

The QERAccProductUsage table is defined as a database view of the **Union** type. In the union view, you establish which service item is used in which products. The following example shows an excerpt from the definition based on system entitlements (table ESet) and report subscriptions (RPSReport table).

Table 20: Example of defining a database view of Union type

Property	Value
Table	QERAccProductUsage
Type	Union
Columns	TableName, UID_AccProduct, XObjectKey
Extension 1: Additional view definition	ESet
Extension 1: Query	select 'ESet' as TableName, g.XObjectKey, g.UID_AccProduct from ESet g
Extension 2: Additional view definition	RPSReport
Extension 2: Query	select 'RPSReport' as TableName, g.XObjectKey, g.UID_AccProduct from RPSReport g
Resulting view definition	create view dbo.QERAccProductUsage as select * from

Property	Value
	<pre>(select convert(varchar(11), null) as TableName, convert (varchar(38), null) as UID_AccProduct, convert(varchar (138), null) as XObjectKey where 1=0 union all select xxTab.TableName, xxTab.UID_AccProduct, xxTab.XObjectKey from (select 'ESet' as TableName, g.XObjectKey, g.UID_ AccProduct from ESet g) as xxTab union all select xxTab.TableName, xxTab.UID_AccProduct, xxTab.XObjectKey from (select 'RPSReport' as TableName, g.XObjectKey, g.UID_AccProduct from RPSReport g) as xxTab) as x</pre>

Related topics

- [Table definition properties](#) on page 68
- [Column definition properties](#) on page 91
- [Creating database views with Union type](#) on page 447
- [Database views of the View type](#) on page 55
- [Database views of the proxy type](#) on page 57
- [Database views of the Read-only type](#) on page 61

Database views of the Read-only type

Database views with the **Read only** table type may be subsets or unions of the underlying tables. Database view with the **Read only** type are predefined database views. Templates

and formatting rules cannot be defined for columns in these views.

Database views of the **Read only** type are for display only and are mainly used for editing the user interface and creating reports.

The following information is used to define a database view of the **Read only** type.

Table 21: Properties for defining a database view of the Read-only type

Property	Meaning
Table	Name of the table in the data model.
Type	Read only type of table.
Additional view definition	<p>Database query as a SELECT statement for setting up the database view.</p> <p>NOTE: Never select NULL as <Column>. Instead, convert this explicitly to the requested value type.</p> <p>Example:</p> <pre>convert(nvarchar(max), NULL) as <column> convert(varchar(38), NULL) as UID_<column> convert(varchar(138), NULL) as ObjectKey<column></pre> <p>Several extensions for the view definition can be defined. The extensions are linked to each other internally with the Union operator.</p> <p>When you add a column to a custom table, an entry is created in the DialogColumn table. When you delete a column, the entry is removed from the DialogColumn table. Changes to the schema of default database views are not permitted.</p>
Condition for view definition	Restricting condition for creating the database view as a WHERE clause for database queries. The condition is attached to the view definition generated from the extension.
Columns	Database view columns.

Related topics

- [Table definition properties](#) on page 68
- [Column definition properties](#) on page 91
- [Creating database views with read-only type](#) on page 445
- [Using Common Table Expressions in read-only database views](#) on page 447
- [Database views of the View type](#) on page 55
- [Database views of the proxy type](#) on page 57
- [Database views of the Union type](#) on page 59

Table scripts

Table scripts help you to define actions that are run before or after saving, loading, or discarding an object. In this way, substantial changes or value checks that cannot be easily done with formatting rules or templates, can be made to an object by running a table script before it is saved. After the object is saved, changes to other objects can be made or task and processes can be generated with table scripts, for example. The side effect and tasks defined in the Customizer are applied following the table scripts.

You can customize predefined default table scripts and create your own additional table scripts. Table scripts are stored in VB.Net syntax which allows use of all VB.Net script functions.

To add table scripts

1. In the Designer, select the **One Identity Manager schema** category.
2. Select the table and start the Schema Editor with the **Show table definition** task.
3. In the **Table properties** view, select the **Table scripts** tab and create the required scripts.

Table 22: Table scripts

script	Description
Script (OnDiscarded)	The script is run after the object is discarded.
Script (OnDiscarding)	The script is run before the object is discarded.
Script (OnLoaded)	The script is run after the object is loaded.
Script (OnSaved)	The script is run after the object is saved.
Script (OnSaving)	The script is run before the object is saved.

4. Select the **Database > Save to database** and click **Save**.

IMPORTANT: Compile the database to bring the table scripts into effect.

Related topics

- [Visual Basic .NET scripts usage](#) on page 341
- [Templates for generating values](#) on page 75
- [Creating formatting scripts](#) on page 83

Working with a globally unique identifier module

To transport, for example, predefined reports, processes, workflows, or mail definitions with a complete system configuration transport, the objects require a primary key with a module GUID. These objects are identified as part of the system configuration through the module GUID.

Syntax

The table primary key has the CCC-[0-9,a-f](32) format.

NOTE: Entries with a module GUID are transferred automatically to the transport package when a transport of the entire system configuration is created.

You can use the following table definition settings for generating a module GUID:

- If the **Module GUID permitted** and **Module GUID required** options are enabled, the objects have to get a module GUID. The objects in this type of labeled tables are given the **CCC** module prefix.
- If only the **Module GUID permitted** option is enabled, the objects can get a module GUID in the required format. By default, the objects obtain a default GUID in the [0-9,a-f](8-4-4-4-12) format. Create the objects with the **CCC** prefix if they should obtain a module GUID. You can do this using the Object Browser.

Example:

- The **Module GUID required** and **Module GUID permitted** options are enabled on the DialogGroup table. When creating a new permissions group, the primary key is automatically generated in the format of a module GUID.
- For the AERole table only the **Module GUID permitted** option is set. To ensure that your own application roles are added to the transport package, create the application roles in the Object Browser with a module GUID.

NOTE:

- In the default case, the table's primary key is created with a default GUID. To subsequently change a default GUID to a module GUID, you use the Object Browser.
- GUIDs in tables that are labeled with IsNoReload = 1 in the QBM_VHeavyLoadTables view cannot be changed.

IMPORTANT: Do not run the following steps for production databases. Only perform these steps within the maintenance window. Otherwise, this could lead to inconsistent data.

To change a default GUID to a module GUID

1. In Object Browser select the object for which you want to change the default GUID.
2. Display the **Properties** context menu.
3. On the **Methods** tab select the **SwitchToModuleGuid()** method and click **Run**.

To change a module GUID to a default GUID

1. In Object Browser select the object for which you want to change the module GUID.
2. Display the **Properties** context menu.
3. On the **Methods** tab select the **SwitchToNormalGuid()** method and click **Run**.

Related topics

- [Table definition properties](#) on page 68


Defining unique columns for tables

If there is a column or column combination for a table that needs to be unique, you define multicolumn uniqueness in the Designer. The columns are collected into a unique groups.

Example:


- For the Hardware table, you must ensure that the name of the hardware is unique. For the Hardware table, a **Hardware** unique group with the Ident_Hardewarelist column is created.
- For the ADSDomain table, the combination of the domain identifier and its defined name must provide unique values. For the ADSDomain table, an **ADSDomain** unique group with the Ident_Domain and DistinguishedName columns are created.

To group together columns in a unique group

1. In the Designer, select the **One Identity Manager schema** category.
2. Select the table and start the Schema Editor with **Show table definition**.
3. In the **Table properties** view, select the **Multicolumn uniqueness** tab and click .

4. Enter the following information.

Table 23: Properties of tables for unique groups

Property	Description
Unique group	Name of the unique group of columns.
Columns	Enable the columns that must be unique when combined.
Error Message	<p>Text for an error message if the default error message is not to be used. Customized error messages are shown in the form:</p> <p><table display name (Plural)>: <error message of the unique group></p> <p>In the error you can include the following place holders:</p> <ul style="list-style-type: none">• {0}: Display value of the other object that already has the same value or has a value combination.• {1}: The conflicting value. <p>The place holder syntax corresponds to a format place holder in Visual Basic .NET.</p> <p>Translate the given text using the  button.</p>
Ignore empty values	<p>Specifies whether empty values are permitted in a unique group. This option can only be set if all columns in the group can be empty.</p> <ul style="list-style-type: none">• If the option is set, empty values are permitted in the relevant columns. If at least one of the relevant columns is not empty, uniqueness is tested. If all the group's columns are empty, uniqueness is not tested. This allows several data records to be inserted that all have empty group columns.• If this option is not set, empty values are permitted but only once for each column. Several data records whereby all the group's columns are empty, cannot be inserted.

5. Select the **Database > Save to database** and click **Save**.

TIP: To prevent empty values in a column, define a minimum length for the column in the column definition.

Related topics

- [Table definition properties](#) on page 68
- [Column definition properties](#) on page 91

Specifying deferred deletion for objects

You can use deferred deletion to specify how long the objects remain in the database after deletion is triggered before they are finally removed.

- If deferred deletion **> 0** days is configured, a deferred operation is created for deletion. The objects are initially disabled. During the retention period, you have the option to restore the objects. If a deleted object is restored, the object properties are reset to their state before deletion. The objects are finally deleted when the deferred deletion time period has expired.
- Object with deferred deletion on **0** days are deleted immediately.

Example:

Deferred deletion is applied especially to target systems. By default, user accounts are finally deleted from the database after 30 days. First, the user accounts are disabled or blocked. You can reenable the user accounts up until deferred deletion runs. After deferred deletion is run, the user accounts are deleted from the database and cannot be restored anymore.

You define a deletion delay for each table. Use the following table properties:

- **Deferred deletion [days]** (default): Number of days to defer the delete operation. If the value is **0**, it is deleted immediately. Use this if all objects of a table are to be handled with a defined deletion delay.
- **Script (deferred deletion)**: Script in VB.Net syntax to determine an object-specific deferred deletion. The script overwrites the value from the **Deferred deletion [days]** property. For example, use the script to define different time periods for individual objects in a table, depending on certain properties.

Example:

Deferred deletion of privileged user accounts is 10 days. The following **Script (deferred deletion)** is entered in the UNSAccountB table.

```
If Not $IsPrivilegedAccount:Bool$ Then
    Value = 10
End If
```

Related topics

- [Editing table definitions](#) on page 68
- [Table definition properties](#) on page 68
- [Working with objects in One Identity Manager](#) on page 22

Editing table definitions

To edit table properties

1. In the Designer, select the **One Identity Manager schema** category.
2. Select the table and start the Schema Editor with the **Show table definition** task.
3. In the **Table properties** view, edit the table properties.
4. Select the **Database > Save to database** and click **Save**.




Related topics

- [Table definition properties](#) on page 68

Table definition properties

Table 24: Table definition properties

Property	Description
Table	Name of the table in the data model.
Usage type	<p>The table's usage type provides the basis for reports and the selection of tasks for daily maintenance.</p> <p>Permitted values are:</p> <ul style="list-style-type: none">• Work tables: The table is a work table and contains transaction data.• Historical transaction data: The table contains transaction data to create histories.• Configuration: The table contains data for the system configuration.• Materialized data: The table contains materialized data. This is recreated through DBQueue Processor calculations.• Read-only data: The table contains read-only data.• User data: The table contains user data.

Property	Description						
Display name (singular)	Display name for a single record in the table. Translate the given text using the  button.						
Display name (plural)	Displays table name The display name is used, for example, to identify the table in a database search or for error output. Translate the given text using the  button.						
Display template	<p>The display template is used to specify the form in which objects will be represented, for example in the administration tool result list or in reports. Translate the given text using the  button. For more information, see Display template for displaying a list on page 132.</p> <p>NOTE: You do not need to enter a display template for many-to-many tables. For these tables, the viDB.DLL forms the display template from the foreign keys.</p>						
Display template (long)	Additional display template for individual tables containing the object's full name.						
Hierarchy path	<p>Enter the foreign key columns here that should be used as a basis for displaying tables hierarchically, for example, on assignment forms. For more information, see Hierarchical display of data on assignment forms on page 149.</p> <p>Example:</p> <p>An Active Directory user account (ADSAccount table) is typically displayed on an assignment form below its Active Directory container (UID_ADSContainer column). The Active Directory container (ADSContainer table) is, on the other hand, displayed underneath its Active Directory domain (UID_ADSDomain column). The path for the hierarchy structure is entered as follows:</p> <table> <tr> <th>Table</th><th>Hierarchy path</th></tr> <tr> <td>ADSAccount</td><td>UID_ADSContainer,UID_ADSDomain</td></tr> <tr> <td>ADSContainer</td><td>UID_ADSDomain</td></tr> </table> <p>An alternative list for objects that do not have values in all foreign key columns can be given after a pipe ().</p> <p>Example:</p> <p>(UID_ADSContainer,UID_ADSDomain UID_ADSDomain)</p>	Table	Hierarchy path	ADSAccount	UID_ADSContainer,UID_ADSDomain	ADSContainer	UID_ADSDomain
Table	Hierarchy path						
ADSAccount	UID_ADSContainer,UID_ADSDomain						
ADSContainer	UID_ADSDomain						
Remarks	Text field for additional explanation.						
Cache information	Loading behavior for tables in the Designer. This data is only required for system tables. Cache information for a table is composed of the sort order and loading behavior.						

Property	Description
	<p>Permitted values are:</p> <ul style="list-style-type: none"> • Base table: The table is loaded before the user interface. • User table: The table is only filled for the current user. • Data table: The table is loaded in the background after the user interface is loaded. • Proxy: The table is displayed as a view of the original table in the Designer. The data is loaded but cannot be modified. • Load BLOBS: Columns with larger data sets (BLOB columns) are loaded. • No caching: The table is not loaded in the Designer.
Disabled by preprocessor	If a table is disabled by a preprocessor condition, the option is set by the Database Compiler. For more information, see Conditional compilation using preprocessor conditions on page 335.
Preprocessor condition	You can add preprocessor conditions to tables. The table is therefore only available together with its columns when the preprocessor condition is fulfilled. For more information, see Conditional compilation using preprocessor conditions on page 335.
Deferred deletion [days]	Number of days to defer the delete operation. If the value is 0 , it is deleted immediately. For more information, see Specifying deferred deletion for objects on page 67.
Script (deferred deletion)	Script in VB.Net syntax to determine an object-specific deferred deletion. For more information, see Specifying deferred deletion for objects on page 67.
Icon	Icon representing the table in the administration tool interface.
Background color	Color used to display the control for this table in the schema overview.
Proxy view	<p>Reference to database view, type Proxy, which uses the table content.</p> <p>Example:</p> <p>The database view UNSRoot is used to map the ADSDomain and LDAPDomain tables in the Unified Namespace.</p> <p>For more information, see Database views of the proxy type on page 57.</p>
Extensions to proxy view	List of columns as SQL text. These are used in the database view's SELECT statement that is selected under Proxy view . For example, use the extensions to the proxy view if columns are doubly mapped or if additional proxy view need to be filled.

Property	Description						
	<p>Example:</p> <p>The view UNSRoot expects the target system type as input in the UID_DPRNameSpace column. This column is not in the ADSDomain and LDAPDomain tables.</p> <p>The proxy view extension is defined as follows:</p> <table> <tr> <td>Table</td><td>Extension to proxy view</td></tr> <tr> <td>ADSDomain</td><td>'ADS-DPRNameSpace-ADS' as UID_DPRNameSpace</td></tr> <tr> <td>LDAPDomain</td><td>'LDP-DPRNameSpace-LDAP' as UID_DPRNameSpace</td></tr> </table> <p>For more information, see Database views of the proxy type on page 57.</p>	Table	Extension to proxy view	ADSDomain	'ADS-DPRNameSpace-ADS' as UID_DPRNameSpace	LDAPDomain	'LDP-DPRNameSpace-LDAP' as UID_DPRNameSpace
Table	Extension to proxy view						
ADSDomain	'ADS-DPRNameSpace-ADS' as UID_DPRNameSpace						
LDAPDomain	'LDP-DPRNameSpace-LDAP' as UID_DPRNameSpace						
Logical disk store	<p>The table's logical disk store. Associated tables are grouped together in logical disk stores. In the default installation, logical disk stores are predefined for the table in each module of One Identity Manager and the system tables. You cannot change the assignments. You can create your own logical disk storage for grouping custom tables. Supporting file groups on page 104</p>						
Scope hierarchy	<p>Comma delimited list of all foreign key columns required for displaying objects in the scope hierarchy in the Synchronization Editor. List of all columns that lead to tables made available by the parent object.</p>						
Person object path for table lookup support	<p>Path to the Person object for finding the person object within the table lookup search for user accounts and email addresses. The resulting data is mapped in QBMSplittedLookup.SplittedElement. If the value no is entered, no employee can be determined for groups or BaseTree derivatives, for example.</p> <p>Example:</p> <p>In the case of Exchange Online mailboxes (O3EMailbox table), the employee is determined through the Azure Active Directory user accounts.</p> <p>Enter the path to the person object as follows: FK(UID_AADUser).UID_Person</p>						
Export for SPML schema	<p>This option determines whether the table should be exported for the SPML schema. For more information, see Preparing the One Identity Manager schema for export to the SPML schema on page 497.</p>						
Many-to-many table	<p>Label for assignment tables (many-to-many tables). Assignment tables are tables used to create relations between two other tables. For more information, see Table types and default columns in the One Identity Manager data model on page 49.</p>						

Property	Description
Many-to-all table	Marks assignment tables, which have a dynamic foreign key as partner. For more information, see Table types and default columns in the One Identity Manager data model on page 49.
No DB Transport	Tables labeled with this option cannot be excluded from a custom configuration package. These tables are excluded from data transport.
Assign by event	Specifies how assignments and deletions are handled in tables. This option only applies to assignment tables (many-to-many tables) in the application data model. <ul style="list-style-type: none"> • If the option is not set, assignments, and deletions are dealt with directly by the DBQueue Processor. • If the option is set, tasks for the HandleObjectComponent process component are set up in the Job queue. These tasks then carry out the relevant operations. This makes it possible to link specific processes directly to the Assign and Remove events. You must implement this behavior on a custom basis.
Retain in memory	Specifies whether the table contents for the data connection can be buffered. The threshold is defined in the Common ResidentTableLimit configuration parameter.
Module GUID permitted	Specifies whether a primary key with a Globally Unique Identifier module (GUID module) is permitted for objects. For more information, see Working with a globally unique identifier module on page 64.
Module GUID required	Specifies whether a primary key with a Globally Unique Identifier module (GUID module) is required for objects. For more information, see Working with a globally unique identifier module on page 64.
Type	Table type. For more information, see Table types in One Identity Manager on page 54.
Base table	Only for database views: Reference to base tables that a view is based on.
Condition for view definition	Only for database views: Limiting condition for creating the database view as WHERE clause for database queries.
Insert values	Specify default settings for a column that is assigned when a new data set is added. The values are entered in VB.Net syntax.
Selection script	Only for database views: Selection script as VB.Net expression to determine at runtime, whether the object passed belongs to this database view.
Script	Script in VB.Net syntax that is run after the object is loaded. For more

Property	Description
(OnLoaded)	information, see Table scripts on page 63.
Script (OnSaving)	Script in VB.Net syntax that is run before the object is saved. For more information, see Table scripts on page 63.
Script (OnSaved)	Script in VB.Net syntax that is run after the object is saved. For more information, see Table scripts on page 63.
Script (OnDis-carding)	Script in VB.Net syntax that is run before the object is discarded. For more information, see Table scripts on page 63.
Script (OnDis-carded)	Script in VB.Net syntax that is run after the object is discarded. For more information, see Table scripts on page 63.
Number of rows	Number of rows in the table The number of rows in the table is determined once a day by maintenance tasks. The data material can help to plan capacities and maintenance work on the database.
Basic record lengths	Maximum length of the data record with (clustered) main indexes. Only the reference is saved for LOBs. The LOB content itself is stored in the HEAP. The basic record length is determined once a day by maintenance tasks. The data material can help to plan capacities and maintenance work on the database.
Table size	The size of the table in MB. The size of the table in the database is determined once a day by maintenance tasks. The data material can help to plan capacities and maintenance work on the database.
Condition for transport	Condition for selecting transportable objects. An empty condition means that all object are transferred.
Layout information	(Only for internal use) Information about the layout in the Designer.
Primary key 1	(Only for internal use) Name of the table's first primary key column. The sort order of primary key 1 and primary key 2 corresponds to the physical order in the schema.
Primary key 2	(Only for internal use) Name of the table's second primary key column. The sort order of primary key 1 and primary key 2 corresponds to the physical order in the schema.
Synchronization mode	Permitted directions of synchronization and handling methods for this table if synchronization is set up automatically between two One Identity Manager databases.
Columns for alternative rules	Comma delimited list of columns to be used for creating alternative object matching rules in an automatically created synchronization project. If the One Identity Manager connector cannot identify a system object through the primary object matching rule, it applies the alternative rules to determine a matching system object. For

Property	Description
	more information about this, see the <i>One Identity Manager User Guide for the One Identity Manager Connector</i> .
CLR type for project generator	.NET class used to consider special cases when generating a synchronization project between two One Identity Manager databases.

Related topics

- [Editing table definitions](#) on page 68

Displaying the table definition Customizer

Customizers run processing logic which would normally be implemented in the object code, such as mutual exclusion of properties. Customizers contain special methods and has side effects on the table columns. Several customizers can be defined for one table.

The One Identity Manager default installation contains various customizers which provide specific behaviors.

To display the customizers for a table definition

1. In the Designer, select the **One Identity Manager schema** category.
2. Select the table and start the Schema Editor with the **Show table definition** task.
3. In the **Table properties** view, select the **Customizer** tab.

Related topics

- [Column dependencies for setting values](#) on page 84

Column definitions

Column definitions for application and system data model tables are kept in the DialogColumn table. The predefined column properties of the One Identity Manager schema are maintained by the schema installation and cannot be edited apart from a few exceptions.

In the Designer, you can edit the One Identity Manager schema's column definitions using the Schema Editor.

Detailed information about this topic

- [Notes on editing table definitions and column definitions](#) on page 53
- [Templates for generating values](#) on page 75

- [Defining unique columns for tables](#) on page 65
- [Dynamic foreign key](#)
- [Limiting column lengths](#) on page 80
- [Defining decimal places for displaying values](#) on page 80
- [Using predefined formatting types](#) on page 81
- [Creating formatting scripts](#) on page 83
- [Column dependencies for setting values](#) on page 84
- [Permitted column values](#) on page 84
- [Specifying requirements for MVP columns](#) on page 86
- [Defining bitmasks](#) on page 86
- [Configuring columns for full-text search](#) on page 87
- [Scripts for conditionally displaying and editing columns](#) on page 89
- [Flagging columns for translation](#) on page 212
- [Editing column definitions](#) on page 90
- [Column definition properties](#) on page 91

Templates for generating values

In One Identity Manager, value templates are implemented for generating user data or for transforming values. You can use these templates to fill object properties with default values or to form property values from other properties. Value templates can take effect within an object as well as between objects. Value templates without dependencies take effect when the value is queried in the column and the column does not have a value assigned. Value templates that refer to other columns are affected when these columns change.

Value templates take effect without regard to the current permissions. No explicit permissions need to be assigned to the dependent columns. When value templates are applied, the accessed columns of an object are also filled if they are not visible on the current form in the Manager.

Column dependencies due to value templates are mapped in the `DialogNotification` table. The connected properties are shown in the table as sender-subscriber pairs. The column that caused the change is the sender and the column that is changed because of it, is the subscriber. The object links are consolidated by the column relations. The entries are created when the value templates are compiled and updated.

NOTE: In the Designer, you can get an overview of existing columns with value templates in the **One Identity Manager Schema > Templates** category. Column dependencies due to value templates are mapped in the schema overview in the Schema Editor.

Detailed information about this topic

- [Editing value templates on page 76](#)
- [Preventing a change to a column on page 77](#)
- [Restricting performance of value templates on page 78](#)
- [Example of local value templates within an object on page 79](#)
- [Example of cross-object value templates on page 79](#)
- [Displaying the column dependencies based on templates on page 48](#)

Related topics

- [Defining unique columns for tables on page 65](#)
- [Permitted column values on page 84](#)
- [Column dependencies for setting values on page 84](#)

Editing value templates

You can customize predefined default value templates and create your own additional value templates.

IMPORTANT: You must take performance factors into consideration when defining value templates. In certain circumstances, changes to a property could cause large numbers of dependent objects to be changed, saved, and processes to be generated through a value template in overwrite mode.

To limit the number of objects changed by a value template you can define thresholds for running value templates. For more information, see [Restricting performance of value templates on page 78](#).

To create a value template

1. In the Designer, select **One Identity Manager Schema**.
2. Select the table and start the Schema Editor with the **Show table definition** task.
3. Select the column and then the **Column properties** view.

4. Select the **Value calculation** tab and edit the following properties.

Table 25: Properties for calculating values of a column

Property	Description
Overwrites	<p>Specifies whether the template can overwrite or not. If this option is set, the value template is always applied. If the option is not set, the value template is only applied when the column is empty.</p> <p>NOTE: The One Identity Manager schema only knows the values 0 and 1 for columns of Bool data type. The value 0 is the same as empty. That means, if the Overwrites option disabled, the template is run if the column value changes from 0 to 1.</p>
Template	<p>Template script. Write the script in VB.Net syntax which allows all VB.Net script functions to be used.</p> <p>TIP: To display the columns that trigger a template, click Triggers for this template.</p>
No automatic truncation by template	<p>Specifies whether the value is automatically truncated to the maximum column length if the maximum length is exceeded when applying a template. If this option is enabled, the value is not automatically truncated to the maximum column length.</p>

5. Select the **Database > Save to database** and click **Save**.

IMPORTANT: Compile the database to bring the value template into effect.

TIP: Test compiling in the Designer using the **Schema > Test compile** menu item.

Related topics

- [Preventing a change to a column](#) on page 77
- [Example of local value templates within an object](#) on page 79
- [Example of cross-object value templates](#) on page 79
- [Preprocessor conditions in VB.Net expressions](#) on page 338
- [Visual Basic .NET scripts usage](#) on page 341
- [Column definition properties](#) on page 91

Preventing a change to a column

You can use value templates to prevent users from changing columns that are filled by a value template. To do this, add the name of this column in the value template in \$-notation. The value template now references itself. Any change to the column is immediately overwritten by the value template. Value templates that overwrite themselves only take effect if they have been labeled as "Overwrites".

Example:

The user should not be able to change an employee's central user account. This should be prevented by the value template.

- Define a custom value template for the `Person.CentralAccount` column.
- For the value templates, enable the **Overwrites** option.
- Extend the default value template with the following entry: '\$CentralAccount\$.

```
'$CentralAccount$
```

```
If Not CBool(Session.Variables.Get("FULLSYNC")) Then
```

```
    Value=VI_AE_BuildCentralAccount(GetValue("UID_
    Person").String,$Lastname$, $Firstname$)
```

```
End If
```

Restricting performance of value templates

To limit the number of objects changed by a value template you can define thresholds.

To define thresholds for a value template

1. In the Designer, select the **One Identity Manager Schema** category.
2. Select the table and start the Schema Editor with the **Show table definition** task.
3. Select the column and then the **Column properties** view.
4. Select the **Value calculation** tab and edit the following properties.
 - **Threshold (asynchron)**: Enter the maximum number of objects that can be changed directly by the value template. Once this limit has been reached, processing takes place synchronously with the One Identity Manager Service.
 - **Threshold (stop)**: Enter the number of objects after which processing is stopped. Once this limit has been reached, processing is stopped with an error message.

NOTE: If a stop threshold value is specified, it must be larger than the threshold for asynchronous processing.

5. Select the **Database > Save to database** and click **Save**.

Related topics

- [Editing value templates](#) on page 76
- [Column definition properties](#) on page 91

Example of local value templates within an object

The an employee's full name (Person.Internalname) will be derived from its surname (Person.Lastname) and first name (Person.Firstname). The value template for the Person.Internalname column looks like:

```
Value = $Lastname$ & ", " & $Firstname$
```

If the value template is labeled as "Overwrites" then each time Lastname changes a test is done to check for dependent columns that reference this value in a template. If this is the case, the value template is processed and the value is entered into the Internalname column. If the value template cannot overwrite, it only applies if there is no value in the Internalname column.

The Person.Lastname and Person.Firstname columns are the sender and the Person.Internalname column is the subscriber. The mapping for adding a database object in the DialogNotification table is:

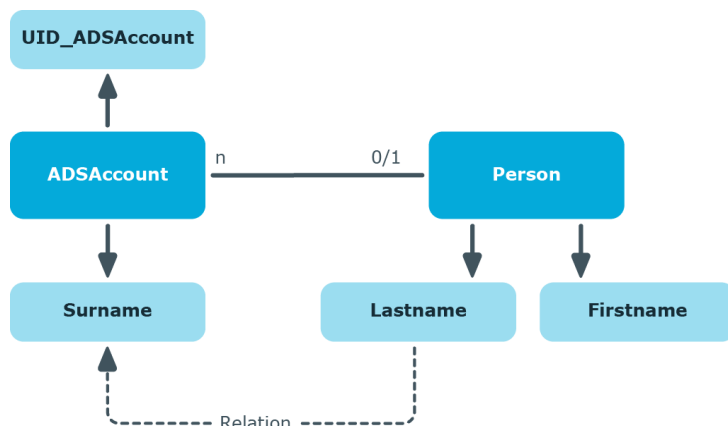
```
person.lastname --> person.internalname
```

```
person.firstname --> person.internalname
```

Example of cross-object value templates

If a value template references a value from another object, it can be accessed using the foreign key (FK) relation.

Figure 9: Effect of cross-object value templates



If, for example, the surname of an Active Directory user account (ADSAccount.Surname) is derived from the surname of an employee (Person.Lastname), enter the template for the ADSAccount.Surname column as follows:

```
Value = $FK(UID_Person),Person.Lastname$
```

If the employee's surname changes, the last name of the Active Directory user changes, too. The Person.Lastname column is therefore the sender and the ADSAccount.Surname column is the receiver. The relation is mapped in the DialogNotification table as follows:

Person.Lastname --> ADSAccount.Surname

Limiting column lengths

You can use the column definition to control the length of the values to be entered. For example, the login name of an Active Directory user account is limited to a maximum of 20 characters. You can also use the column definition to define which columns are required.

To define the length of a column

1. In the Designer, select the **One Identity Manager Schema** category.
2. Select the table and start the Schema Editor with the **Show table definition** task.
3. Select the column and then the **Column properties** view.
4. Select the **Value calculation** tab and edit the following properties.
 - **Max. length:** Enter the maximum length of the column. If the value is equal to **0**, the length is taken from the database schema.
 - **Min. length:** Enter the minimum length of the column. Columns with a minimum length of **1** or greater are flagged as required fields in the front-ends.
5. Select the **Database > Save to database** and click **Save**.

Related topics

- [Column definition properties](#) on page 91

Defining decimal places for displaying values

In the user interface, you can define the number of decimal places for displaying values of columns with the .Net data types Double, Decimal, Int, Long or Short.

In columns with the .Net data types Int, Long or Short, the decimal point is shifted in the value display when the decimal places are specified. In columns with the .Net data types Double or Decimal, the value is displayed with the corresponding number of places after the decimal point. Take this behavior into account when calculating with different data types.

Example:

- Prices with the .Net data type `Int` are given with cent after the decimal point. In the front-end a price of \$3.50 is displayed as **3.50** and saved in the database as **350**.
- Quantities with the .Net data type `double` are, for example, specified with three places after the decimal point. In the front-end, a quantity of 100 pieces is displayed with the value **100,000**, while the value **100** is saved in the database.

To define the number of decimal places

1. In the Designer, select the **One Identity Manager Schema** category.
2. Select the table and start the Schema Editor with the **Show table definition** task.
3. Select the column and then the **Column properties** view.
4. Select the **Column** tab and enter the number of decimal places to be used in the **Number of decimal places** input field.
5. Select the **Database > Save to database** and click **Save**.

Related topics

- [Column definition properties](#) on page 91

Using predefined formatting types

You can specify column formats based on predefined formatting types. By combining formatting types with each other, you can obtain the formatting you required.

NOTE: If there is a column or column combination for a table that needs to be unique, define multi-column uniqueness in the Designer. For more information, see [Defining unique columns for tables](#) on page 65.

To specify formatting types

1. In the Designer, select the **One Identity Manager schema** category.
2. Select the table and start the Schema Editor with the **Show table definition** task.
3. Select the column and then the **Column properties** view.
4. Select the **Value calculation** tab and define the formatting types in the **Column**

format input field.

Table 26: Permitted formatting types

Value	Formatting type	Permitted values
0	None	No special formatting = default
1	IP address	IP address [0-9] ³ . [0-9] ³ . [0-9] ³ . [0-9] ³
2	MAC ID	MACID [0-9,A-F]12
4	Drive letter	Drive letter [A-Z]1:
8	Number	[0-9] ⁺
16	Uppercase	Uppercase
32	Uppercase server dependent	(only maintained for compatibility reasons)
64	NT name	All characters are permitted except for „!@/\.:\"[]; - = + * ? < > \"
128	Phone	Phone [0123456789#/-+*]n
256	Exchange name	All characters are permitted except for „ÄÖÜäöüß\"!§\$%&\\ /<>#*{}[] ²³ ~^,\"
512	ASCII characters and numbers	All characters of the ASCII character set
2048	URI	Uniform Resource Identifier
4096	Email address	Valid email address
8192	Prevent XSS characters	<p>Certain characters that can be used for cross-site scripting (XSS) are invalid. The < and > characters are tested.</p> <p>Testing only takes place if the QBM XssCheck configuration parameter is set. If the QBM XssCheck Sync configuration parameter is set, testing is also done during synchronization.</p>

5. Select the **Database > Save to database** and click **Save**.

| IMPORTANT: Compile the database to implement the formatting type.

Related topics

- [Creating formatting scripts](#) on page 83
- [Column definition properties](#) on page 91

Creating formatting scripts

You can use a formatting script to verify column values. Formatting scripts, as opposed to value templates, are only run when a value is assigned to the column.

To create a formatting script

1. In the Designer, select the **One Identity Manager Schema** category.
2. Select the table and start the Schema Editor with the **Show table definition** task.
3. Select the column and then the **Column properties** view.
4. Select the **Value calculation** tab and enter the formatting script for the column in the **Formatting script** input field.

Write the script in VB.Net syntax which allows all VB.Net script functions to be used.

5. Select the **Database > Save to database** and click **Save**.

| IMPORTANT: Compile the database to bring the formatting script into effect.

| TIP: Test compile using the **Schema > Test compile** menu item.

Example:

The value in the column Mail in the ADSAccount table should correspond to SMTP format. If this is not the case, an error message is sent. The formatting script for the ADSAccount.Mail column can be formulated as follows:

```
Dim str as String = Convert.ToString(Value)
If str.Length > 0 Then
    If Not VID_IsSMTPAddress(str) Then
        Throw New Exception(""" & str & """" is not a valid SMTP
        address.")
    End If
End If
```

Related topics

- [Using predefined formatting types](#) on page 81
- [Visual Basic .NET scripts usage](#) on page 341
- [Column definition properties](#) on page 91

Column dependencies for setting values

There may be dependencies between individual values, for example, by using value templates or customizers that require values to be set in a specific order. In the case of One Identity Manager tools the correct order is enforced through blocking or releasing input fields. In the case of data import and when using SPML and web service interfaces, the correct order for setting values also has to be safeguarded.

The following data sources assume the following sequence for specifying the order for setting values:

1. Customizer

The dependencies between columns and an object are stored in customizers.

2. Custom defined dependencies

To define custom dependencies between columns

- a. In the Schema Editor, select a table column.
- b. In the **Column properties** view on the **Dependencies** tab, define the predecessor of this column.

3. Column dependencies due to value templates

In this case, values used by a template (for example, `Person.Firstname`, `Person.Lastname`) are set before values that are created by a template (for example, `Person.CentralAccount`).

If circular dependencies occur whilst determining the order for setting the values, they are stopped at the point of lowest priority.

Related topics


- [Displaying the table definition Customizer](#) on page 74
- [Templates for generating values](#) on page 75




Permitted column values

To only allow a specific value in a column, you can define a list with the permitted values. This list of permitted value is resolved once the column's display value has been formatted. For some columns of the One Identity Manager schema, already permitted values are supplied when the schema is installed.

NOTE: You can only enter or extend a list of permitted values for a column if the option **Customizing permitted values list is not allowed** is not set.

To create a list of permitted values

1. In the Designer, select the **One Identity Manager Schema** category.
2. Select the table and start the Schema Editor with the **Show table definition** task.
3. Select the column and then the **Column properties** view.
4. Select the **Value settings** tab and enable the **Defined list of values** option.
5. Click  and enter the following properties.
 - **Value:** Value for the value list (technical name).
 - **Display value:** Name used to display the value. Translate the given text using the ... button.
 - **Sort:** Sort order of displaying the values in the list.
 - **Disabled:** Specifies whether the value is disabled.
6. Select the **Database > Save to database** and click **Save**.

NOTE: To edit a value, select the value and click . To delete a value, select the value and click . To delete all values, click .

IMPORTANT: Compile the database to bring the list of permitted values into effect.

Example:

In the **Spare field no. 01** input field for an employee, the values **internal** and **external** should be permitted. The list of permitted values is defined as followed:

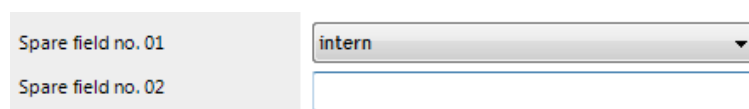
1=internal 2=external

For an employee with the value **1**, the display value **internal** is shown on the forms in the Manager.

Display columns with permitted values in the Manager

A special control element is used in the Manager to display columns for which a list of permitted values has been defined. The control element is displayed as a simple input field if no list is defined. If a list is defined the control element is shown as a menu.

Figure 10: Input field for list of defined values (with and without defined entries)



The control element is only available for columns on default predefined forms as well as custom columns (usually CustomProperty01-CustomProperty10).

Related topics

- [Column definition properties](#) on page 91

Specifying requirements for MVP columns

Values in multi-values property (MVP) columns are delimited by **char(7)** or **chr(7)**. Specify other requirements for each value of the MVP column.

To specify requirements for a MVP column

1. In the Designer, select **One Identity Manager Schema**.
2. Select the table and start the Schema Editor with the **Show table definition** task.
3. Select the column and then the **Column properties** view.
4. Select the **Value settings** tab and set the **Multi-value column** option.
5. In the **Multi-value specification** menu, configure the following settings.
 - **Unique**: Set this option if the value must be unique.
 - **Case sensitive**: Set this option if the case sensitivity should be taken into account when the value is tested.
 - **Accent insensitive**: Set this option if accent characters should not be taken into account when the value is tested.
6. Select the **Database > Save to database** and click **Save**.

Related topics

- [Column definition properties](#) on page 91

Defining bitmasks

You can only define bitmasks for columns with the **int** type.


NOTE: You can only enter or extend a bitmask for a column if the option **Customizing bitmasks is not allowed** is not set.

To create a bitmask




1. In the Designer, select **One Identity Manager Schema**.
2. Select the table and start the Schema Editor with the **Show table definition** task.
3. Select the column and then the **Column properties** view.
4. Select the **Value settings** tab and set the **Defined bitmask** option.
5. Under **Sort criteria of bitmask**, select the sort criteria for displaying the value.

You have the following options:

- **Bit position:** Sort by position.
- **Display:** Sort by display value.

6. Click  and enter the following properties.

- **Bit position:** Each bit position. The first bit in the definition start with the index **0**.
- **Display value:** Name used to display the value. Translate the given text using the ... button.
- **Disabled:** Specifies whether the value is disabled.

NOTE: To edit a value, select the value and click . To delete a value, select the value and click . To delete all values, click .

7. Select the **Database > Save to database** and click **Save**.

Related topics

- [Column definition properties](#) on page 91

Configuring columns for full-text search

Full-text searching uses an external search index, which returns an object key as result. The object key is used to run a search query in the database. This database search query takes the permissions of the logged in user into account during the search. A maximum of 1000 objects can be returned by through the search index.

The One Identity Manager full text search can be used in the Web Portal and in the Manager. For more information, see the *One Identity Manager Web Designer Web Portal User Guide* and the *One Identity Manager User Guide for One Identity Manager Tools User Interface*.

- Prerequisites for using full text search is an application server installed with the search service.
- If you run the Web Portal directly over an application server installed with the search service, you can use the full text search immediately.
- If you are working with the Web Portal and an application server without a search service installed or with a direct database connection, you will need to enter an application server with a search service in the Web Portal configuration file. Full text search is available in the Web Portal once this has been done.
- To use full text search in the Manager, you must run the Manager over an application server with an installed search service.

For more information about installing an application server and configuring the Web Portal for full-text search, see the *One Identity Manager Installation Guide*.

The following applies for the configuration of the full text search:

- The columns `XDateInserted`, `XDateUpdated`, and `XObjectKey` must be available if you want to index a table or database view for full text search.
- Columns for full text searching must be weighted. Increasing weighting results in a higher position in the search results. The default installation provides columns for the full-text search with a weighting of **1**.

Example:

The column `Person.CentralAccount` is weighted with the value **1**. The column `ADSAccount.SAMAccountName` is weighted with the value **0.5**. This results in the employee being listed before the user account in the full text search.

- Only columns with the .Net data types `string` or `text` can be included in the full-text search.
Exception: Columns that contain a list of permitted values, can always be included to the full text search.
- Columns of tables with the **Work tables** or **Historical transaction data** usage type cannot be included in the full-text search.
- Columns of assignment tables (M:N tables, M:all tables) cannot be included to the full-text search.

The search service indexes the following:

- Column content
- Foreign key column display value
- Display values for lists of permitted values
- Translation for every active language
- Object display value, if the table's primary key column is configured for full text search

The object's display value comes from the display pattern defined for the table. The display value's weighting comes from the table's primary key column weighting

Example:

The `Person.UID_Person` column is configured for full-text search. The display pattern of the `Person` table is defined as `%InternalName% (%CentralAccount%)`.

So the display value of **Clara Harris (CLARAH)** is indexed for the employee Clara Harris.

The searched index is updated when changes are made to a table with indexed columns, to referenced tables or translations.

Certain important columns are already indexed for full text search in the default installation. You configure more columns for full text searching if you require.

To configure a column for full text search

1. In the Designer, select the **One Identity Manager Schema** category.
2. Select the table and start the Schema Editor with the **Show table definition** task.
3. Select the column and then the **Column properties** view.
4. Select the **Column** tab and edit the **Index weighting** property.
 - If the value is less than or equal to **0**, no indexing takes place.
 - If the value is greater than **0**, the data value is indexed.
5. Select the **Database > Save to database** and click **Save**.

Related topics

- [Column definition properties](#) on page 91

Scripts for conditionally displaying and editing columns

In principle, a user's permissions for displaying and editing columns are controlled by permissions in permissions groups.

However, you can also use scripts to conditionally display or edit scripts. For example, in this way you can control whether or not a column, on a main data form in the Manager, is displayed or can be edited only if another column has a specific value.

Example:

A system role is disabled until the release data is reached. During this time period, the user must be prevented from changing the **disabled** option in the Manager.

To do this, a script is created specifying the edit permissions for the `ESet.IsInactive` column.

```
If $ReleaseDate:Date$>Connection.LocalNow Then
    Value = False
Else
    Value = True
End If
```

NOTE:

- The script does not change the user's permissions but simply the behavior if the object is loaded in one of the One Identity Manager tools. If you want to limit visibility and editability of a column, change the column permissions of the permissions groups. For more information, see the *One Identity Manager Authorization and Authentication Guide*.
- The scripts only affect interactively loaded objects.
- For example, in lists in the Manager or in the Web Portal, objects are not loaded interactively so the scripts do not work.
- In the Web Portal, a lot of objects are not loaded interactively due to performance reasons. If you want to use this behavior in the Web Portal, you must customize the components in the Web Designer. In this case, there can be adverse effects on performance when objects are loaded. For more information about editing Web Designer components, see the *One Identity Manager Web Designer Reference Guide*.

To specify a script for conditionally displaying and editing a column

1. In the Designer, select the **One Identity Manager Schema** category.
2. Select the table and start the Schema Editor with the **Show table definition** task.
3. Select the column and then the **Column properties** view.
4. Select the **Permissions scripts** and enter the following scripts in VB.Net syntax.
 - **Visibility script:** Script for conditionally displaying the column. If the return value is **false**, the column is not displayed in any One Identity Manager tools.
 - **Editability script:** Script for conditionally editing the column. If the return value is **false**, the column cannot be edited in any One Identity Manager tools.
5. Select the **Database > Save to database** and click **Save**.

Related topics

- [Column definition properties](#) on page 91
- [Working with objects in One Identity Manager](#) on page 22

Editing column definitions

To edit column properties



1. In the Designer, select the **One Identity Manager schema** category.
2. Select the table and start the Schema Editor with the **Show table definition** task.
3. Select the column in the Schema Editor and edit the column properties.
4. Select the **Database > Save to database** and click **Save**.

Related topics

- [Column definition properties](#) on page 91

Column definition properties

Table 27: Column properties

Property	Description
Table	Name of the table to which the column belongs.
Column	Name of the column in the data model.
Display name	Language-dependent column name for displaying in the administration tools user interface. Translate the given text using the  button.
Comment	Additional information about the column. The comment is displayed under the help function for a column in the individual administration tools. Translate the given text using the  button.
Disabled by preprocessor	If a column is disabled by a preprocessor condition, the option is set by the Database Compiler. For more information, see Conditional compilation using preprocessor conditions on page 335.
Preprocessor condition	<p>You can add preprocessor conditions to columns. The column is therefore only available when the preprocessor condition is fulfilled. For more information, see Conditional compilation using preprocessor conditions on page 335.</p> <p>NOTE: In the Designer, you can find an overview of existing preprocessor dependencies in the One Identity Manager Schema > Preprocessor dependencies category.</p>
Sort order	The sort order specifies the position for displaying the column on the generic form and the custom tabs of the default form. Columns with a value less than 1 are not displayed on the forms. For more information, see Displaying custom columns and tables on main data forms on page 146.
Group	Group is used to display the column on general main data forms. A new tab is created for each group on the generic form. For more information, see Displaying custom columns and tables on main data forms on page 146.
Base column	<p>If a database view has the View table type, the reference to the column in the base table is entered here. For more information, see Database views of the View type on page 55.</p> <p>Example:</p>

Property	Description						
	<p>The Department database view is part of the Basetree base table. The columns of the Basetree table are entered as base columns.</p> <table> <tr> <td>Column</td><td>Base column</td></tr> <tr> <td>Department.DepartmentName</td><td>BaseTree.Ident_Org</td></tr> <tr> <td>Department.Description</td><td>BaseTree.Description</td></tr> </table>	Column	Base column	Department.DepartmentName	BaseTree.Ident_Org	Department.Description	BaseTree.Description
Column	Base column						
Department.DepartmentName	BaseTree.Ident_Org						
Department.Description	BaseTree.Description						
Adjustment of permitted values list is not allowed	Specifies whether permitted values can be customized for this column. For more information, see Permitted column values on page 84.						
Defined list of values	Marks whether the value in this column must correspond to the values in the List of permitted values , or are empty. For more information, see Permitted column values on page 84.						
List of permitted values	If a column is enabled for editing the permitted values (that is, the Customizing permitted values list is not allowed option is not set and the Defined list of values option is set), you can add to or extend a value list. For more information, see Permitted column values on page 84.						
Customizing bitmask is not allowed	Specifies whether the bitmask can be customized for this column. For more information, see Defining bitmasks on page 86.						
Defined bitmask	Meaning of each bit position if the column contains a bitmask. The first bit in the definition start with the index 0 . For more information, see Defining bitmasks on page 86.						
Sort criteria of bitmask	<p>Sort criteria for displaying the values. You have the following options:</p> <ul style="list-style-type: none"> • Bit position: Sort by position. • Display: Sort by display value. <p>For more information, see Defining bitmasks on page 86.</p>						
Multilingual	<p>Specifies whether this column can be given in multiple languages.</p> <p>Permitted values are:</p> <ul style="list-style-type: none"> • Translation target: The column content is displayed in translation. • Translation source: The column supplies the translation. • #LD content: The column has contents in #LD notation. The contents are extracted for translation. • Without text memory fallback: The text store is not 						

Property	Description
	<p>used as fallback for the column.</p> <p>The combination of values determines the resulting translation. For more information, see Flagging columns for translation on page 212.</p>
Syntax	<p>Syntax type of data in this column. The syntax type is used to give One Identity Manager tools the appropriate syntax highlighting or input assistance.</p> <p>Permitted syntax types are:</p> <ul style="list-style-type: none"> • HTML: Input in HTML format. • Picture: Images. • SQL.Query: Full database queries. • SQL.Special: Special syntax for database queries. • SQL.WhereClause: WHERE clause for database queries. • Text.Dollar: Input in \$ notation. • UNC: UNC path. • URL: URL. • VB.Class: Full VB.Net class definitions. • VB.Instruction: VB.Net statements in the form <code>Value =</code>. • VB.Method: Single methods or functions in VB.Net. • XML: Input in XML format. • ConnectionString: Input as a connection string. • JSON: Input in JSON format. • Color: Input of color codes.
Number of decimal places	<p>Number of decimal places used to display values. For more information, see Defining decimal places for displaying values on page 80.</p>
Date add-on	<p>Additional information about displaying date and time in One Identity Manager tools.</p>
Index weighting	<p>Column weighting in indexing. Used for indexing the full-text search. Increasing weighting results in a higher position in the search results.</p> <p>If the value is less than or equal to 0, no indexing takes place. If the value is greater than 0, the data value is indexed. Columns to be indexed are assigned a weighting of 1 in the default installation.</p>

Property	Description																		
	For more information, see Configuring columns for full-text search on page 87.																		
Data type in database	<p>Shows the .Net data type for the column. This is used internally and cannot be edited. The Net data types are mapped internally to SQL data types. If no value is given, the data type is taken from the database schema.</p> <p>Permitted syntax types are:</p> <table> <tr> <th>.Net data type</th><th>Mapped SQL data type</th></tr> <tr> <td>Binary</td><td>varbinary, timestamp</td></tr> <tr> <td>Bool</td><td>bit</td></tr> <tr> <td>Date</td><td>datetime</td></tr> <tr> <td>Double</td><td>float</td></tr> <tr> <td>Int</td><td>int</td></tr> <tr> <td>Long</td><td>bigint</td></tr> <tr> <td>String</td><td>nvarchar/varchar/nchar</td></tr> <tr> <td>Text</td><td>nvarchar/varchar</td></tr> </table>	.Net data type	Mapped SQL data type	Binary	varbinary, timestamp	Bool	bit	Date	datetime	Double	float	Int	int	Long	bigint	String	nvarchar/varchar/nchar	Text	nvarchar/varchar
.Net data type	Mapped SQL data type																		
Binary	varbinary, timestamp																		
Bool	bit																		
Date	datetime																		
Double	float																		
Int	int																		
Long	bigint																		
String	nvarchar/varchar/nchar																		
Text	nvarchar/varchar																		
Maximum size in DB	Maximum column length in the database schema.																		
Minimum size in DB	Minimum column length in the database schema.																		
Primary key	The primary key is given when the database is created.																		
UID column	Specifies whether this is UID column. This option is only permissible for columns with the String .Net data type and a length of 38 characters.																		
Default value	Specifies whether a default value is defined by a template for this column.																		
BLOB value	This option is used to label text columns whose data contents is so large that they cannot be kept internally in one line in the SQL sever and are therefore saved as a reference. This allows speedier access to the data.																		
Log changes	Specifies whether changes to this column are logged. For more information, see Logging data changes on page 318.																		
Log changes when deleting	Specifies whether the column is to be logged when an object is deleted. For more information, see Logging data changes on page 318.																		

Property	Description
Export for SPML schema	Specifies whether the table is to be exported for the SPML schema. For more information, see Preparing the One Identity Manager schema for export to the SPML schema on page 497.
Not for export (XML export)	This column is not exported in data transports. The property is taken into account when data is transported between databases.
Not for import (XML import)	This column is not imported in data transports. The property is taken into account when data is transported between databases.
MVP column	This column is a multi-valued-property (MVP) containing individual value entries that are separated by char(7) or chr(7) .
Multi-value specification	<p>You can specify additional requirements on the values of an MVP column. You have the following options:</p> <ul style="list-style-type: none"> • Unique: Set this option if the value must be unique. • Case sensitive: Set this option if the case sensitivity should be taken into account when the value is tested. • Accent insensitive: Set this option if accent characters should not be taken into account when the value is tested. <p>For more information, see Specifying requirements for MVP columns on page 86.</p>
Multiline	Specifies whether the parameter contents can have multiple lines. Columns that are labeled with this option are displayed on a generic form with multiline input fields.
Permissions not issued automatically	For custom columns in a predefined table, permissions are not automatically assigned to predefined permissions groups, even though the Common AutoExtendPermissions configuration parameter is set.
Column contains description	One column with a description can be labeled with this option per table. The description is only displayed on user interface assignment controls.
Contains name properties for password check	Specifies whether the column contains name properties. Depending on the password policy configuration, columns with name properties may be included in the password check. For more information about password policies, see the <i>One Identity Manager Operational Guide</i> .
Column contains hierarchy information	One column which maps hierarchy information in readable form can be labeled with this option per table. The column is used to map the hierarchy to the user interface assignment controls.
Part of primary key	This column is part of the primary key.

Property	Description
Part of alternative primary key	Alternative primary keys are already specified in the default version, but the definition can be customized. Alternative primary keys are used for data transport amongst other things.
Part of the key of a many-to-all table	Identifier of the foreign key of a many-to-all table The foreign key and the dynamic foreign key of a many-to-all table are identified with this option.
Show in wizards	Specifies whether the column is offered in the Rule Editor for compliance rules to create queries and in the Web Portal to display tabular overviews.
Recursive key	<p>Specifies whether this column references a parent object. This input is needed for displaying hierarchical tables.</p> <p>Example:</p> <p>In the ADSContainer table, the UID_ParentADSContainer column contains the reference to the parent Active Directory container. The UID_ParentADSContainer column is labeled with this option in order to display this hierarchical link on forms.</p>
Encrypted	<p>Specifies whether the value in this column is encrypted. When the database is encrypted the value in this column is encrypted.</p> <p>NOTE: If you set this option on database columns, you must encrypt the database again. For more information about database encryption, see the <i>One Identity Manager Installation Guide</i>.</p>
Dynamic foreign key	Dynamic foreign keys refer to the object key in other tables. The object key comprises the table name and the values of the primary key of the actual object. Permitted tables can be limited. All tables are permitted, if there are no restrictions. For more information, see Dynamic foreign key on page 102.
No log	Specifies whether the column content is recorded in logs. For example, in the One Identity Manager Service log.
Proxy view column	If the column is used in a database view of the Proxy type, the corresponding column is entered in the view. For example, the column ADSDomain.DisplayName is mapped in the UNSRoot view to column RootObjectDisplay. For more information, see Database views of the proxy type on page 57.
Table Lookup Support	<p>Each value in these columns is prepared for fast table lookup support. The search is also supported by single values in MVP columns. The internal mapping of prepared data is done in the QBMSplittedLookup table.</p> <p>Permitted values are:</p>

Property	Description
	<ul style="list-style-type: none"> • Central user account (CentralAccount) • Email address (EMail) <p>You can extend the list of permitted values and customize the results.</p> <ul style="list-style-type: none"> • In the Designer, flag columns containing a user account name with the Central user account value in the Table lookup support property. • In the Designer, flag columns containing an email address with the Email address value in the Table lookup support property. <p>The functionality can be used for finding a unique central user account, for example, or a unique default email address for an employee. In the default installation, columns that are taken into account when the central user account or an email address are mapped are labeled with this property. The results are shown in the QERCentralAccount and QERMailAddress database views.</p>
Remarks (custom)	Text field for additional explanation.
Custom template/-formatting not permitted	Specifies whether column's the default configuration can be changed by the user, such as, templates, formatting, minimum length, maximum length, column format.
Max. length	Maximum length of the column. If the value is equal to 0 , the length from the database schema is used.
Minimum Length	Minimum length of the column. For columns that are displayed as required input fields in the administration tools user interface, set the minimum length to 1 or higher.
Column format	<p>Specify the format permitted for value in this column. For more information, see Using predefined formatting types on page 81.</p> <p>You can control the permitted format for the column with formatting types and formatting scripts.</p>
Overwrites	Specifies whether the template can overwrite or not. If this option is set, the value template is always applied. If the option is not set, the value template is only applied when the column is empty. For more information, see Editing value templates on page 76.
Template	Template script. Write the script in VB.Net syntax. This allows all VB.Net script functions to be used. For more information, see Editing value templates on page 76.
Threshold (stop)	Limit for the number of objects changed directly by a template.

Property	Description
	<p>Once this limit has been reached, processing is stopped with an error message. For more information, see Restricting performance of value templates on page 78.</p> <p>NOTE: If a stop threshold value is specified, it must be larger than the threshold for asynchronous processing.</p>
Threshold (asynchronous)	Limit for the number of objects changed directly by a template. Once this limit has been reached, processing takes place synchronously with the One Identity Manager Service. For more information, see Restricting performance of value templates on page 78.
No automatic truncation by template	Specifies whether the value is automatically truncated to the maximum column length if the maximum length is exceeded when applying a template. If this option is enabled, the value is not automatically truncated to the maximum column length. For more information, see Editing value templates on page 76.
Formatting script	Formatting script for the column. The formatting script for checking values is written in VB.Net syntax, which allows usage of all VB.Net script functions.
Visibility script	Script for conditional displaying of columns in One Identity Manager tools. For more information, see Scripts for conditionally displaying and editing columns on page 89.
Editability script	Script for conditional editing of columns in One Identity Manager tools. For more information, see Scripts for conditionally displaying and editing columns on page 89.
Foreign key	The column references an object in another table.
Average column length	Information is determined once a day through the maintenance tasks. The data material can help to plan capacities and maintenance work on the database.
Template changed	(Only for internal use) This indicates that the template was changed.
No DB Transport	Columns labeled with this option cannot be excluded from a custom configuration package. These columns are excluded from data transport.
Mapping direction	Specifies the mapping direction that will be applied to this column when synchronizing between two One Identity Manager databases. For more information about this, see the <i>One Identity Manager User Guide for the One Identity Manager Connector</i> .

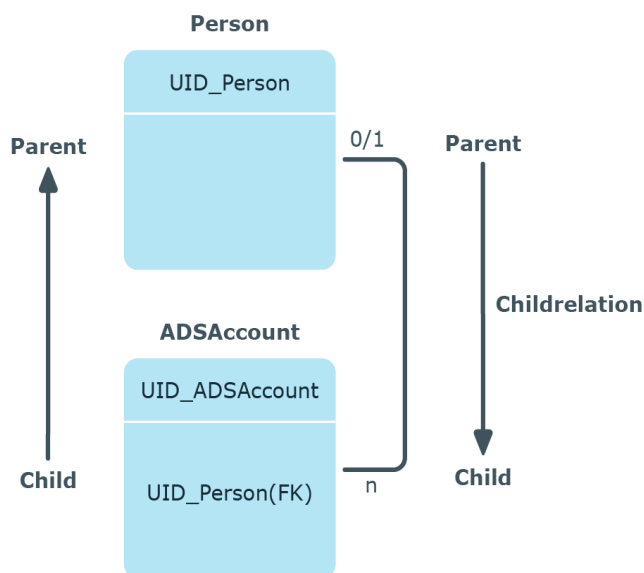
Related topics

- [Editing column definitions](#) on page 90
- [Visual Basic .NET scripts usage](#) on page 341

Table relations

As you can see from the One Identity Manager data model, parent/child relations exist between objects. If an object is processed by the One Identity Manager's object layer, all ForeignKey (FK) objects that are related to this object can be accessed. Use VB.Net notation to access objects access using relations.

Figure 11: Parent/child relation using the example of an employee ADSAccount



NOTE: You can always edit table relations of custom tables. The table relation supplied with the default tables can only be edited if the referential integrity has been tested using the object layer.

To edit table relations

1. In the Designer, select the **One Identity Manager schema** category.
2. Select the table and start the Schema Editor with the **Show table definition** task.
3. Under **Table relations**, select the table relation and edit the properties in the **Relation properties** view.
4. Select the **Database > Save to database** and click **Save**.

Table 28: Table relation properties

Property	Description
Display name	Language-dependent relation for displaying in the administration tool's user interface.
Only transport as group	<p>Specifies if the contents of the table should be transferred together with the contents of the referenced table during data transports. You can combine the values. Permitted values are:</p> <ul style="list-style-type: none">• No value: Dependencies are not taken into account.• CR direction: The table relations are labeled with the values CR direction and FK direction. Superset handling is carried out.• FK direction: All objects referenced by a foreign key are also exported. Superset handling is carried out.• Ignore in superset handling: Referenced objects that are in the target system but not included in the transport package are not deleted. <p>Example:</p> <p>When a process is transported (JobChain table), the process steps (Job table), events (JobEventGen and QBMEvent tables) and the process step parameters (JobRunParameter table) should also be transported. This should happen whether or not the process, a single process step or a process step parameter is transferred to a transport package. The table relations are labeled with the values CR direction and FK direction.</p> <p>The parameter templates (JobParameter table) that are used in the (JobRunParameter table) process step parameters must not be transferred during the transport. The table relations are not labeled with a value.</p>
Update dependencies modification date	When many-to-many entries are added, changed, or deleted, the value in the XDateSubItem column the associated parent entries is updated. Required for provisioning memberships in the target system.
Export for SPML schema	Specifies whether the table is to be exported for the SPML schema.
Parent object in Job queue	Specifies whether the parent object is added to the list of objects affected by a process. This can prevent the parent object from being processed simultaneously more than once.
Parent column	Unique parent column identifier.
Configurable parent relation	Specifies whether referential integrity can be configured.
Parent relation test instance	Specifies who will run these referential integrity tests. Permitted values are:

Property	Description
	<ul style="list-style-type: none"> • DLL: Checks through the object layer. • Trigger: Triggers and constraints are implemented to monitor the database. The triggers and constraints are created automatically and modified as necessary taking the preset restrictions of the DBQueue Processor into account. In the case of customized tables, specify the test instance and the limitations of the One Identity Manager schema extension. • Nothing: No test.
Parent relation constraint	<p>Constraint on the relation. Permitted values are:</p> <ul style="list-style-type: none"> • Delete: Dependencies are not taken into account on deletion. • Delete Cascade: All dependent objects are deleted when this object is deleted. • Delete Restrict: The object can only be deleted when no more references to other objects exist. • Delete Set NULL: When deleting the object, references to the object being deleted are removed from all dependent object (SetNULL).
Generated restriction test for parent relation	Identifier for triggers and constraints that are automatically generated by the DBQueue Processor.
Connected column	Unique connected column identifier.
Configurable child relation	Specifies whether referential integrity can be configured.
Child relation test instance	<p>Specifies who will run these referential integrity tests. Permitted values are:</p> <ul style="list-style-type: none"> • DLL: Checks through the object layer. • Trigger: Triggers and constraints are implemented to monitor the database. The triggers and constraints are created automatically and modified as necessary taking the preset restrictions of the DBQueue Processor into account. In the case of customized tables, specify the test instance and the limitations of the One Identity Manager schema extension. • Nothing: No test.
Child relation constraint	<p>Constraint on the relation. Permitted values are:</p> <ul style="list-style-type: none"> • Insert: Dependencies are not taken into account on insertion.

Property	Description
	<ul style="list-style-type: none"> • Insert Restrict: Checks for the referenced object when the object is added.
Generated restriction test for child relation	Identifier for triggers and constraints that are automatically generated by the DBQueue Processor.
Relation ID	Relation identifier. This is used for both directions.
M:N relation	Specifies whether the child relation can be reached by a many-to-many relation.
table relation	Unique identifier for table relation.
Relation (base)	Link to underlying base relation assuming a view is part of a the relation.
Relation (M:N)	Unique identifier for the M:N relation.

Related topics

- [Dynamic foreign key](#) on page 102
- [Displaying data models in the Designer](#) on page 47
- [Preparing the One Identity Manager schema for export to the SPML schema](#) on page 497

Dynamic foreign key

Dynamic foreign keys are used if a reference can point to different tables. For example, the manager of a user account (<MMM>Account.ObjectKeyManagertable) can be another user account (<MMM>Account table) or a group (<MMM>Group table).

Dynamic foreign keys reference the object key (XObjectKey) of the permitted tables. Permitted tables can be limited. All tables are permitted, if there are no restrictions. Restrictions are stored in the DialogValidDynamicRef table.

If you are defining custom dynamic foreign keys, at least one of the participating partners (dynamic foreign key column or referenced table) must be a custom object. It is not possible to extend predefined dynamic foreign keys by adding references to predefined tables.

To display a dynamic foreign key

1. In the Designer, select the **One Identity Manager schema** category.
2. Select the table and start the Schema Editor with the **Show table definition** task.
Dynamic foreign keys are displayed under **Dynamic table relations**.

To define a dynamic foreign key


1. In the Designer, select **One Identity Manager Schema**.
2. Select the table and start the Schema Editor with the **Show table definition** task.
3. Select the column and then the **Column properties** view.
4. On the **Miscellaneous** tab, enter the following information.
 - a. Enable the **Dynamic foreign key** option.
 - b. If the dynamic key is part of a many-to-all table, enable the **Part of key of many-to-all table** option.
5. Enter the following information on the **Valid reference tables** tab by clicking  next to **Dynamic referenced tables** menu and enter the following information:

Table 29: Properties of dynamic foreign keys

Property	Description
Table	Select the table to find the object key in.
Parent relation constraint	Constraint on the relation. Permitted values are: <ul style="list-style-type: none">• Delete: Dependencies are not taken into account on deletion.• Delete Cascade: All dependent objects are deleted when this object is deleted.• Delete Restrict: The object can only be deleted when no more references to other objects exist.• Delete Set NULL: When deleting the object, references to the object being deleted are removed from all dependent object (SetNULL).
Parent relation test instance	Specifies who will run these referential integrity tests. Permitted values are: <ul style="list-style-type: none">• DLL: Checks through the object layer.• Trigger: Triggers and constraints are implemented to monitor the database.
Child relation constraint	Constraint on the relation. Permitted values are: <ul style="list-style-type: none">• Insert: Dependencies are not taken into account on insertion.• Insert Restrict: Checks for the referenced object when the object is added.
Child relation test instance	Specifies who will run these referential integrity tests. Permitted values are: <ul style="list-style-type: none">• DLL: Checks through the object layer.• Trigger: Triggers and constraints are implemented to monitor the database.

Property	Description
Only transport as group	The column content is always transported together with the content of the referenced column.
Parent object in Job queue	Specifies whether the parent object is added to the list of objects affected by a process. This can prevent the parent object from being processed simultaneously more than once.

6. Select the **Database > Save to database** and click **Save**.

Related topics

- [Table relations](#) on page 99
- [Table types and default columns in the One Identity Manager data model](#) on page 49

Supporting file groups

One Identity Manager supports file groups to group tables together to help with administration, data assigning and data distribution. A distinction is made between logical disk stores and physical disk stores.

In the default installation, logical disk stores are predefined for the table in each module of One Identity Manager and the system tables. You cannot change the assignments. You can create your own logical disk storage for grouping custom tables.

To define logical storage for custom tables

1. In the Designer, select the **One Identity Manager Schema > Logical disk stores** category.
2. Select the **Object > New** menu item.
3. Enter a name and description for the logical storage.
4. Assign custom tables to the logical disk store.
5. Select the **View > Select table relations** menu item and enable the DialogTable table. This shows the **Tables** tab for assigning tables.

You can link logical storage with physical storage - the file groups - in the One Identity Manager schema.

If, for example, tables with employee data and tables with Active Directory content are created on different a data storage medium, performance can be improved by parallel access through your own E/A controller. Performance can also be improved if, for example, tables for processing DBQueue Processor tasks or table for handling processes in file groups are grouped together.

NOTE: You cannot move the following groups into other file groups. If you do so, proper functioning of the One Identity Manager database cannot be guaranteed.

- DialogColumn
- DialogTable
- DialogValidDynamicRef
- QBMDBQueueTask
- QBMDBQueueTaskDepend
- QBModuleDef
- QBModuleDepend
- QBMRRelation
- QBMViewAddOn
- QBMDiskStoreLogical
- QBMDiskStorePhysical

One Identity Manager supports the distribution of tables to file groups with a variety of database procedures that you run in a suitable program for running SQL queries in the database.

WARNING: Only carry out the following steps for implementing file groups, together with an experienced database administrator.

Ensure that the database cannot be accessed while file groups are being set up, for example, by the Job server, application server, web server, user interfaces, or the Web Portal. After restarting the DBQueue Processor, wait for all DBQueue tasks to be processed before reconnecting the database.

IMPORTANT: Select a user that you use for migrating the database to run the SQL queries.

To distribute tables to file groups under SQL Server

1. Create your file groups. For detailed information about this, see the documents for your currently installed version of SQL Server.
2. Synchronize the file groups to the One Identity Manager database. Run the query below using a suitable program for carrying out SQL queries in the database.

```
exec QBM_PDiskStorePhysicalSync
```
3. In the Designer, assign physical storage to logical storage.
 - a. In the Designer, select the **One Identity Manager Schema > Logical disk stores** category.
 - b. Select the logical disk store and in the **Properties** view, select the file group under **Physical disk store**.
 - c. Select the **Database > Save to database** and click **Save**.

4. Disable processing of DBQueue Processor tasks and process handling. Run the queries below using a suitable program for carrying out SQL queries in the database.

```
exec QBM_PWatchDogPrepare 1
```

```
exec QBM_PDBQueuePrepare 1
```

5. Move the tables into the configured file groups. Run the query below using a suitable program for carrying out SQL queries in the database.

```
exec QBM_PTableMove
```

6. Reactivate the DBQueue Processor. Run the queries below using a suitable program for carrying out SQL queries in the database.

```
exec QBM_PDBQueuePrepare 0,1
```

```
exec QBM_PWatchDogPrepare
```

Editing the user interface

Certain components of the One Identity Manager's graphical user interface are stored in the One Identity Manager schema and can be tailored to suit customer requirements. Menu items in the navigation structure, interface forms, and task definitions can be configured in this way.

Menu items, interface forms, and task definitions are assigned to permissions groups. The user's effective components of the user interface depend on the authentication module used for logging in to the One Identity Manager tools. If a user logs in to a One Identity Manager tool, a system user is found and the available menu items, interface forms, task definitions, and individual program functions are identified depending on the permission groups to which this system user belongs and the adapted user interface is loaded.

Data is displayed as objects in the user interface. User interface objects are meta-objects. You provide a selection of configurable elements that describes how the data stored in the database is perceived. These objects enable data to be distinguished by specific properties. They provide an additional control function for configuring the user interface. Hence, interface forms and tasks are linked to object definitions, which means that different forms and tasks are displayed in the user interface depending on which object is selected.

You can only modify the supplied user interface components to a certain extent and they are overwritten by schema installation. You can integrate components of the default user interface into your own user-defined user interface. If necessary you can disable individual components of the default user interface to stop them from being displayed. The system users provided are not effected by this limitation. Components labeled as disabled remain so after schema installation.

Captions are used in the user interface to create user friendly names for different components of the user interface such as menu items, tasks, and column names. You can maintain multi-language display text in One Identity Manager which enables you to display captions in different languages.

The default One Identity Manager installation is supplied in the **English - United States [en-US]** and **German - Germany [de-DE]** language. You can add other languages to the user interface and display text if required. In this instance, you must translate the text before One Identity Manager goes live. There is a Language Editor in the Designer to help you do this. A special control is provided in the One Identity Manager tools that aids multi-language input.

A user interface is always set up for one application. The standard version of One Identity Manager includes the applications and predefined navigation for the Manager, Designer, and Launchpad tools.

Detailed information about this topic

- [Object definitions for the user interface](#) on page 108
- [User interface navigation](#) on page 112
- [Forms for the user interface](#) on page 138
- [Statistics in One Identity Manager](#) on page 171
- [Extending the Launchpad](#) on page 187
- [Task definitions for the user interface](#) on page 191
- [Applications for configuring the user interface](#) on page 195
- [Icons and images for configuring the user interface](#) on page 198
- [Language-dependent data representation](#) on page 210

Object definitions for the user interface

The data in the user interface is represented by objects. Objects in the user interface map the data stored in the database. These objects can be configured and enable data to be distinguished by specific properties.

User interface forms and task definitions are linked to object definitions and displayed depending on the selected object definition. Object definitions provide an additional control function for configuring the user interface.

You can assign several objects to each table in the One Identity Manager schema. Basically, each database table should have at least one object definition that is generally valid, that means, without limited selection criterion. Other object definitions then relate to the respective special case limited by the general case.

TIP: To create object definitions for new tables, run the **Missing DialogObject** consistency check in the Designer and use the repair method. You must edit object definitions created like this afterward.

Table 30: Example relationship between tables and user interface object definitions

Table	Object definition	Limitation according to Object Definition
ESet	System roles (ESet)	None
ESet	System roles for IT Shop (ESet_ITShop)	System roles that can be excluded from the IT Shop

Detailed information about this topic

- [Selection criteria for object definitions](#) on page 109
- [Using the captions for object definitions](#) on page 109
- [Creating and editing object definitions](#) on page 110
- [Object definition properties](#) on page 111
- [Effects of object definitions when displaying interface forms](#) on page 145

Selection criteria for object definitions

The table entries to be displayed are found through a selection script and an object definition condition.

- Formulate a selection script as a VB.Net expression which returns either **True** (True) or **False** (False), depending on whether or not the transferred data record belongs to this object definition.
- Formulate a condition as a WHERE clause database query so that an object definition can also be used for display in result lists.

IMPORTANT: You must compile the database for the selection criteria to come into effect.

Example: Displaying system roles for the IT Shop

Selection script to determine at runtime whether this data record concerns a system role for the IT Shop:

Value = \$IsForITShop:Bool\$

Condition (WHERE clause) to determine whether this system role concerns the IT Shop:

IsForITShop=1

Related topics

- [Creating and editing object definitions](#) on page 110
- [Object definition properties](#) on page 111

Using the captions for object definitions

You can define the following captions to represent each object definition in the administration tool user interface.

- List caption

The list caption is used in One Identity Manager tools as the title for result lists. The display text of the object definition that you specified through the selected menu item, is used as the list title.

- Form caption

The form caption is used to display the current object definition, for example, in the Manager's status bar.

The current object definition is determined when an item in the administration tool result list is selected. Valid object definitions and thereby the possible captions are determined by selection scripts. From the possible display texts, the caption of the object definition with the lowest sort order is shown.

Example:

Table 31: Captions depending on the sort order of the object definitions

Object definition	Restrictions	Sort order	Caption
System roles (ESet)	None	99	System roles
System roles for IT Shop (ESet_ITShop)	System roles that can be excluded from the IT Shop	8	System roles for IT Shop

When you select an employee in the result list, the related caption is **System roles**. If the system role is also labeled for the IT Shop (IsForITShop=1), this object is assigned to another object definition by means of the VB.Net expression and the display text **System roles for IT Shop** is used.

Related topics

- [Object definition properties](#) on page 111
- [Editing lists](#) on page 130

Creating and editing object definitions

Predefined configurations are maintained by the schema installation and cannot be edited apart from a few properties.

To create or edit an object definition

1. In the Designer, select the **User interface > Object definitions** category.
2. Select one of the object definitions in the list.
- OR -
From the menu bar, add a new object definition using the **Object > New** menu item.
3. Enter the object definition's main data.
4. Select the **Database > Save to database** and click **Save**.

Related topics

- [Customizing the One Identity Manager default configuration](#) on page 28
- [Object definition properties](#) on page 111

Object definition properties

Table 32: Object definition properties

Property	Description
Exclusive	Objects labeled with this option are considered exclusive. That means, all other possible matching object definitions are not accepted as valid. If several object definitions of one table are labeled as exclusive, the object definition with the lowest sort order applies.
Display template	The display template specifies the form in which the data sets in the administration tool result lists are displayed.
Display name	The object's display name is used, for example, to identify the table in a database search or for error output. Display names can be given in more than one language.
List caption	Caption used to display the list title in the user interface.
Form caption	Caption used to display the form title in the user interface.
Selection script	Selection script as a VB.Net term, to determine during runtime whether the database object passed down belongs to this object definition. NOTE: The database needs to be compiled after changing modifying the selection script.
Processing status	Object processing status. The processing status is used for creating custom configuration packages.
Condition	Condition required for the object definition to be used for displaying in lists. You define the condition as a valid WHERE clause for database queries.

Property	Description
	NOTE: The selection script and the condition must match. If one of the properties is given then the other one also has to be given!
Remarks	Text field for additional explanation.
Disabled by preprocessor	If an object definition is excluded through a preprocessor condition, this option is set by the Database Compiler.
Insert values	Default settings for fields that are assigned when a new data set is added. The input is in VB.Net syntax. NOTE: The database needs to be compiled after changing modifying the values.
Background color	Color, with which the control for this object is displayed in the schema overview.
Object name	Name of the object.
Preprocessor condition	Object definitions can have preprocessor conditions added. This means, an object definition is only available when the preprocessor condition is fulfilled.
Sort order	The sort order is used for displaying the form title when an object is selected. The smaller the sort order magnitude, the stronger the restrictions defined for the object.
Icon	Icon for displaying the object definition.
Table	Table for which the object definition is created.

Related topics

- [Creating and editing object definitions](#) on page 110
- [Selection criteria for object definitions](#) on page 109
- [Using the captions for object definitions](#) on page 109
- [Display template for displaying a list](#) on page 132
- [Language-dependent data representation](#) on page 210
- [Defining insert values](#) on page 132
- [Conditional compilation using preprocessor conditions](#) on page 335
- [Icons and images for configuring the user interface](#) on page 198

User interface navigation

One Identity Manager administration tools with their own user interface are given their own navigation view. The navigation defines specific entry points into the One Identity Manager

tool's user interface and controls the user oriented navigation down to the selection of an object in the result list. You can set up the structure of the user interface navigation through a menu. There are different types of menu items with specific uses. You can design a multifaceted navigation by combining different types of menu items.

In the Designer, the navigation is displayed and edited in the **User interface > User interface navigation** category. The type of menu item determines the availability and editability of the properties.

Detailed information about this topic

- [Navigation elements](#) on page 113
- [Recommendations for editing menu navigation](#) on page 115
- [Tips for working with the User Interface Editor](#) on page 116
- [Selecting the user interface navigation view for editing](#) on page 116
- [Simulating user interface navigation during editing](#) on page 119
- [Copying existing user interface navigation for new permissions groups](#)
- [Copying menu items](#) on page 122
- [Creating a new user interface navigation](#) on page 121
- [Creating new menu items](#) on page 123
- [Creating new menu categories](#) on page 123
- [General menu item properties](#) on page 125
- [Creating database queries for data-dependent menu items](#) on page 128
- [Editing lists](#) on page 130
- [Using links in the navigation](#) on page 133
- [Using variables in the navigation](#) on page 135

Navigation elements

Table 33: Types of menu items

Type	Description
Menu category	Menu categories are displayed at the navigation top level and provide a method of grouping the data to be managed from different viewpoints. Menu categories constitute entry points into the interface navigation view. Menu categories are displayed as categories in the user interface.
Fixed menu item	Fixed menu items are used to organize data more clearly within menu categories. These menu items are always shown in the navigation view. List properties can only be defined for fixed menu items. These specify how the table entries are displayed in the user interface result list.

Type	Description
Data-Dependent Menu Item	Data-dependent menu items are generated by a database query that returns several data sets as output. These menu items are therefore not individual menu items, but a set of menu items depending on the output of the database query. List properties can be defined for data-dependent menu items. These specify how the table entries are displayed in the user interface result list.
Detached Menu Item	Detached menu items are used to group other menu items or to define a main menu item for an application. For example, you can specify a web interface home page with a detached menu item. Detached menu items should always be created at the navigation top level. However, they do not appear in the administration tools navigation view.
Link	Links support the navigation configuration. They are used to reference frequently accessed menu items. Parts of the navigation interface that require an application several times, only need to be set up once. The referenced menu items are always shown in navigation interface as opposed to the links.
Main Form Element	Main form elements are not menu items in the navigation view, but are used as the main elements in object overview forms. All child menu items are assigned to the main element.
Task category	Task categories are displayed at the navigation top level and are used to group together action-based processes. Task categories are not mapped in the navigation view but on a special form in the administration tools.
Task	Tasks are used to map single tasks within a task category. They are used, for example, as starting points for administration tool wizards. Tasks are always listed under a task category menu item. Task categories and their tasks are not displayed in the navigation view but on a special form.
Statistics	This menu item is used to display statistics. Statistics are typically displayed in the info system. All statistics that are defined in one menu level can be displayed on one form or as individual menu items. Statistics can also be included as form elements.

Related topics

- [Recommendations for editing menu navigation](#) on page 115
- [Tips for working with the User Interface Editor](#) on page 116
- [General menu item properties](#) on page 125
- [Creating database queries for data-dependent menu items](#) on page 128
- [Editing lists](#) on page 130
- [Using links in the navigation](#) on page 133
- [Working with overview forms](#) on page 162
- [Including statistics in the user interface](#) on page 176

Recommendations for editing menu navigation

- For fixed and database-dependent menu items you can specify list properties like display templates or object definition to be used. These properties determine how the table entries are displayed in the user interface result list.

TIP: You can define display templates for menu item, object definition, and table lists.

The display template is determined by the following in order:

1. List display template for the menu item
2. Object definition display template
3. Table display template

- Create menu items you can use as references (links). Thus, the parts of the navigation interface an application uses several times, only need to be created once. The referenced menu items are always shown in navigation interface as opposed to the links.
- Utilize variables in designing menu item names and display templates as well as in insert values and database queries.

TIP: Define the required variables in the menu item for the menu category. Variables are inherited within a hierarchical navigation. This means that variables in deeper levels of a hierarchy can be reused or overwritten. At run-time the actual value is passed to the variables.

- To display menu items in the user interface, assign the menu items to the program. For example, the **Manager**.
- Assign the menu items to permissions groups for non role-based and role-based login.

Related topics

- [Navigation elements](#) on page 113
- [Assigning menu items to applications](#) on page 124
- [Assigning menu items to permissions groups](#) on page 125
- [Editing lists](#) on page 130
- [Using links in the navigation](#) on page 133
- [Using variables in the navigation](#) on page 135

Tips for working with the User Interface Editor

Use the User Interface Editor to edit the navigation of the One Identity Manager tools. All menu items are hierarchically displayed in the navigation overview.

- You can use the User Interface Editor's wizard to create a preselection of menu items to be edited.
- Use "drag and drop" to move menu items around within the hierarchy.
- Use the **Options > Show captions** menu item to switch between the technical names of the menu items and the user-friendly captions.
- Use the **Options > Create menu markers** menu item to mark menu items. Define the menu items using a WHERE clause wizard. These are highlighted in red in the navigation overview. Use the **Options > Remove menu markers** menu item to remove the highlighting.
- You can display additional columns in the navigation overview using the **Options > Select columns** menu item.
- Use simulation mode to simulate the navigation view during editing.

Related topics

- [Selecting the user interface navigation view for editing](#) on page 116
- [Simulating user interface navigation during editing](#) on page 119

Selecting the user interface navigation view for editing

There are several ways of selecting the user interface navigation for editing. You can either load the entire user interface navigation, select the user interface navigation for an individual application or load the User Interface Editor wizard to pre-select menu items to be edited.

Detailed information about this topic

- [Loading a complete user interface navigation](#) on page 117
- [Loading menu navigation using an application](#) on page 117
- [Direct loading of menu items](#) on page 117
- [Loading menu items using permissions groups](#) on page 118
- [Loading menu navigation using a where clause](#) on page 119

Loading a complete user interface navigation

Select this task to load the entire user interface navigation for editing.

To load the entire user interface navigation

1. In the Designer, select **User interface > User interface navigation** category.
2. Select the **Modify user interface navigation** task.

The menu items are loaded and displayed in the User Interface Editor for editing.

Related topics

- [Loading menu navigation using an application](#) on page 117
- [Direct loading of menu items](#) on page 117
- [Loading menu items using permissions groups](#) on page 118
- [Loading menu navigation using a where clause](#) on page 119

Loading menu navigation using an application

Select this task to load the entire user interface navigation for editing. The standard version of One Identity Manager includes the applications and predefined navigation for the Manager, Designer, and Launchpad tools.

To load the navigation of an application

1. In the Designer, select **User interface > User interface navigation** category.
2. Select the **Manager**, the **Designer**, or the **Launchpad**.
3. Select the **Edit navigation for application** task.

The menu items are loaded and displayed in the User Interface Editor for editing.


Related topics

- [Loading a complete user interface navigation](#) on page 117
- [Direct loading of menu items](#) on page 117
- [Loading menu items using permissions groups](#) on page 118
- [Loading menu navigation using a where clause](#) on page 119

Direct loading of menu items

Use this task to select the menu items that you want to edit directly in the User Interface Editor wizard.

To select menu items directly

1. In the Designer, select **User interface > User interface navigation** category.
2. Select the **Load wizard to edit user interface navigation** task.
3. On the start page of the wizard, click **Next**.
4. On the **Select loading method** page, click .
5. The interface navigation of all applications from the database is displayed on the **Select navigation** page. Enable the menu items that you want to load.
6. Click **Finish** to complete the wizard.

The menu items are loaded and displayed in the User Interface Editor for editing.


Related topics

- [Loading a complete user interface navigation](#) on page 117
- [Loading menu navigation using an application](#) on page 117
- [Loading menu items using permissions groups](#) on page 118
- [Loading menu navigation using a where clause](#) on page 119

Loading menu items using permissions groups

Use this task in the User Interface Editor wizard to select the menu items that you want to edit through permissions groups.

To load menu items using permissions groups

1. In the Designer, select **User interface > User interface navigation** category.
2. Select the **Load wizard to edit user interface navigation** task.
3. On the start page of the wizard, click **Next**.
4. On the **Select loading method** page, click .
5. On the **Select permissions group** page, select the permissions groups whose menu items you want to load.

You can restrict the permissions groups through a specific system user or directly select the permissions groups.

6. Click **Finish** to complete the wizard.

The menu items are loaded and displayed in the User Interface Editor for editing.

Related topics


- [Loading a complete user interface navigation](#) on page 117
- [Loading menu navigation using an application](#) on page 117

- [Direct loading of menu items](#) on page 117
- [Loading menu navigation using a where clause](#) on page 119

Loading menu navigation using a where clause

With this task, you can select the menu items that you want to edit in the User Interface Editor wizard using a WHERE clause.

To load the user interface navigation using a WHERE clause

1. In the Designer, select **User interface > User interface navigation** category.
2. Select the **Load wizard to edit user interface navigation** task.
3. On the start page of the wizard, click **Next**.
4. On the **Select loading method** page, click .
5. On the **User-defined selection** page, enter the WHERE clause to load the navigation. Enter the WHERE clause manually or use the WHERE clause wizard.
6. Click **Finish** to complete the wizard.

The menu items are loaded and displayed in the User Interface Editor for editing.

Related topics

- [Loading a complete user interface navigation](#) on page 117
- [Loading menu navigation using an application](#) on page 117
- [Direct loading of menu items](#) on page 117
- [Loading menu items using permissions groups](#) on page 118

Simulating user interface navigation during editing

By simulating the user interface navigation in the User Interface Editor, you can see which menu items are displayed to system users because of their permissions group.

To simulate the navigation of an application

1. In the Designer, select **User interface > User interface navigation** category.
The menu items are loaded and displayed in the User Interface Editor for editing.
2. Define the simulation data.
 - a. In the User Interface Editor, select the menu **Simulation > Define simulation data**.

- b. On the start page of the wizard, click **Next**.
 - c. On the **Define simulation data** page, specify the following settings.
 - **System user for simulation:** Select the system user for whom you want to simulate the navigation.
 - **Application for simulation:** Select the application for which you want to simulate the navigation.
 - d. On the **Select base object** page, click **Next**.
 - e. To end the wizard, click **Finish** on the last page.
3. Start the simulation from the **Simulation > Start simulation** menu.


The application is opened in the simulation window.

NOTE: You can end the simulation at any time by closing the simulation window. Use **F9** to restart the simulation. The simulation data (system users and application) are retained.

Copying existing user interface navigation for new permissions groups

Using the User Interface Editor's wizard, you can select and copy the menu items from one permissions group for another permissions group. You can also optionally use the wizard to transfer the required permissions to tables and columns, as well as the object definitions and task definitions for the permission group.

To copy an existing user interface navigation

1. In the Designer, select **User interface > User interface navigation** category.
2. Select **Load wizard to edit user interface navigation**.
3. On the start page of the wizard, click **Next**.
4. On the **Select loading method** page, click .
5. On the **Select permissions group** page, select the permissions groups whose menu items you want to copy.

You can restrict the permissions groups through a specific system user or directly select the permissions groups.

6. On the **Define target permissions group** page, enter the following information.
 - **Copy to (new) permissions group:** Permissions group to which the individual elements of the navigation are copied.
 - Enter the name of the new permissions group. Ensure that your permissions groups begin with the customer prefix.
 - Select an existing permissions group.

- **Name prefix/suffix:** Additional labeling of menu items. At least a name prefix is required to create names for the new menu items. The name prefix is **CCC**. You can optionally enter a name suffix.
7. (Optional) Select copy options.
 - **Copy column permissions:** Copies the column permissions of the permissions group.
 - **Copy table permissions:** Copies the table permissions of the permissions group.
 - **Copy user interface form assignments:** Copies the user interface forms of the permissions group.
 - **Copy task assignments:** Copies the task definitions of the permissions group.
 8. To start copying, click **Next**.
 9. The copied elements are displayed on the **Copy menu data** page. The copy process can take some time depending on the number of selected parts. The components to be copied are displayed. Once processing is complete, click **Next**.
 10. Click **Finish** to complete the wizard.

The menu items are loaded and provided in the User Interface Editor for editing.

NOTE: After inserting, editing, or deleting a menu item, you must compile the database.


Related topics

- [Creating a new user interface navigation](#) on page 121
- [Assigning menu items to applications](#) on page 124
- [Assigning menu items to permissions groups](#) on page 125
- [General menu item properties](#) on page 125

Creating a new user interface navigation

Use this task and the User Interface Editor's wizard to create a new navigation with an initial menu category.

To create a new menu navigation

1. In the Designer, select **User interface > User interface navigation** category.
2. Select the **Load wizard to edit user interface navigation** task.
3. On the start page of the wizard, click **Next**.
4. On the **Select loading method** page, click .
5. Click **Finish** to complete the wizard.
6. Edit the main data of the menu category. Enter at least the name of the menu item.

7. Assign an application and the permissions groups.
8. Select the **Database > Save to database** and click **Save**.

NOTE: After inserting, editing, or deleting a menu item, you must compile the database.

Related topics

- [Copying existing user interface navigation for new permissions groups on page 120](#)
- [Creating new menu categories on page 123](#)
- [Assigning menu items to applications on page 124](#)
- [Assigning menu items to permissions groups on page 125](#)
- [General menu item properties on page 125](#)

Copying menu items

Use this task to copy a menu item from a user interface navigation and add it to another point in the navigation.

To copy and paste a menu item

1. In the Designer, select **User interface > User interface navigation** category.
2. Select the **Manager**, the **Designer**, or the **Launchpad**.
3. Select the **Edit navigation for application** task.
The menu items are loaded and displayed in the User Interface Editor for editing.
4. Select the menu item you want to copy in the navigation overview.
5. Select one of the copy options in the context menu.
 - **Copy:** Select this option to copy the selected menu item.
 - **Copy with child items:** Select this option to copy the selected menu item and its submenu items.
6. Select the menu item under which you want to create the menu item in the navigation overview.
7. Select **New**.
8. Edit the main data of the menu item.
9. Assign an application and the permissions groups.
10. Select the **Database > Save to database** and click **Save**.

NOTE: After inserting, editing, or deleting a menu item, you must compile the database.

Related topics

- [Selecting the user interface navigation view for editing](#) on page 116
- [Creating new menu items](#) on page 123
- [Creating new menu categories](#) on page 123
- [Assigning menu items to applications](#) on page 124
- [Assigning menu items to permissions groups](#) on page 125
- [General menu item properties](#) on page 125

Creating new menu items

Use this task to create a new menu item in an existing user interface navigation.

To create a new menu item

1. In the Designer, select **User interface > User interface navigation** category.
The menu items are loaded and displayed in the User Interface Editor for editing.
2. Select the menu item under which you want to create the menu item in the navigation overview.
3. Select the **New** context menu item.
4. Edit the main data of the menu item.
5. Assign an application and the permissions groups.
6. Select the **Database > Save to database** and click **Save**.

| NOTE: After inserting, editing, or deleting a menu item, you must compile the database.

Related topics

- [Selecting the user interface navigation view for editing](#) on page 116
- [Creating a new user interface navigation](#) on page 121
- [Creating new menu categories](#) on page 123
- [Copying menu items](#) on page 122
- [Assigning menu items to applications](#) on page 124
- [Assigning menu items to permissions groups](#) on page 125
- [General menu item properties](#) on page 125

Creating new menu categories

Use this task to create a new menu category in an existing user interface navigation.

To create a new category

1. In the Designer, select **User interface > User interface navigation** category.
The menu items are loaded and displayed in the User Interface Editor for editing.
2. Select the **Menu item > New navigation category** menu item.
3. Edit the main data of the menu category. Enter at least the name of the menu item.
NOTE: If the entry is to represent a task category, change the entry type of the menu item to **Task category**.
4. Assign an application and the permissions groups.
5. Select the **Database > Save to database** and click **Save**.

NOTE: After inserting, editing, or deleting a menu item, you must compile the database.

Related topics

- [Selecting the user interface navigation view for editing on page 116](#)
- [Creating new menu items on page 123](#)
- [Creating a new user interface navigation on page 121](#)
- [Assigning menu items to applications on page 124](#)
- [Assigning menu items to permissions groups on page 125](#)
- [General menu item properties on page 125](#)

Assigning menu items to applications

All menu items to be displayed in an application user interface must be assigned to an application.

To assign a menu item to an application

1. In the Designer, select **User interface > User interface navigation** category.
The menu items are loaded and displayed in the User Interface Editor for editing.
2. Select the menu item in the navigation overview.
3. In the edit view, select the **Application** view and then the application.
4. Select the **Database > Save to database** and click **Save**.

TIP: Use the **Recursively assign application** context menu to assign the application to the selected menu item and its child menu items. Use the **Recursively remove application** context menu to remove the application's assignment to the selected menu item and its child menu items.

Related topics

- [Assigning menu items to permissions groups](#) on page 125
- [Applications for configuring the user interface](#) on page 195

Assigning menu items to permissions groups

All menu items to be displayed in an application user interface must be assigned to a permissions group. Assign the menu items to permissions groups for non role-based and role-based login. The menu items are available to system users depending on their permissions group memberships. For more information about permissions groups, see the *One Identity Manager Authorization and Authentication Guide*.

To assign a menu item to a permissions group

1. In the Designer, select **User interface > User interface navigation** category.
The menu items are loaded and displayed in the User Interface Editor for editing.
2. Select the menu item in the navigation overview.
3. In the edit view, select the **Permissions group** view and select the permissions groups.
4. Select the **Database > Save to database** and click **Save**.

TIP: Use the **Assign permissions group recursively** context menu to assign the permissions group to the selected menu item and its child menu items. Use the **Remove permissions group recursively** context menu to remove the permissions group assignment to the selected menu item and its child menu items.

Related topics


- [Assigning menu items to applications](#) on page 124

General menu item properties

The properties described below are valid for all menu items: Other properties may be required for different menu item types.

Table 34: General menu item properties

Property	Description
Menu item	Unique menu item relation. You should assign descriptive names here if possible. These are then propagated in the child structures. This makes

Property	Description
	it easier to trace the position of child menu items. The parent menu item and the hierarchy is determined by the insert position in the user interface navigation. The menu item name can contain variables in order to represent the menu items.
Entry type	Menu item entry type.
Caption	Language-dependent caption for displaying the menu item in the user interface. The caption for data-dependent menu items can contain fixed strings and variables. The caption for recursive data-dependent menu items is inherited from the parent menu item. Translate the given text using the  button.
Sort order	If several menu items have the same parent menu item, the sort order of the individual menu items specifies their position in the display order. If the configuration flag Re-sort data submenu items by caption is set for the parent menu item, the sort order specified here is invalid.
Icon	Icon for displaying the menu item in the navigation. If no icon is specified for recursively data-dependent menu items, the icon from the parent menu item is inherited.
Overlay icon definition	VB.Net expression for defining overlays for the icon. Used to display the status in the Launchpad.
Condition	Specifies the conditions under which the menu item is displayed in the navigation. The input must satisfy the WHERE clause database query syntax. You can use variables to formulate a condition.
Configuration flag	Special functions are set for menu items with the configuration flag. For more information, see Table 35 on page 127.
Preprocessor condition	You can add preprocessor conditions to menu items. This means that a menu item is only available when the preprocessor condition is fulfilled. NOTE: In the Designer, you can find an overview of existing preprocessor dependencies in the One Identity Manager Schema > Preprocessor dependencies category.
Disabled by preprocessor	If a menu item is excluded through a preprocessor condition, this option is set by the Database Compiler.
Description	Text field for additional explanation.
Deactivated	Specifies if the menu item is displayed in the user interface or not. Disabled menu items are never displayed in the user interface. NOTE: This change is also permitted for menu items in the default user interface and is not overwritten on schema installation.
Show under	This option marks the menu items in the Manager to be displayed in the

Property	Description
"My One Identity Manager"	My One Identity Manager category.

Table 35: Configuration flags for special functions

Configuration flag	Description
Auto-reload on insert	If this configuration flag is set, the menu item is reloaded after new data is added.
Hide on empty result	If no submenu items are generated for a menu item labeled the same way during runtime, the menu item is similarly hidden in the user interface.
Not expandable by user	Menu items that are labeled with this option cannot be opened even if submenu items are available. The configuration flag is mainly used in the info system for displaying statistics.
Ignore user interface forms in result list	No forms are provided in the result list for menu items with this option. This can be useful to prevent navigating to objects in the list on an overview form. This is useful if, for example, forms are not defined for some objects in the result list. Otherwise, an empty form is displayed.
Ignore user interface forms	This configuration flag can be used for data-dependent menu items. If the configuration flag is set, no object-dependent interface forms are displayed when the menu item is selected in the user interface. This configuration flag is mainly used for structuring the user interface for Web front-ends.
Force open menu item	If this configuration flag is set, the menu item is always open. There is no test to see if the menu item is assigned to something, for example, the interface form.
Re-sort data-dependent menu item by caption	This configuration flag can be used for data-dependent menu items. The configuration flag should be set if language-dependent data is displayed. If the configuration flag is set, the data for menu navigation to be shown is sorted by language after it is loaded.
Re-sort data result by caption	This configuration flag can be used for lists. The configuration flag should be set if language-dependent data is displayed. If the configuration flag is set, the data to be shown is sorted by language in the result list after it is loaded.
Re-sort data submenu items by caption	<p>The configuration flag should be set if language-dependent data is displayed. If the configuration flag is set, the data for all submenu items to be shown are sorted by language.</p> <p>This enables all user accounts, groups, and containers in a container structure, for example, to be sorted alphabetically. The sort order not</p>

Configuration flag	Description
	only affects data-dependent menu items but also all submenu items.
Take navigation context into account on enabling	<p>If this configuration flag is set, the location in the navigation where the menu item is opened is taken into account when the objects are loaded. If an object appears several times within a navigation structure, the content is loaded and displayed depending on the location of the navigation structure.</p> <p>If the configuration flag is not set, the data is retained, even if the object is opened again from another location in the navigation.</p>

Related topics

- [Navigation elements](#) on page 113
- [Creating database queries for data-dependent menu items](#) on page 128
- [Editing lists](#) on page 130
- [Using links in the navigation](#) on page 133
- [Working with overview forms](#) on page 162
- [Including statistics in the user interface](#) on page 176
- [Extending the Launchpad](#) on page 187
- [Using variables in the navigation](#) on page 135
- [Icons and images for configuring the user interface](#) on page 198
- [Conditional compilation using preprocessor conditions](#) on page 335

Creating database queries for data-dependent menu items

Data-dependent menu items are generated by a database query that returns several data sets as output. These menu items are therefore not individual menu items, but a set of menu items depending on the output of the database query.

For more information about general properties of menu items, see [General menu item properties](#) on page 125. The following properties are necessary to put together a database query:

Table 36: Database query properties

Property	Description
Table	Table that the values are read from.

Property	Description
Sort order	<p>Display elements are sorted by these table columns. The input must satisfy the Order By syntax of database queries. Sorting is given by the columns of the display template if no value is entered. You should use a sort order if the data has a date or represents language-dependent data.</p> <p>NOTE: For language-dependent sorting, use the Re-sort data dependent menu item by caption configuration switch.</p>
Condition	<p>Condition for limiting the number of results displayed. The input must satisfy the WHERE clause database query syntax. You can use variables for formulating a condition. If the menu items are recursively data-dependent then variables have to be used.</p> <p>NOTE: The condition must not contain a JOIN and the query may need to be formulated as a subquery.</p>
Unique	<p>The query result cannot contain doubled items. By setting option, any doubt is eliminated.</p> <p>Menu items that are labeled with the Unique option have to contain variables in their names to achieve uniqueness.</p> <p>If, for example, all software applications (Application table) are grouped by language, the name of the corresponding menu item must contain a variable, which references the UID_DialogCulture column in the Application table.</p> <p>NOTE: No interface forms are shown for objects that result from a database query.</p> <p>NOTE: The option is disabled if the configuration switch Force open menu item is set.</p>
Recursive invocation	<p>This menu item is the recursive successor of the previous menu item. If the option is not set, the results are represented by a flat structure. Set the option if the menu item is required to represent a hierarchical structure. You have to define recursive data-dependent menu items below a data-dependent item without recursion.</p>

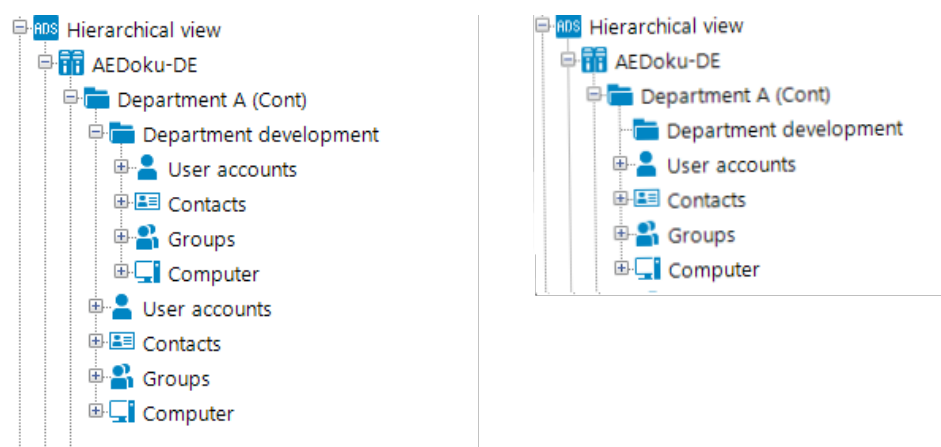
Related topics

- [Creating new menu items](#) on page 123
- [General menu item properties](#) on page 125
- [Recursive data-dependent menu items](#) on page 130
- [Using variables in the navigation](#) on page 135

Recursive data-dependent menu items

The heart of the hierarchy is variable replacement. Variables are passed down through the hierarchical navigation view and can therefore be used at lower levels or can be overwritten. In the case of recursive data-dependent menu items, the variable contained in a database query is initially replaced by the existing variable value from the parent level and then the query is started. The resulting value immediately determines a new value for the variable that is processed again in the parent node's next step. The original value of the old variable is no longer available after the database query has been run. If the database query delivers an empty result, the recursion is stopped.

Figure 12: Example of data-dependent menu items with recursive calling (left) and without recursive calling (right)



Related topics

- [Creating database queries for data-dependent menu items](#) on page 128
- [Using variables in the navigation](#) on page 135

Editing lists

You can apply list properties to fixed and data-dependent menu items. These properties determine how the table entries are displayed in the user interface result list.

For more information about general properties of menu items, see [General menu item properties](#) on page 125. To define a list, you need to use the following properties:

Table 37: List properties

Property	Description
Display	The display template for displaying table entries in the administration tool

Property	Description
template	result lists are displayed. If a customer specific display template exists it is used instead of the default display template. Syntax: %column name%
Object	Definition of the object which determines the list items.
Condition	Condition for limiting the number of results. The input must comply with the WHERE clause syntax of database queries. The condition relates to the given object definition. The condition is consolidated with the condition which is already stored for the object definition. The variables can be used that are available in the navigation interface.
Icon	Icon for displaying the items in the list.
Sort order	Columns to use for the list order. The input must satisfy the Order By syntax of database queries. You should use a sort order if the data has a date or represents language-dependent data. For language-dependent sorting, use the configuration switch Re-sort data result by caption .
Insert values	Insert values initialize individual values when a new data set is added over the result list. Enter insert values in VB.Net syntax. When defining insert values, you can apply the variables currently available in the navigation.
Insert in list permitted	Specifies whether you are generally allowed to insert entries in the corresponding result list Whether or not users are allowed to insert entries depends on their permissions. For more information about assigning permissions, see the <i>One Identity Manager Authorization and Authentication Guide</i> .
Permit deletion in list	Specifies whether you are generally allowed to delete entries in the corresponding result list. Whether or not users are allowed to delete entries depends on their permissions. For more information about assigning permissions, see the <i>One Identity Manager Authorization and Authentication Guide</i> .

Related topics

- [Creating new menu items](#) on page 123
- [General menu item properties](#) on page 125
- [Display template for displaying a list](#) on page 132
- [Defining insert values](#) on page 132
- [Using variables in the navigation](#) on page 135
- [Language-dependent data representation](#) on page 210
- [Object definitions for the user interface](#) on page 108

Display template for displaying a list

You use a list display template to specify the form in which the table entries will be represented in the administration tool result list. You can define display templates for menu items, object definitions and table lists.

The display template is determined by the following in order:

1. List display template for the menu item
2. Object definition display template
3. Table display template

The display template for displaying a list can be described in the following syntax:

`%columnname%`

All the columns that belong to the table that will be displayed can be used in the display template. Variables may not be used in display templates for lists.

Replacing the display template supports the `??` operator. Thus you can formulate conditional display templates with the following syntax.

`%columnname1??columnname2??columnname3%`

`%columnname1?? columnname2%`

The first column that returns a value from the list of column names is used. Spaces are permitted before and after the `??` operator. Spaces are not allowed at the beginning and end of the conditional display template for performance reasons.

Example:

The Active Directory user account (ADSAccount table) should be shown as follows:

Common Name (fully qualified domain name)

The display template for the ADSAccount table to be specified for this purpose is:

`%cn% (%CanonicalName%)`

Related topics

- [Editing lists](#) on page 130
- [Editing parameter value definitions](#) on page 422

Defining insert values

You can use insert values to initialize individual values when a new data set is added over the result list. You can apply insert values to interface forms, object definitions, menu item lists, and tables.

Enter insert values in VB.Net syntax. The `Base.` syntax Always accesses the object that is currently loaded. Insert values are described with the following syntax:

- Simple value assignment
`Base.PutValue("<column>", <value>)`
- Value assignment with variable replacement (value must be a character string)
`Base.PutValue("<column>", context.Replace(<value>))`

All the columns of the table to be displayed may be applied. You can use variable for defining insert values.

Example:

```
Base.PutValue("IsITShopOnly", 1)
Base.PutValue("UID_ADSTContainer", context.Replace("%cont%"))
```

| NOTE: If you changed insert values, you must recompile the database.

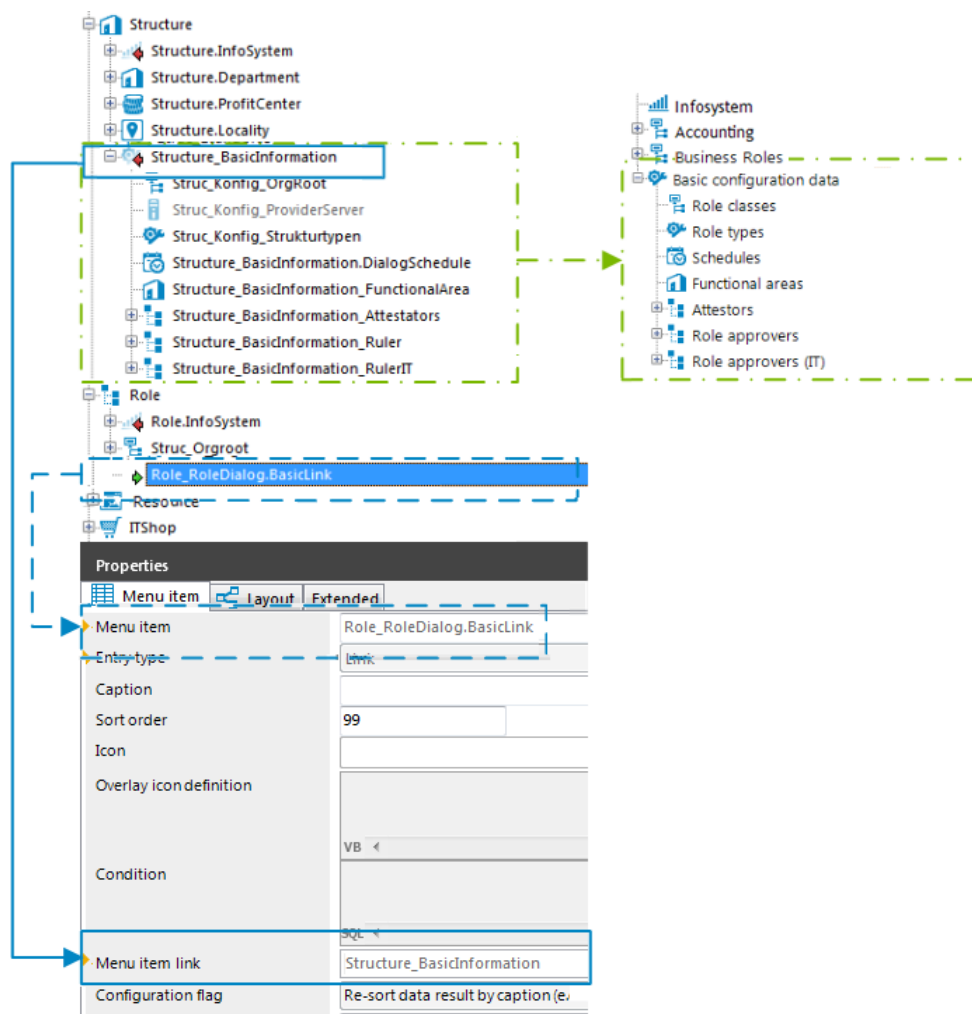
Related topics

- [Using variables in the navigation](#) on page 135

Using links in the navigation

Links support the navigation configuration. Links are implemented to reference frequently used menu items. Parts of the navigation interface that require an application several times, only need to be set up once. The links themselves do not appear in the navigation. Instead the referenced menu items and their child menu items are shown.

Figure 13: Structure of the navigation interface using links in the User Interface Editor (left) and display in the Manager (right).



Special features of using links

- Links inherit some properties of the reference entry.
- You can use variables in the reference entry, for example in conditions for lists or data-dependent menu entries. Value assignment to the variables only takes place in the link. You must define the variables in the link.
- The caption and the icon of the reference entry are overwritten with the corresponding values from the link.

To use links

1. Create the menu item that you want to use as the reference entry.
2. If necessary, create other menu items below the reference entry.

3. Create the menu items that link to the reference entry. Enter at least the following information for the link.
 - **Menu item:** Enter the name of the menu item.
 - **Entry type:** Select the **Link** entry type.
 - **Menu item link:** Select the reference entry to be shown at runtime when the link is called.
4. Assign an application and the permissions groups.
5. Select the **Database > Save to database** and click **Save**.

TIP:

- If the menu item is of the **Link** type, you can use the **Follow link node** context menu to navigate to the reference entry.
- For a reference entry, you can use the **Referenced by** context menu to display all links that refer to this reference entry and then navigate to these entries.

Related topics

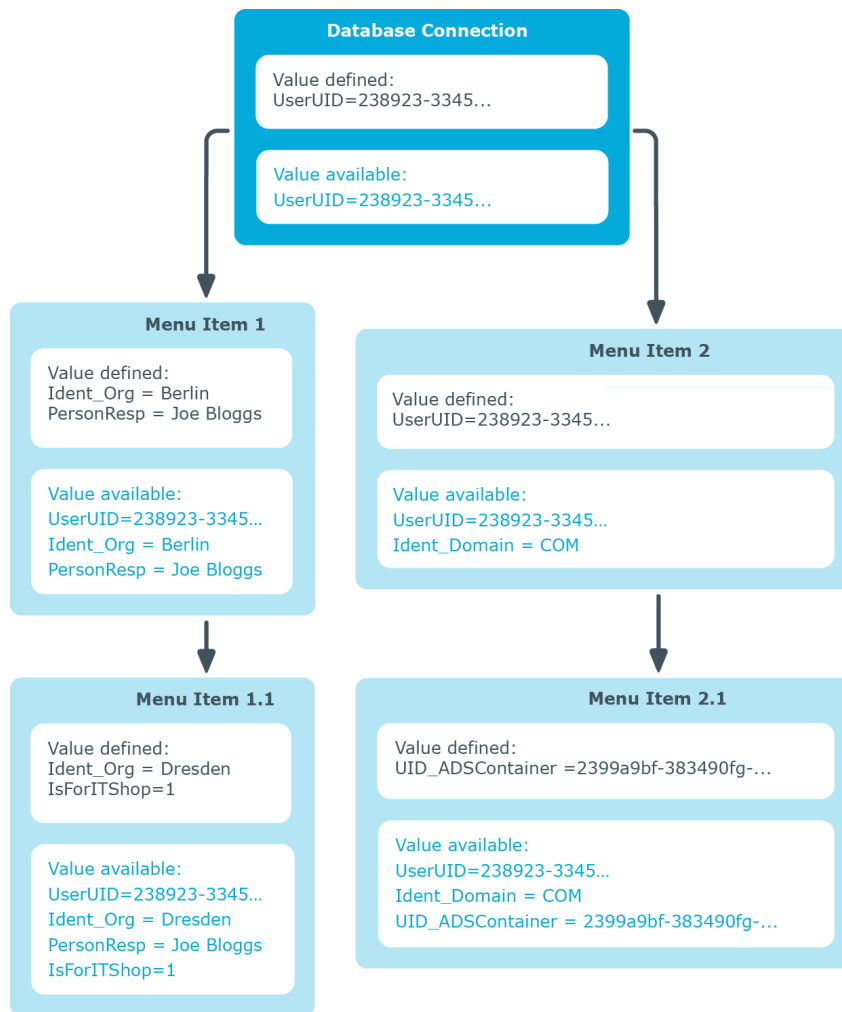
- [Creating new menu items](#) on page 123
- [General menu item properties](#) on page 125
- [Using variables in the navigation](#) on page 135

Using variables in the navigation

You may use variables to configure identifiers and menu item display templates for menu items in insert values and database queries. In some parts of the navigation interface you have to implement variables as, for example, in the case of formulating database queries for recursive data-dependent menu items.

Variables are inherited within a hierarchical navigation. This means that variables in deeper levels of a hierarchy can be reused or overwritten. The actual run-time value is passed to the variable.

Figure 14: Inheriting variables in a hierarchical navigation interface



The variables of the session object that are listed below are always available when the menu items are being set up.

Table 38: Global session object variables

Variable	Meaning
EnvUserName	Name of user to be authenticated in the environment, for example, Domain\User in Active Directory
LogonUser	DialogUser.Username of the currently logged in user.
DialogUserID	DialogUser.UserID of the logged in user.
UserName	Name displayed in XUserInserted or XUserUpdated.
UserID	Logged in user's UID_Person, if user related authentication is being used.
ShowCommonData	Specifies whether system data is shown (1) or not shown (0) The

Variable	Meaning
	variable is evaluated in the Designer by the program settings.
SessionType	<p>Specifies whether a direct database connection or a connection over an application server is supported.</p> <p>Direct database connection only: '%SessionType%' = 'Direct'</p> <p>Connect with the application server only: '%SessionType%' = 'AppServer'</p>

Use the following syntax to access the variables:

%<variable>%

Related topics

- [Creating and displaying variables](#) on page 137
- [General menu item properties](#) on page 125
- [Creating database queries for data-dependent menu items](#) on page 128
- [Editing lists](#) on page 130
- [Using links in the navigation](#) on page 133
- [Querying session object global variables](#) on page 353

Creating and displaying variables

In addition to the variables belonging to the session object, you can also define other variables. The variable definition is made up of variable type, variable name and the value. Basically, any string is permitted in the variable definition. However, events have proved that it is a good idea to use a pattern that is unlikely to occur in the data but is accepted as a string by the database server in use.

Use the following syntax to access the variables:

%<variable>%

Table 39: Variable definitions

Variable Type	Variable name	Value	Usage
Column	Any string	Current object's column name	Only used in data-dependent menu items.
Display value	Any string	Current object's	Only used in data-dependent menu items. The Multilingual and List of permitted values


Variable Type	Variable name	Value	Usage
		column name	column properties are resolved when creating the display value for a column.
Text	Any string	Freely defined value	Can be used in all menu items.

To create variables

1. In the Designer, select **User interface > User interface navigation** category.
The menu items are loaded and displayed in the User Interface Editor for editing.
2. Select the menu item in the navigation overview.
3. In the edit view, select **Variable definitions**.

In this view, all the variable definitions that belong to the selected menu item are displayed in tabular form with type, name, and assigned value.

TIP: To display variables inherited from parent nodes, click .

4. To create a variable, click  and enter the following information.
 - **Type of variable:** Select **Column**, **Display value**, or **Text**.
 - **Variable:** Enter the name of the variable.
 - **Value:** Enter the value of the variable. The value to be entered depends on the variable type.
5. Select the **Database > Save to database** and click **Save**.

The actual value stored in the variable can be shown in the administration tools as additional navigation information.

To display variable values of a menu item in the Manager

- In the Manager, enable the **Show additional navigation information** program setting.
- In the Manager, select menu item in the navigation and select the **Definition > Defined variables** context menu item.

Related topics

- [Using variables in the navigation](#) on page 135

Forms for the user interface

User interface forms are used to display and edit data in the user interface. The basic information for representing data on the user interface forms is described in form

definitions and form templates. The form definition referenced by the interface form needs to be found. The form template given in the form definition is checked for existence in the form archive and to see if it is labeled for the correct display purposes.

Detailed information about this topic

- [Recommendations for editing forms](#) on page 139
- [Editing user interface forms](#) on page 140
- [Forms for custom extensions](#) on page 147
- [Working with overview forms](#) on page 162

Recommendations for editing forms

- If necessary, you can disable individual predefined forms to prevent them being shown in the user interface. They remain disabled even after schema installation.
- The default installation of One Identity Manager already provides a series of form templates and definitions, for example for editing main data as well as many-to-many relations and object relations (Parent/ChildRelation). These can be used for easily creating your own forms.
- To display information about a base object, you create an overview form.
 - You can do this using the Overview Form Editor in the Designer.
 - Create menu items for object relations you need to display frequently, and use these menu items as reference in the form elements of the overview form.

TIP: You can have the Overview Form Editor create the menu items for object relations.

- Select the object relation you want to display and drag and drop it on an element in the element area of the Overview Form Editor.
- Use the context menu items **Create list element reference** or **Create reference to data element**.

The menu items are entered below the InfoSheets.QIM.Links menu item with the InfoSheet.List.<table> and InfoSheet.Node.<table> labels, respectively.

The condition for the menu items is defined as the %<table>WhereClause% variable. In the form element you assign a condition as WHERE clause to the variable.

- Default forms can be used to customize column extensions on default tables under certain conditions.
- To edit the main data of custom tables, use the Designer's Form Editor to create an interface form with the **VI_Generic_MasterData** form definition.

- To define mappings, create additional interface forms with the **MemberRelation** form type.
- Assign the forms and menu items to the application, such as the **Manager** program.
- Assign the forms and menu items to the permissions groups for non role-based and role-based login.
- If necessary, you can provide your own form templates in a form archive (*.Forms.vif).

Related topics

- [Editing user interface forms](#) on page 140
- [Disabling user interface forms](#) on page 141
- [Creating user interface forms](#) on page 142
- [Displaying custom columns and tables on main data forms](#) on page 146
- [Forms for custom extensions](#) on page 147
- [Replacing default forms with custom forms](#) on page 150
- [Working with overview forms](#) on page 162

Editing user interface forms

User interface forms are connected to object definitions, so that different forms are offered in the user interface depending on which object is selected. These interface forms are made available to system users, taking into account their permissions group memberships, by the additional assignment of interface forms to permissions groups. Further more, interface forms can be defined for separate menu items. When the associated menu item is selected in the navigation or the item is selected in the result list, the interface forms are shown for all system users without taking their permissions group memberships into account.

Predefined configurations are maintained by the schema installation and cannot be edited apart from a few properties.

NOTE: You can disable individual predefined tasks to prevent them being shown in the user interface. They remain disabled even after schema installation.

Related topics

- [Tips for working with the Form Editor](#) on page 141
- [Disabling user interface forms](#) on page 141
- [Copying user interface forms](#) on page 142
- [Creating user interface forms](#) on page 142
- [Displaying custom columns and tables on main data forms](#) on page 146
- [Forms for custom extensions](#) on page 147

Tips for working with the Form Editor

Use the Form Editor to create and edit the interface forms, such as the main data forms or assignment forms. All user interface forms are displayed in the form overview.

- Forms that are disabled by preprocessor conditions are grayed out in the form overview.
- In the Form Editor, use **F5** to refresh the form overview.
- Forms can be displayed hierarchically or in a list. The interface forms are grouped by form template and form definition in the hierarchical representation. To change the display, select the **Options > Tree/List view** menu item.
- Using the **Options > Show captions** menu, you can switch between the forms' technical names and the user-friendly captions.
- To display additional columns in the form overview, use the **Options > Select columns** menu.
- Define filters to restrict the number of forms displayed in the form list. Select the menu items Define filter or Manage filters for this purpose. For more information about working with user-defined filters in the Designer, see *One Identity Manager User Guide for One Identity Manager Tools User Interface*.
- Use the form preview while editing main data forms. In the Form Editor, select the **View > Form preview** menu item to display an additional **Form preview** tab.

The form preview shows the contents of the interface form. You can see which base tables will be used to display the data. The permissions of the logged in the Designer user are taken into account when loading and displaying an interface form.

If a form cannot be loaded, an appropriate error message is displayed.

Disabling user interface forms

If required, you can disable individual user interface forms to prevent them being shown in the user interface. Predefined user interface forms remain disabled even after the schema has been updated.

To disable a user interface form

1. In the Designer, select the **User interface > Forms > User interface forms** category.
2. Select the **Edit form** task.
3. Select the user interface form in the Form Editor.
4. In the edit view, select the **Properties** view.
5. Select the **User interface forms** tab and set the **Disabled** option.
6. Select the **Database > Save to database** and click **Save**.

In addition, user interface forms can be disabled using pre-processor conditions.

Related topics

- [User interface form properties](#) on page 151

Copying user interface forms

Use this task if you want to make minor modifications only, such as changing the caption or the sort order.

To copy a user interface form

1. In the Designer, select the **User interface > Forms > User interface forms** category.
2. Select the **Edit form** task.
3. In the Form Editor, select the user interface form you want to copy.
4. Select the **Form > Insert** menu item.
This creates a copy of the selected user interface form.
5. Edit the other user interface form main data.
6. Assign the user interface form to the applications and permissions groups.
7. (Optional) Assign the user interface form to the object definitions.
8. (Optional) Assign the user interface form to the menu items.
9. Select the **Database > Save to database** and click **Save**.

NOTE: Disable the original user interface form. Otherwise both forms are displayed in the user interface.

Related topics

- [Disabling user interface forms](#) on page 141
- [User interface form properties](#) on page 151
- [Assigning user interface forms to applications](#) on page 143
- [Assigning user interface forms to permissions groups](#) on page 144
- [Assigning user interface forms to object definitions](#) on page 144
- [Assigning user interface forms to menu items](#) on page 146
- [Creating user interface forms](#) on page 142

Creating user interface forms

Create a new user interface form, for example, if you want to display custom schema extensions in the user interface. One Identity Manager provides an array of form templates

and definitions in the default installation. These can be used for easily creating your own forms.

To create a new user interface form

1. In the Designer, select the **User interface > Forms > User interface forms** category.
2. Select the **Edit form** task.
3. Select the **Form > Insert** menu item.
The Form Editor opens a **new sheet** form in the edit view.
4. On the **Form definition** tab, select a form template and enter the name of the form definition.
5. On the **User interface form** tab, enter a form name and the caption. Edit the other main data of the user interface form.
6. Assign the user interface form to the applications and permissions groups.
7. (Optional) Assign the user interface form to the object definitions.
8. (Optional) Assign the user interface form to the menu items.
9. Select the **Database > Save to database** and click **Save**.

Related topics

- [User interface form properties](#) on page 151
- [Assigning user interface forms to applications](#)
- [Assigning user interface forms to permissions groups](#)
- [Assigning user interface forms to object definitions](#)
- [Assigning user interface forms to menu items](#) on page 146
- [Copying user interface forms](#) on page 142
- [Displaying custom columns and tables on main data forms](#) on page 146
- [Forms for custom extensions](#) on page 147

Assigning user interface forms to applications

To display a user interface form in an application's user interface, you must first assign the form to the application.

To assign a user interface form to an application

1. In the Designer, select the **User interface > Forms > User interface forms** category.
2. Select the **Edit form** task.
3. In the Form Editor, select the user interface form.

4. In the edit view, select the **Application** view and then the application.
5. Select the **Database > Save to database** and click **Save**.

Related topics

- [Creating user interface forms](#) on page 142
- [Assigning user interface forms to permissions groups](#) on page 144
- [Applications for configuring the user interface](#) on page 195

Assigning user interface forms to permissions groups

All user interface forms to be displayed in an application user interface must be assigned to a permissions group. Assign the user interface forms to permissions groups for non role-based and role-based login. The interface forms are available to system users depending on their permissions group memberships. For more information about permissions groups, see the *One Identity Manager Authorization and Authentication Guide*.

To assign a user interface form to permissions groups

1. In the Designer, select the **User interface > Forms > User interface forms** category.
2. Select the **Edit form** task.
3. In the Form Editor, select the user interface form.
4. In the edit view, select the **Permissions group** view and select the permissions groups.
5. Select the **Database > Save to database** and click **Save**.

Related topics

- [Creating user interface forms](#) on page 142
- [Assigning user interface forms to applications](#) on page 143

Assigning user interface forms to object definitions

If you want to display a user interface form in the user interface depending on the particular object selected, you must assign the form to the valid object definition.

To assign a user interface form to an object definition

1. In the Designer, select the **User interface > Forms > User interface forms** category.
2. Select the **Edit form** task.
3. In the Form Editor, select the user interface form.
4. In the edit view, select the **Object assignment** view and select the object definition.
5. Select the **Database > Save to database** and click **Save**.

Related topics

- [Creating user interface forms](#) on page 142
- [Object definitions for the user interface](#) on page 108
- [Effects of object definitions when displaying interface forms](#) on page 145

Effects of object definitions when displaying interface forms

Interface forms that need to be valid for all entries in a database table are allocated a general object definition. Other limited object definitions can have more interface forms. If an entry is selected in the user interface, the currently valid object definitions are used to gather all the interface forms and display them in the user interface in their sort order in the task view and in the context menu.

Example:

The following object definitions with interface forms are set up for the ESet table.

Table 40: Example: Interface forms for object definitions

Object definition	Assigned Interface Form
System roles (ESet)	System role overview
System roles for the IT Shop (ESet_ITShop)	Add to IT Shop

The following interface forms are displayed when an employee object fulfills the **System roles** object definition:

- System role overview

The following interface forms are displayed when an employee object fulfills the **System roles for IT Shop** object definition:

- System role overview
- Add to IT Shop

Related topics

- [Assigning user interface forms to object definitions](#) on page 144
- [Object definitions for the user interface](#) on page 108

Assigning user interface forms to menu items

You can assign user interface forms for individual menu items. The user interface form is displayed when a user chooses the menu item in the navigation view or an entry in the result list. It is displayed irrespective of the user's permission groups.

To assign a user interface form to a menu item

1. In the Designer, select the **User interface > Forms > User interface forms** category.
2. Select the **Edit form** task.
3. In the Form Editor, select the user interface form.
4. In the edit view, select the **Menu assignment** view and select the menu item.
5. (Optional) Enable the **Show in navigation** option to open the form from the navigation view.
6. Select the **Database > Save to database** and click **Save**.

Related topics

- [Creating user interface forms](#) on page 142

Displaying custom columns and tables on main data forms

Displaying columns in custom tables

To display custom database table in the administration tool user interfaces and edit the main data:

- Create a user interface form using the form definition **VI_Generic_MasterData**. This form definition allocates the control element for editing main data in the user interface.
- In the Designer, specify the order for displaying input fields in the **Sort order** property (DialogColumn.SortOrder). Columns with a sort order of less than one are not displayed.
- Achieve a better overview of the input fields by grouping database columns. In the Designer, customize the **Group** property (DialogColumn.ColumnGroup) in the column definition. Each group has its own tab. The name of the tag corresponds to the group.
- Columns whose data contents can be multiline are displayed in a multiline field on the generic form. Label these columns as **multi-line**.

Displaying custom columns in predefined tables

Separate tabs can be shown for custom column extensions to default tables on the predefined main data forms.

The preceding features apply if the predefined main data form uses the **VI_Generic_MasterData** form definition. Otherwise the following prerequisites are required for using this functionality:

- Main data form already has tabs. Simple main data forms without tabs are not extended.
- To change the sort order in which the input fields on the form are displayed, select the **Sort order** property (DialogColumn.SortOrder) of the database columns. Columns with a sort order of less than one are not displayed.
- Database columns are grouped. In the Designer, customize the **Group** property (DialogColumn.ColumnGroup) in the column definition. Each group has its own tab. The name of the tag corresponds to the group. If no group is specified, a tab with the name **Custom** is displayed.

NOTE: Other special features apply to displaying custom schema extensions on the UNSAccountB, UNSContainerB, UNSGroupB, UNSItemB, and UNSRootB tables. For more information, see the *One Identity Manager Administration Guide for Connecting to Custom Target Systems*.

Related topics

- [Forms for custom extensions](#) on page 147
- [Editing user interface forms](#) on page 140
- [Column definition properties](#) on page 91

Forms for custom extensions

One Identity Manager provides an array of form templates and definitions in the default installation. These can be used for easily creating your own forms.

Another way to create custom forms is to make custom form archives available. Normally, default forms in One Identity Manager are replaced with self developed forms.

Table 41: Form templates and definitions for custom extensions

Form template	Form definition	Usage
FrmCommonChildRelationGrid	VI_Common_ChildRelation_Grid	For editing many-to-many relations with extended properties in the form of a table.
FrmCommonOneChildAndMemberRelation FrmCommonOneMemberAndChildRelation	A custom form must be created on which the data to be configured is displayed.	Assigns many-to-many relations and object relations (parent/child relations) on one form. Two tabs for displaying the data are shown on the form.
FrmCommonOneChildRelation	A custom form must be created on which the data to be configured is displayed.	Mapping object relations (Parent/ChildRelation). If several additional object relations are mapped on a form, the FrmCommonTwoChildRelation and FrmCommonThreeChildRelation form templates can be used as alternatives. One tab is shown per object relation.
FrmCommonOneDynamicRelation	A custom form must be created on which the data to be configured is displayed.	Displays dynamic many-to-many relations whose assigned object is referenced through a dynamic Permitted dynamic objects are found in the DialogValidDynamicRef table. A menu is provided for choosing the object type.
FrmCommonOneGenericRelation	A custom form must be created on which the data to be configured is displayed.	Displaying dynamic many-to-many relations. <ul style="list-style-type: none"> Base object can be referenced through a dynamic key. - OR - Assigned object is referenced through a dynamic key. In this case, the MembersTableName property must be defined in

Form template	Form definition	Usage
		the form configuration.
FrmCommonOneMemberRelation	A custom form must be created on which the data to be configured is displayed.	Assigning many-to-many relations. If several additional many-to-many relations are mapped on a form, the FrmCommonTwoMemberRelation , FrmCommonFourMemberRelation , and FrmCommonFiveMemberRelation form templates can be used as alternatives. On tab is shown per many-to-many table.
FrmElementNavigation	VI_ElementNavigation	For displaying the overview form.
frmGeneric	VI_Generic_MasterData	For editing object main data.
ReportForm	VI_Report	For displaying reports.
WizardForm	VI_Wizard	For including wizards. The forms are displayed in a modal dialog window.

Related topics

- [Hierarchical display of data on assignment forms](#) on page 149
- [Configuration data for displaying many-to-many and object relations on forms](#) on page 156
- [Replacing default forms with custom forms](#) on page 150

Hierarchical display of data on assignment forms

Forms of the **MemberRelation** type are used to display the data in an assignment list (many-to-many relations). Enter the hierarchy path in the table definition to display the table hierarchically. Enter the foreign key column that the hierarchy should be based on.

Example:

An Active Directory user account (ADSAccount table) is typically displayed on an assignment form below its Active Directory container (UID_ADSTContainer). The Active Directory container (ADSTContainer table) is, on the other hand, displayed underneath its Active Directory domain (column UID_ADSDomain). The path for the hierarchy structure is entered as follows:

Table 42: Example of a hierarchy path

Table	Hierarchy path
ADSAccount	UID_ADSTContainer, UID_ADSDomain
ADSTContainer	UID_ADSDomain

You can specify an alternative list for objects that do not have values in all foreign key columns after a pipe (|).

Example:

(UID_ADSTContainer,UID_ADSDomain|UID_ADSDomain)

Related topics

- [Table definition properties](#) on page 68

Replacing default forms with custom forms

Self developed form templates can be provided for custom forms in a form archive (*CustomForms*.vif). You need to add the form template, form definition and interface form with help of the Form Editor if you want to display your custom forms in the user interface.

A wizard is available to swap a default form with all its dependencies for a custom form. The wizard creates the interface form with the form definition and the form template. The properties of the new form are taken from the form it is replacing. The necessary assignments (object definition, menu item, permissions group, and application) are created for the new form and the replaced form is disabled.

To replace custom forms with all dependencies

1. In the Designer, select the **User interface > Forms > User interface forms** category.
2. Select hierarchical representation of the form overview. Set the **Options > Tree/list view** menu option to do this.

3. At the highest hierarchy level of the form overview, select the form template of the form to be replaced and start the wizard using the **Replace by** context menu.
4. On the start page of the wizard, click **Next**.
5. On the **Select file and form** page, enter the following information.
 - **Form archive file:** Select the form archive file (*.CustomForms.*.vif).
 - **Form template:** Select the form template for the new user interface.
6. On the **Define form structure** page, check the names of the form definitions and user interface forms. The names of the form definitions and user interface forms should all begin with **CCC**. Use **F2** to change the names and select **Enter** to save the change.
7. On the **Select permissions group** page, select the permissions group to which the new user interface form should be assigned. Use the **+** button to create a new permissions group.
8. The last page of the wizard summarizes the settings for replacing a form. To replace the form, click **Finish**.



The wizard is closed after replacement is complete. The new form is displayed in the Form Editor form overview after the wizard is complete and you can continue editing it. The replaced form is disabled and can therefore no longer be available in the user interface.





Related topics

- [Forms for custom extensions](#) on page 147

User interface form properties

Table 43: User interface form properties

Property	Meaning
Form name	<p>The form name is used to quickly select interface forms, for example, in the Designer.</p> <p>TIP: The form name is displayed in the administration tool as extra navigation information.</p>
Form definition	<p>Form definition linked to the user interface form.</p> <p>NOTE: Use  next to the input field to link in a new form definition for the interface form.</p>
Caption	<p>Caption shown on the user interface form. The caption is used to represent the user interface form in the task view and in the form context menu of the user interface. Translate the given text using the  button.</p>
Online help link	<p>The form's help key for navigating to the relevant chapter in the online</p>

Property	Meaning
	help.
Description	Detailed description of the user interface form. TIP: The description is shown as a tooltip in the task view.
Icon	Icon marks the user interface form in the user interface.
Sort order	The sort order determines the position of the interface form in the task view and in the form's context menu in the administration tools. NOTE: When you enter objects in the Manager, the user interface form of the Edit form type is always displayed with the lowest sort order.
Preprocessor condition	User interface forms can be given a preprocessor condition. This means that an interface form is only available when the preprocessor condition is fulfilled. NOTE: In the Designer, you can find an overview of existing preprocessor dependencies in the One Identity Manager Schema > Preprocessor dependencies category.
Control deactivation	Specifies which buttons in the toolbar are to be disabled in the front-ends Permitted values are: <ul style="list-style-type: none"> • Insert object: The  button is disabled. • Update object: The  button is disabled. • Delete Object: The  button is disabled. • Save to database: The  button is disabled
Deactivated	Use this option to label interface forms that should not be shown in the user interface. NOTE: This change is also permitted for user interface forms in the default user interface and is not overwritten on schema installation.
Disabled by preprocessor	If an object definition is excluded through a preprocessor condition, this option is set by the Database Compiler.
Show modal	Specifies whether the form is displayed in a separate dialog box. Used by wizards for entering data.
Open on new tab	The form is opened on a new tab.
Configuration	The configuration is used to limit the tables and columns on display. Templates for the configuration data definition are found in the pop-up list XML templates . In the Properties section, you can transfer special properties of the

Property	Meaning
	form that were implemented during form development. For example, the report name and special report parameters can be passed to the report interface form by using this section.
Insert values	Insert values are only of relevance to interface forms of the Edit form type. With them you can specify the default values for the columns that are assigned when a new object is added. The input is in VB.Net syntax.

Related topics

- [Form definitions and form templates](#) on page 153
- [Hierarchical display of data on assignment forms](#) on page 149
- [Defining insert values](#) on page 132
- [Conditional compilation using preprocessor conditions](#) on page 335

Form definitions and form templates

Form definitions and form templates make up the basis of interface form design. Form definitions contain information about the data that will appear in the forms, for example, tables, and columns as well as titles for form tabs and root nodes in hierarchically ordered elements (ChildRelationControl, membership tree) for the form templates defined in the form archives (Forms.*.vif).

Detailed information about this topic

- [Form templates](#) on page 153
- [Form definitions](#) on page 155
- [Configuration data for displaying many-to-many and object relations on forms](#) on page 156

Form templates

You can find all the form templates in the Designer in the **User interface > Forms > Form templates** category. It is not usually necessary to define your own form templates.

To display a form template for a user interface form

1. In the Designer, select the **User interface > Forms > User interface forms** category.
2. Select the user interface form in the Form Editor.
3. In the edit view, in **Properties**, select the **Form template** tab.

Table 44: Form template properties

Property	Meaning
Form source type	Source of the form template. Permitted values are: <ul style="list-style-type: none">• Form: For displaying a form from a form archive.• Assembly: For displaying controls. It is not necessary to build a form, because the control is displayed directly as form.
Assembly name	Name of the assembly file.
Class	Full type name of the control.
Form template name	The form template name is necessary for loading the form template from the form archive. TIP: The form template name is shown in the administration tools as additional navigation information.
Form archive	Name of the form archive (Forms.*.vif), containing the form template.
Description	Detailed description of the form template.
Alternative form template	It might be necessary to use different form templates of display the interface form, for example, to show an the One Identity Manager web interface or in an administration tool. The form templates can be linked in order to avoid adding a form definition and an interface form for each form template. For this, you need to assign an alternative form template to the form template. This alternative form template is used when the conditions for displaying the original template are not fulfilled. The form template referenced is determined in order to display the interface form. The form template given in the form definition is checked for existence in the form archive and to see if it labeled for the correct display purposes. If these conditions are not fulfilled then the alternative form template is tested for suitability. The form template that fulfills the conditions is used for the user interface display.
Form type	Type of form.
Enabled for	Specifies the intended use of the form template. Permitted values are: <ul style="list-style-type: none">• Visible in graphical interface• Visible in web application• TimeTrace supported• Multiobject editing possible• Deferred operation possible• Application server not supported

Table 45: Form types and their usage

Form type	Usage
Info (I)	Forms of the Info type are only used to display information. Changes to data on these forms cannot be saved. These forms automatically omitted by the automatic form selection in quick edit mode.
Edit (E)	Forms of the Edit type are used to edit data. This is the first form to be loaded by the automatic form selection in quick edit mode.
Grid (G)	Forms of the Grid type are used to display data in tabular form.
MemberRelation (M))	Forms of the MemberRelation type are used to display the data in an assignment list (many-to-many relations).
Report (R)	Forms of the Report type are used to display data in a report form.
Virtual (V)	Forms of the Virtual type are not available in the forms menu. This form type is used to show editors in the Designer.
Wizard (W)	Forms of the Wizard type are used to enter data by means of a wizard. The forms are displayed in a modal dialog window.

Related topics

- [Form definitions](#) on page 155
- [Hierarchical display of data on assignment forms](#) on page 149

Form definitions


You can find form definitions in the Designer in the **User interface > Forms > Form definitions** category. It is not normally necessary to define your own form definitions.

To display a form template for a user interface form

1. In the Designer, select the **User interface > Forms > User interface forms** category.
2. In the Form Editor, select the user interface form.
3. In the edit view in **Properties**, select the **Form definition** tab.

Table 46: Form definition properties

Property	Meaning
Form definition name	Name of the form definition. This name is used for displaying the form definition in the Designer.
Form template	Name of the form template to load from the form archive. A form template can be used by several form definitions, such as the form templates for displaying membership trees or the form template for

Property	Meaning
	displaying reports. Use the  button next to the input field to integrate a new form template in the form definition.
Base form for form sequence	By entering a form definition as a base for a sequence of forms, you can create a group of form definitions for one object definition. All form sequence form definitions contain the same base form. The definition of the interface form can only be made for this base form. When the interface form is loaded in the display, the referenced form definitions for all other form definitions in the form sequence are also loaded. You can navigate arbitrarily within the form sequence without leaving the scope of the interface form.
Description	Detailed description of the form.
Configuration	The configuration data is used to describe the form properties. The definition of the form properties is written in XML notation.
Required tables	A form definition can be assigned additional tables that are used to display data. NOTE: If one of the given tables is disabled by a preprocessor condition then the form definition is also considered to be disabled and the corresponding interface form is not shown in the user interface.

Related topics

- [Form templates](#) on page 153
- [Configuration data for displaying many-to-many and object relations on forms](#) on page 156

Configuration data for displaying many-to-many and object relations on forms

Form properties are specified by the form definition configuration data. The definition of the form properties is written in XML notation.

Example: Configuration data structure

```
<DialogFormDefinition FormatVersion="1.0">
  <ComponentDefinitions>
    <ComponentDefinition Name="TabPage1" Type="VI.Components.TabPage">
      <Properties>
```

```

        <Property name="Caption" value="Department"/>
        <Property name="CaptionTranslationSource"
value="DatabaseSchema" />
    </Properties>
</ComponentDefinition>
...
<ComponentDefinition name="MemberRelation1"
Type="VI.Components.MemberRelation">
    <Properties>
        <Property name="MNBaseColumnName" value="UID_ADSTGroup"
IsMandatory="True" />
        <Property name="MNTableName"
value="DepartmentHasADSTGroup" IsMandatory="True" />
        ...
    </Properties>
</ComponentDefinition>
...
</ComponentDefinitions>
</DialogFormDefinition>

```

Displaying relations

Table 47: Properties of relation definitions

Component	Property	Meaning
All		Valid for all maps.
	WhereClause	<p>Limited condition for applying to the displayed objects (member, child).</p> <p>The expression %column% can be used in the WhereClause to reference values of the base object.</p> <p>\$ expressions are permitted to reach other values from the base object, for example \$FK(UID_ADSTGroup).UID_ADSTDomain\$.</p> <p>Example:</p> <pre><Property name="WhereClause" value="IsITShopOnly=0" /></pre>

Component	Property	Meaning
	EditWhereClause	<p>Edit condition. The elements that match the condition can be edited. All other elements are also displayed but cannot be edited.</p> <p>Example:</p> <pre><Property name="EditWhereClause" value="XMarkedForDeletion=0" /></pre>
MemberRelation 1- MemberRelationN		<p>Displaying M:N relations</p> <p>Example:</p> <pre><ComponentDefinition name="MemberRelation1" Type="VI.Components.MemberRelation"></pre>
	MNTableName	<p>M:N table.</p> <p>Example:</p> <pre><Property name="MNTableName" value="OrgHasADSGroup" /></pre>
	MNBaseColumnName	<p>Column of the M:N table that points to the base object.</p> <p>Example:</p> <pre><Property name="MNBaseColumnName" value="UID_ADSGroup" /></pre>
	RootFilterTableName	<p>Table for filtering assignable elements from users. If defined, the control element shows a menu with objects from this table. If, for example,</p> <p>Example:</p> <pre><Property name="RootFilterTableName" Type="String" value="OrgRoot" /></pre>
	RootFilterWhereClause	<p>Condition for filtering elements of the RootFilterTableName in the menu.</p> <p>Example:</p> <pre><Property name="RootFilterWhereClause" Type="String" value="UID_OrgRoot"</pre>

Component	Property	Meaning
		<pre>in (select UID_OrgRoot from Org) and exists (select 1 from OrgRootAssign where IsDirectAssignmentAllowed = 1 and UID_OrgRoot=OrgRoot.UID_OrgRoot and UID_BaseTreeAssign='ADS- AsgnBT-ADSGroup')"</pre>
	RootFilterMemberWhereClause	<p>Condition formatted after selecting a base object and attached to the WhereClause. The condition must always contains a column relation to the base object.</p> <p>Example:</p> <pre><Property name="RootFilterMemberWhereClause" Type="String" value="UID_ OrgRoot=N'%UID_OrgRoot%' "</pre>
	ShowExtendedProperties	<p>Specifies whether many-to-many tables with additional columns are offered an additional Extended properties context menu item on the assignment form. You use the context menu item to navigate to the detailed form where you can edit the extended properties.</p> <p>Example:</p> <pre><Property name="ShowExtendedProperties" Value="True" /></pre>
ChildRelation1-ChildRelationN		<p>Displaying parent-child relations.</p> <p>Example:</p> <pre><ComponentDefinition name="ChildRelation1" type="VI.Components.MemberRelation"></pre>
	CRTTableName	<p>Table in which child objects are mapped.</p> <p>Example:</p> <pre><Property name="CRTTableName" value="ADSAccount" /></pre>

Component	Property	Meaning
	CRColumnName	Child table foreign key that points to the base object. Example: <pre><Property name="CRColumnName" value="UID_Person" /></pre>
	ShowForeign	Specifies whether foreign assignments (object assigned to another object) can be displayed. Example: <pre><Property name="ShowForeign" value="True" /></pre>
GenericRelation1- GenericN		Displaying dynamic many-to-many relations. Example: <pre><ComponentDefinition Name="GenericRelation1" Type="VI.Components.MemberRelation"></pre>
	MNTableName	M:N table. Example <pre><Property name="MNTableName" value="ADSPolicyAppliesTo"/></pre>
	MNBaseColumnName	Column of the M:N table that points to the base object. Example: <pre><Property name="MNBaseColumnName" value="ObjectKeyAppliesTo" /></pre>
	MNMembersColumnName	Column of the M:N table that points to the members. Example: <pre><Property name="MNMembersColumnName" value="UID_ADSPolicy" /></pre>
	MembersTableName	Tables whose objects must be assigned. Example: <pre><Property name="MembersTableName" value="ADSPolicy"/></pre>

Using tabs

Use the components `TabPage` to display tabs for the mapped relations. Usually tabs are used for forms that map multiple relations, such as **FrmCommonTwoMemberRelation** or **FrmCommonTwoChildRelation**. `TabPage1` maps the tab for `Relation1`, `TabPage2` maps the tab for `Relation2`.

Table 48: Properties of tab definitions

Component	Property	Meaning
TabPage1- TabPageN		Displays 1-n tab for each relation to be shown. Example: <pre><ComponentDefinition Name="TabPage1" Type="VI.Components.TabPage"></pre>
	Caption	Tab captions. Table names or any string can be used as captions. Example: <pre><Property name="Caption" value="Department"/></pre>
	CaptionTranslationSource	Source for translating the tab names. <code>value="DatabaseSchema"</code> finds the table captions translation from the One Identity Manager schema table given under <code>Caption</code> . <code>value="TranslationAddOnSource"</code> finds the translation from the text store. Example: <pre><Properties> <Property name="Caption" value="Department"/> <Property name="CaptionTranslationSource" value="DatabaseSchema" /> </Properties> <Properties> <Property name="Caption" value="is member of"/> <Property name="CaptionTranslationSource" value="TranslationAddOnSource" /> </Properties></pre>

Related topics

- [Forms for custom extensions](#) on page 147
- [Form definitions](#) on page 155

Working with overview forms

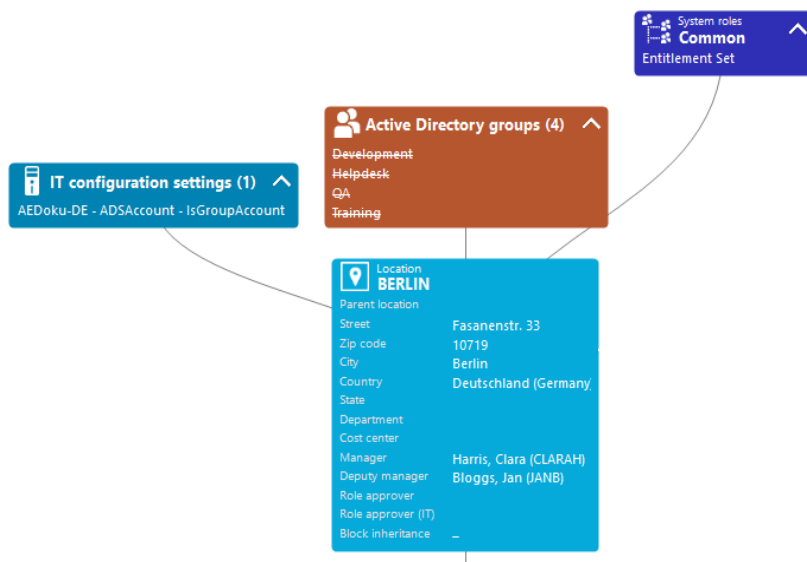
There is a special control element for displaying the overview form in the user interface. The information to be displayed on the overview form is configured with menu items. The menu items are represented as form elements that are linked to each other on the overview form. A hierarchical structure of menu items is also included in the interface configuration.

The basis is formed by a menu item with the **Main form element** item type. This menu item specifies the main element on the overview form. An interface form that links to this menu item has to be configured in order for it to be displayed in the application. The main form element is always displayed in the middle of the overview form.

The other menu item such as fixed, data-dependent, link, or statistic menu items are configured under the menu item for the main form element. These menu items are grouped around the main form element on the overview form as additional form elements.

The color and positioning of the form elements on the overview as well as the properties that are shown, are specified by layout information for the menu items.

Figure 15: Example of elements in an overview form



The display text of the menu item, the display text for the objects to be shown and the menu item icon are displayed in the header of a form element. Other data represents the object properties and values. There is a tooltip for each property showing a description for use. Some form element entries are highlighted in color when you click on them with the mouse. You can jump to the referenced object by clicking on the entry with the mouse.

If the form element is used for mapping lists, the items are displayed with their names. The number of items is shown in the form element header. There is also an icon in the form element header for showing and hiding the items. There is no tooltip for list items.

Table 49: Form element icon

Icon	Meaning
▼	Show list items.
▲	Hide list items.

NOTE: Objects marker for deletion are ~~struck through~~ on the overview form.

Detailed information about this topic

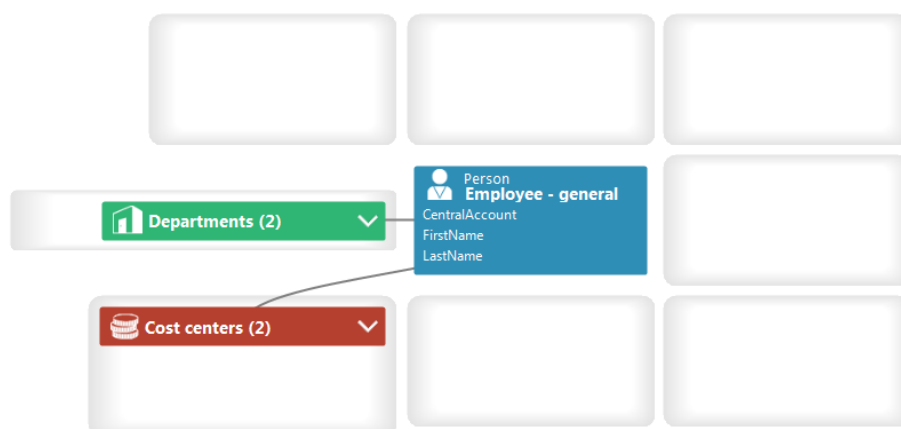
- [Creating overview forms](#) on page 163
- [Adding more form elements to overview forms](#) on page 165
- [Special features of editing overview forms](#) on page 166
- [Previewing an overview form during editing](#) on page 169
- [Customizing the form elements layout](#) on page 167
- [Disabling overview forms and form elements](#) on page 169
- [Deleting form elements](#) on page 170
- [Deleting overview forms](#) on page 170

Creating overview forms

The Overview Form Editor helps you to create overview forms. The Overview Form Editor performs the following steps to create the overview form.

- Creating a menu item with the **Main form element** item type.
- Creates other menu items under the main form element.
- Creates a user interface form for the main form element.


Figure 16: Design view in the Overview Form Editor



To create a new overview form

1. In the Designer, select the **User interface > Forms > Overview forms** category.
2. Select the **Create new overview form** task.
3. Enter the basic properties for the overview form.

Table 50: Basic data for an overview form

Property	Meaning
Menu item	Name of the menu item. Assign descriptive names here if possible. These are then propagated in the child structures.
Caption	Caption shown on the user interface form. The caption is used to represent the user interface form in the task view and in the form context menu of the user interface. Translate the given text using the  button.
Object	Object definition for which the form should be displayed.
Parent menu item	Parent menu item for grouping together the overview forms; usually a menu category.
Product assignment	Application in which the form will be shown.
Group assignment	Permissions group for which the form will be shown.
Display columns	Columns to be displayed on the main form element. TIP: Use the Show column captions link to switch between column captions and technical names.

4. To create the overview form, click **OK**.

This displays the overview form design in the Overview Form Editor. You can continue editing the overview form.

Related topics

- [Adding more form elements to overview forms](#) on page 165
- [Special features of editing overview forms](#) on page 166
- [Previewing an overview form during editing](#) on page 169
- [Customizing the form elements layout](#) on page 167
- [Disabling overview forms and form elements](#) on page 169
- [User interface navigation](#) on page 112

Adding more form elements to overview forms

To add more form elements to the overview form

1. In the Designer, select the **User interface > Forms > Overview forms** category.
2. Select the overview form, and then the **Edit overview form** task.
3. Select the view **Object relations**.

All the object foreign key relations (FK), object child relations (CR), and object member relations (M:N) are displayed.
4. Select the object relation that you want to display and drag and drop it on an element in the element area of the design view.
5. Select the type of menu item you want to create. You have the following options:
 - **Create list element:** A fixed menu item with predefined list properties is created.
 - **Create data element:** A data-dependent menu item is created.
 - **Create list element reference:** A menu item is created with the **Link** item type and a reference to a menu item for display as a list.
 - **Create reference to data element:** A menu item is created with the **Link** item type and a reference to a data-dependent menu item.

The menu item's main data is automatically generated by the Overview Form Editor. The form element is display in the Overview Form Editor's design view.

The following special features apply when you create form elements using the **Create list element reference** and **Create reference to data element** context menus.

- The reference entries under the InfoSheets.QIM.Links menu item are used.
- If the required reference entries are not yet available, new reference entries are created with the names InfoSheet.List.<table> or InfoSheet.Node.<table>.
- In the reference entry condition, a %<Table>WhereClause% variable is used.

- A variable with the **Text** variable type is used on the form element. A condition formulated as a WHERE clause is assigned to these variables on the form element. You can further modify this condition as required. In the Overview Form Editor's edit view, edit the variable in the **Variable definitions** view.

TIP: Use the **Create element** context menu to create more menu items, links, or statistics as form elements in the Overview Form Editor's design view. In this case, enter the main data of the menu item, link, or statistics manually.

Related topics

- [Creating overview forms](#) on page 163
- [Special features of editing overview forms](#) on page 166
- [Previewing an overview form during editing](#) on page 169
- [User interface navigation](#) on page 112
- [Using links in the navigation](#) on page 133
- [Using variables in the navigation](#) on page 135
- [Including statistics in the user interface](#) on page 176

Special features of editing overview forms

Special features of a user interface form for the main form element of an overview form

- The user interface form is created using the **VI_ElementNavigation** form definition. This form definition provides the control element for displaying the overview form in the user interface.
- Enter the name of the main form element in the user interface form's configuration data in the Properties section.

Example:

```
<DialogSheetDefinition FormatVersion="1.0">
  <Properties>
    <Property Name="OverviewNode">VI_Person_Person_
      Overview</Property>
  </Properties>
</DialogSheetDefinition>
```

Special features of mapping lists on an overview form

If a form element is used for mapping lists, the items are displayed with their names. You can jump to the referenced object by clicking on the entry with the mouse.

To prevent navigation to the referenced object, set the value of the configuration switch on the menu item to **Ignore user interface forms in result list**. This is useful if, for example, forms are not defined for some objects in the result list. Otherwise, an empty form is displayed.

Related topics

- [Creating overview forms](#) on page 163
- [Adding more form elements to overview forms](#) on page 165
- [Customizing the form elements layout](#) on page 167
- [User interface form properties](#) on page 151
- [Forms for custom extensions](#) on page 147

Customizing the form elements layout

The color and positioning of the form elements on the overview as well as the properties that are shown, are specified by layout information for the menu items. You can modify these properties for predefined overview forms as well.

To customize the form element's layout information

1. In the Designer, select the **User interface > Forms > Overview forms** category.
2. Select the overview form and open it in the Overview Form Editor.
3. Select the form element in the design view.
4. Select the tab **Layout** in the **Properties** view and change the properties.

Table 51: Form elements layout

Property	Meaning
Alignment	Positioning of the form elements on the overview form. You cannot align the main form element. The main form element is always displayed in the middle of the overview form. All child menu items are positioned relevant to the main form element.
Background color	Color for displaying the form element on the overview form The background color of the main form element cannot be configured. When a link is set up, it is given the background color of the referenced menu item.
max. similar	If a menu item defines a list of items, each item in the menu item's

Property	Meaning
elements count	<p>result list is displayed in a separate form element.</p> <p>Define up to how many items should be displayed in separate form elements. If the number is exceeded the items are grouped into a list and displayed in one form element. In this case, any given columns are not displayed.</p> <p>The items are shown with their display template. The number of items is shown in the form element header. There is also an icon in the form element header for showing and hiding the items.</p> <p>TIP: If you want a list in the display template with no more than two column names, you can use a table to create a two-column display.</p>
Display columns	<p>Specifies which columns from the valid object definition are to be displayed in the form element. The columns for the main form element refer to the object definitions of the associated overview form. All other form elements get their object definitions from the menu items. When a link is configured, the selected columns of the referenced menu item are initially copy to the link. The order of displaying the properties in a form element corresponds to the column sort order defined in the menu item.</p> <p>TIP: If you want to display a line in a form element to visibly separate the information, enter a minus sign (-) in the column to be displayed.</p> <p>You can use scripts in the column definitions to affect the value displayed in the column. The column definition is activated when you click and hold on the column name. Extend the column definition as follows:</p> <pre>Column[S(script name)]</pre> <p>If columns contain a web address and you want to navigate to this web address, use the URI formatting type for the column definition. This display a link on the overview form.</p>

Designing the form element header

The menu item display text, display text for the objects to be shown and the menu item icon are displayed in the header of a form element.

TIP: You can open an interface form by clicking the caption in the form element header.

- To do this, assign a fixed menu item to the interface form that is allocated below the main form element. The interface form, however, must refer to the main form element, for example, a form for assigning this object.
- Use the option **Navigation view** in the form assignment view to access forms in the user interface.

Related topics

- [Using predefined formatting types](#) on page 81
- [Assigning user interface forms to menu items](#) on page 146

Previewing an overview form during editing

To create a preview of an overview form

1. In the Designer, select the **User interface > Forms > Overview forms** category.
2. Select the overview form and open it in the Overview Form Editor.
3. Select the main form element's table in **Table** in the Overview Form Editor's toolbar and select a fixed object to use for the **Object** preview from.

| **NOTE:** In the **Object** menu, select the **No object** item to end the preview.

Disabling overview forms and form elements

You can disable single predefined overview forms or single form elements of an overview form if necessary. This prevents them being displayed in the user interface. They remain disabled even after schema installation.

To disable an overview form

1. In the Designer, select the **User interface > Forms > Overview forms** category.
2. Select the overview form and start the Form Editor with the **Edit interface form** task.
3. Set the **Disabled** option.
4. Select the **Database > Save to database** and click **Save**.

To disable a form element on an overview form

1. In the Designer, select the **User interface > Forms > Overview forms** category.
2. Select the overview form and start the Overview Form Editor with the **Edit overview form <form name>** task.
3. Select the form element in the design view.
4. Set the option **Disabled**.
5. Select the **Database > Save to database** and click **Save**.

You can also disable overview forms or single form elements using preprocessor conditions.

| **NOTE:** In the Designer, you can find an overview of existing preprocessor dependencies in the **One Identity Manager Schema > Preprocessor dependencies** category.

Related topics

- [Deleting form elements](#) on page 170
- [Deleting overview forms](#) on page 170

Deleting form elements

To delete a form element on an overview form

1. In the Designer, select the **User interface > Forms > Overview forms** category.
2. Select the overview form then select the **Edit overview form** task.
3. In the Overview Form Editor's design view, select the form element, and choose the **Delete element** context menu item.
4. Select the **Database > Save to database** and click **Save**.

Related topics

- [Deleting overview forms](#) on page 170
- [Disabling overview forms and form elements](#) on page 169

Deleting overview forms

To delete an overview form, delete the user interface form, the menu item for the main form element and the child menu items for the other form elements.

To delete an overview form

1. In the Designer, select the **User interface > Forms > Overview forms** category.
2. Select the overview form, and then **Edit interface form**.
3. In the context menu, select **Delete**.
4. In the Designer, select **User interface > User interface navigation** category.
The menu items are loaded and displayed in the User Interface Editor for editing.
5. In the navigation overview, select the menu item that was linked to the overview form.
6. To delete the menu item and its child menu item, select the **Delete** context menu item.
7. Select the **Database > Save to database** and click **Save**.

Related topics

- [Deleting form elements](#) on page 170
- [Disabling overview forms and form elements](#) on page 169

Statistics in One Identity Manager

The One Identity Manager info system provides you with a quick overview of the system situation. Statistics are recalculated at regular intervals and visualized in the user interface in various charts. Statistic definitions are already supplied with One Identity Manager. You can create more statistic data in the Designer if required.

The following steps are necessary to make statistics available:

1. Create statistic definitions
2. Link statistics into the user interface

Detailed information about this topic

- [Creating and editing statistic definitions](#) on page 171
- [Disabling statistics definition](#) on page 175
- [Including statistics in the user interface](#) on page 176
- [Diagram types for visualizing statistics](#) on page 179
- [Examples of statistic definitions](#) on page 183

Creating and editing statistic definitions

The basis for the info system is the definition of statistics. Predefined configurations are maintained by the schema installation and cannot be edited apart from a few properties. The default configuration is moved to a configuration buffer during handling. You can retrieve changes from the configuration buffer and restore the default configuration in this way.

To create or edit a statistic definition

1. In the Designer, select the **User interface > Statistic definitions** category.
2. Select a statistic definition and then the **Change main data** task.

- OR -

To create a new statistic definition, select **Object > New**.

3. Enter the general properties on the **General** tab.
4. Enter the inventory query on the **Queries** tab.
5. Check the queries and statistic definition for errors.

- Use the **Check query** button to test each query.

The SQL query and its result are tested for validity. This checks the number of columns, column relations, and data types.

- Use **Check** button to check the entire statistic definition.

To run the test, the statistic is saved in the database and the calculation is simulated. After simulation, the simulated test calculation is removed from the database.



6. Select the **Database > Save to database** and click **Save**.


Detailed information about this topic

- [General properties statistic definitions](#) on page 172
- [Querying statistic measurements](#) on page 174
- [Examples of statistic definitions](#) on page 183

General properties statistic definitions

Table 52: Properties of a statistic definition

Property	Meaning
Statistics	Name of the statistic
Display name	<p>This display name is used to show the statistic definition in the settings for the info system in the administration tools. The display name forms the title of a statistic. Translate the given text using the  button.</p> <p>NOTE: If a caption is entered in the menu item, it overwrites the statistic definition display name.</p>
Description	<p>Description of the statistic definition. The statistic definition description is shown in the info system settings in the administration tools. Translate the given text using the  button.</p>
Calculation schedule	<p>Select the schedule for calculating the statistic information. The Calculate statistics, Calculate weekly statistics, and Calculate monthly statistics on the 1st schedules are provided.</p> <p>NOTE: In the Designer, enable the schedules for calculating statistics in the Basic data > General > Schedules category. For more information about editing schedules, see the <i>One Identity Manager Operational Guide</i>.</p>
Aggregate function	<p>Use the aggregate function if the measurements query returns several values but there should only be one value displayed in the statistics.</p> <p>Example:</p> <p>Determines the number of employees for which a department head is responsible. Use the SUM aggregate function to display a statistic with the total number of employees in all departments for which one person is responsible. Do not use an aggregate function to display statistics by department.</p>

Property	Meaning
Base aggregate function	<p>Use the base aggregate function if a unique base value cannot be attained from the measurements query.</p> <p>NOTE:</p> <ul style="list-style-type: none"> Aggregate and base aggregate functions are only evaluated if the formulated measurement value query is limited by a condition on the logged in user. Aggregate and base aggregate functions are only taken into account for statistics that are displayed in the Web Portal.
Threshold green	Threshold factor in the value range from 0 to 1 . This threshold factor is used to determine the percentage of the base measurement that reflects a correct status.
Threshold red	Threshold factor in the value range from 0 to 1 . This threshold factor is used to determine the percentage of the base measurement that reflects an acceptable status.
Unit of measure	Unit for measured values. The unit of measure is displayed in the info system statistics. Translate the given text using the  button.
Time scale	Enter the display accuracy of the data on the time axis for statistic definitions that contain a time query (for example, the number of new employees in the last week). Permitted values are Hour , Day , Week , Month , Quarter , and Year .
Measurement runs to archive	The number of measurement run (apart from the current measurement) to be archived for displaying in the history. Enter the value 0 if you only want to retain the most recent measurement in each case.
Deactivated	Specifies whether the statistic definition is disabled. Statistic definition which are disabled are not calculated.
Preprocessor condition	You can add preprocessor conditions to statistics. This means that a statistic definition is only available when the preprocessor condition is fulfilled.
Disabled by preprocessor	If a statistic definition is excluded through a preprocessor condition, this option is set by the Database Compiler.
Instant calculation	Set this for statistic definitions, which are calculated at the moment they are displayed in the Web Portal. If this option is not set, the statistics are calculated during maintenance tasks.
Imported statistic data	Specifies whether these statistics are calculated at the moment they are displayed (for use in the Web Portal). If this option is not set, the statistics are calculated asynchronously by the DBQueue Processor.

Related topics

- [Querying statistic measurements](#) on page 174
- [Examples of statistic definitions](#) on page 183
- [Conditional compilation using preprocessor conditions](#) on page 335

Querying statistic measurements

Table 53: Measurement query properties

Property	Meaning
Measurements query	<p>Enter the complete database query in SQL syntax to determine the statistic measurements. The query must return the <code>ElementName</code> and <code>ElementValue</code> columns as results.</p> <p>To display statistic information in the Web Portal, you can also optionally output the <code>ElementObjectKey</code>, <code>ElementObjectKey2</code>, and <code>ElementValue2</code> columns.</p> <p>You can, optionally, control the display order of statistic measurements with the <code>ElementOrder</code> column. If the <code>ElementOrder</code> column does not exist, they are sorted by the <code>ElementName</code> column.</p>
Base measurements query	<p>Enter the complete database query in SQL syntax to determine the statistic measurements. The query must return the <code>ElementName</code> and <code>ElementValue</code> columns as results.</p> <p>To display statistic information in the Web Portal, you can also optionally output the <code>ElementObjectKey</code>, <code>ElementObjectKey2</code>, and <code>ElementValue2</code> columns.</p> <p>You can, optionally, control the display order of statistic measurements with the <code>ElementOrder</code> column. If the <code>ElementOrder</code> column does not exist, they are sorted by the <code>ElementName</code> column.</p> <p>The threshold factors entered in the fields Threshold green and Threshold red refer to the result in the <code>ElementValue</code> column. To determine the base measurement percentage, the result from column <code>ElementValue</code> is applied with 100%.</p> <p>NOTE: The name of the <code>ElementName</code> column in the base measurements query must match the name of the <code>ElementName</code> column in the measurements query.</p>
Condition	<p>Formulate a condition with which the statistic measurements can be limited to the current user. The condition has to be formulated as a valid WHERE clause for database queries and limits the result of the query further based on the <code>ElementObjectKey</code> using the variable <code>%UserID%</code> column.</p> <p>NOTE: The condition is only taken into account for statistics that are shown in the Web Portal.</p>

Example: Calculating the threshold

Threshold factors are used to determine the percentage of the base measurement that reflects a correct or acceptable status.

Table 54: Example of finding the state

Base Measure-ments	Threshold green	Threshold red	Percentage	State
100	0.25	0.75	< = 25	correct
			>25 to >75	acceptable
			>= 75	unacceptable
	0.75	0.25	> = 75	correct
			<75 to <25	acceptable
			<= 25	unacceptable

Related topics

- [General properties statistic definitions](#) on page 172
- [Examples of statistic definitions](#) on page 183

Disabling statistics definition

You have the option to disable individual statistic elements as required. Statistic definitions that are disabled are not calculated. Predefined user statistic definitions remain disabled even after the schema has been updated.

To edit an statistic definition

1. In the Designer, select a statistic definitions in the **User Interface > Statistics definitions** category.
2. In the edit view, select the **Properties** view.
3. Select the **Properties** tab and set the **Disabled** option.
4. Select the **Database > Save to database** and click **Save**.

In addition, statistic definitions can be disabled through pre-processor conditions.

Related topics

- [General properties statistic definitions](#) on page 172

Including statistics in the user interface

In order to visualize statistics in the One Identity Manager administration tools, such as the Manager, you have to link the statistics into the user interface as a custom menu item.

You will typically find statistics in the Manager under the **Info System** navigation item in nearly any category. You should set up custom menu items for statistics under an info system like this. All statistics that are defined at one menu level are displayed on one form.

You can show reports that you create in the Report Editor or in the Manager in the statistics. In the Manager's info system, the report opens when you double-click on the statistics header.

Statistics can also be linked as form elements into overview forms. To do this, use the Overview Form Editor.

NOTE: If you set up a custom info system, ensure that the menu item under which you define the statistics, is labeled with **Not expandable by user** and **Force open menu item**.

For more information about general properties of menu items, see [General menu item properties](#) on page 125. Take note of the following properties for menu items.

Table 55: Statistics properties

Property	Meaning
Entry type	Select the entry type Statistics .
Caption	The caption given here, overwrites the statistic definition caption. Leave this field empty if you want to use the statistic definition display name.
Statistics	Enter the statistic definition to be displayed.
Diagram type	Select the diagram type that is going to represent the statistic.
Alignment	Positioning of statistics on the overview form. This layout information is used if the statistic is used as a form element on an overview form.
Background	Background color of the form elements on the overview form. This layout information is used if the statistic is used as a form element on an overview form.

All menu items that are to be displayed in an application user interface have to be assigned to a permissions group and an application.

Related topics

- [Diagram types for visualizing statistics](#) on page 179
- [Examples of statistic definitions](#) on page 183
- [Creating new menu items](#) on page 123
- [Using reports in statistics](#) on page 177

- [Using simple reports in statistics](#) on page 178
- [Creating overview forms](#) on page 163

Using reports in statistics

In the Manager's info system, you can display reports that you create in the Report Editor as statistics. To do this, you must alter the **Manager's** user interface. The report opens when you double-click on the statistic's header.

To display a report as a statistic

1. In the Designer, create a user interface form.
 - a. In the Designer, select the **User interface > Forms > User interface forms** category.
 - b. Select the **Edit form** task.
 - c. Select the **Form > Insert** menu item.
 - d. Edit the interface form's main data.

Take the following cases into account:

- Use the **VI_Report** form definition
This form definition is configured for displaying in the graphical user interface and in web applications. You only need to set up one interface form for this. Which form template will be used to display the interface form is decided dynamically, depending on usage.
- In the form's configuration data, pass the name of the report to run (DialogReport.ReportName) in the Properties section.

Syntax:

```
<DialogSheetDefinition FormatVersion="1.0">
  <Properties>
    <Property Name="ReportName">ReportName from the
      DialogReport</Property> table
  </Properties>
</DialogSheetDefinition>
```

- e. Assign the user interface form to the applications and permissions groups.
2. In the Designer, create a menu item.
 - a. In the Designer, select **User interface > User interface navigation** category.
 - b. In the User Interface Editor, select the menu item for the statistics item to show the report.
 - c. Select **New**.

- d. Edit the main data of the menu item.
 - e. Assign the menu item to the **Manager** application and permissions groups.
3. Assign the user interface form to the menu item.
4. Select the **Database > Save to database** and click **Save**.

Related topics

- [Creating and editing reports in the Report Editor](#) on page 404
- [Creating user interface forms](#) on page 142
- [Assigning user interface forms to menu items](#) on page 146
- [Creating new menu items](#) on page 123
- [Using simple reports in statistics](#) on page 178

Using simple reports in statistics

Simple reports that you create in the Manager can be displayed as statistics in the Manager's info system. To do this, you must alter the **Manager**'s user interface in the Designer. In the Manager's info system, the report opens when you double-click on the statistic's header.

For more information about how to create reports in the Manager, see the *One Identity Manager Report Subscriptions Administration Guide*.

To display a simple report in the statistics

1. In the Designer, create a user interface form.
 - a. In the Designer, select the **User interface > Forms > User interface forms** category.
 - b. Select the **Edit form** task.
 - c. Select the **Form > Insert** menu item.
 - d. Edit the interface form's main data.

Take the following cases into account:

- Use the **VI_Report** form definition.

This form definition is configured for displaying in the graphical user interface and in web applications. You only need to set up one interface form for this. Which form template will be used to display the interface form is decided dynamically, depending on usage.
- In the form's configuration data, enter the UID of the simple report (RPSReport.UID_RPSReport) in the Properties section.

Syntax:

```

<DialogSheetDefinition FormatVersion="1.0">
  <Properties>
    <Property name="UIDRPSReport">UID_RPSReport from the
    RPSReport</Property> table
  </Properties>
</DialogSheetDefinition>

```

- e. Assign the user interface form to the **Manager** program and the permissions groups.
2. In the Designer, create a menu item.
 - a. In the Designer, select **User interface > User interface navigation** category.
 - b. In the User Interface Editor, select the menu item for the statistics item to show the report.
 - c. Select **New**.
 - d. Edit the main data of the menu item.
 - e. Assign the menu item to the **Manager** application and permissions groups.
3. Assign the user interface form to the menu item.
4. Select the **Database > Save to database** and click **Save**.

Related topics

- [Creating user interface forms](#) on page 142
- [Assigning user interface forms to menu items](#) on page 146
- [Creating new menu items](#) on page 123
- [Using reports in statistics](#) on page 177

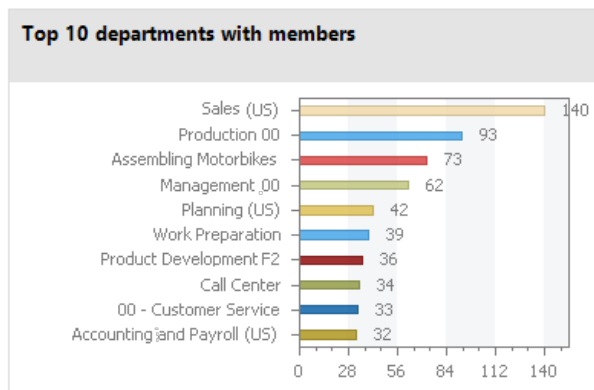
Diagram types for visualizing statistics

There are several diagram types available for visualizing statistics.

Bar chart

A bar chart can be used to visualize comparisons between measurements. The actual measurement of the ElementValue column and the identifier for ElementName column are used to label the diagram.

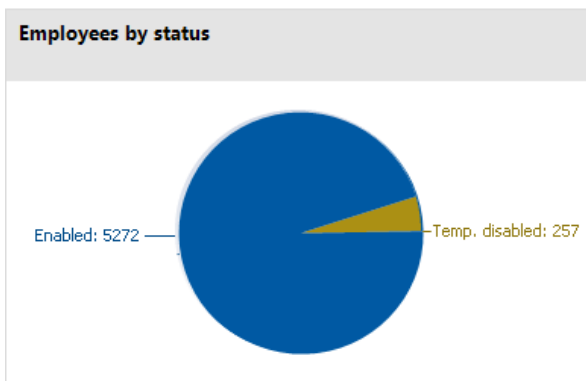
Figure 17: Bar chart example



Pie chart

A pie chart can be used to visualize the measurements as a percentage of the base measurement. The actual measurement of the ElementValue column and the identifier for ElementName column are used to label the diagram.

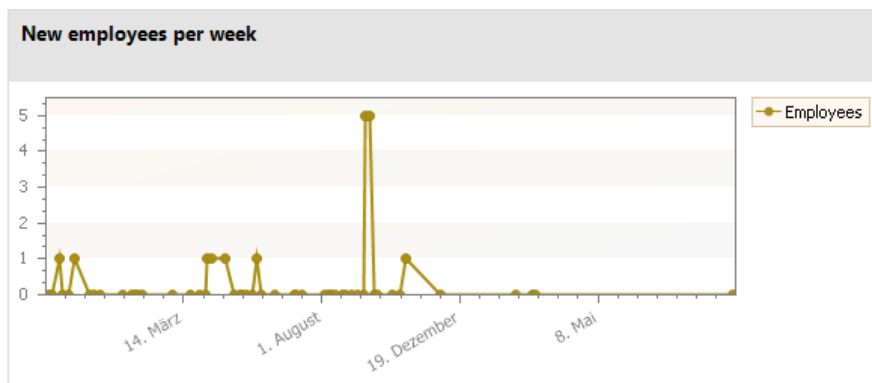
Figure 18: Pie chart example



Line diagram

A line diagram can be used to visualize a data sequence over a specified time period. The time axis is scaled in proportion to the time scale given in the statistic definition. The number of measurements in the line diagram results from measurement runs that are entered in the statistic definition from the history data. Click with the mouse on a point of measurement and a tooltip showing the measurement is displayed.

Figure 19: Line diagram example



Traffic light

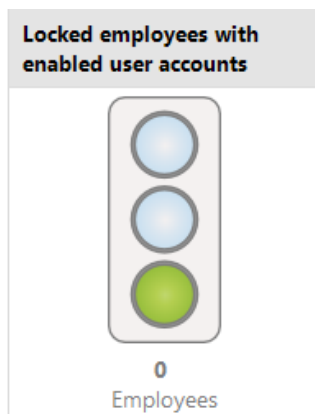
A traffic light diagram can be used to visualize the state of the system. The state is indicated by the color. The threshold factors given in the statistic definition determine when which status is reached.

Table 56: Meaning of the colors

Color	State
Green	correct
Yellow	acceptable
Red	unacceptable

The actual measurement of the ElementValue column and the identifier for ElementName column are used to label the diagram.

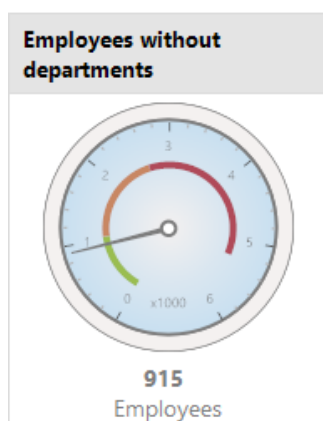
Figure 20: Traffic light example



Tachometer

A tachometer diagram can be used to visualize the state of the system in more detail than in a traffic light diagram. The base measurement is also displayed. The state is indicated by the color. The threshold factors given in the statistic definition determine when which status is reached. The actual measurement of the `ElementValue` column and the identifier for `ElementName` column are used to label the diagram.

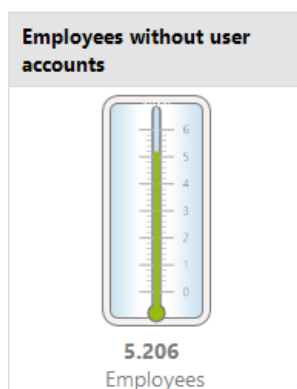
Figure 21: Tachometer diagram example



Thermometer

A thermometer diagram can be used to visualize the state of the system in more detail than in a traffic light diagram. The state is indicated by a color scale on the side of the diagram. The threshold factors given in the statistic definition determine when which status is reached. The actual measurement of the `ElementValue` column and the identifier for `ElementName` column are used to label the diagram.

Figure 22: Thermometer diagram example



Table

This diagram type can be used to visualize the measurements in table form. Enter a number of archived measurements runs in the statistic definition, to present the data over

a specified time period.

Figure 23: Table example

Number of employees		
	21.09.2017	
Employees	5.274	

Examples of statistic definitions

Example 1:

The number of people in the company should be displayed in the statistics. This statistic should be calculated daily. The statistics definition could look like:

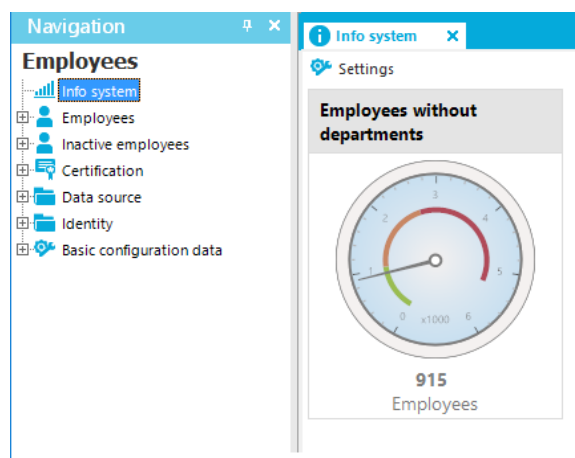
Statistic:	CountEmployees
Display name:	Number of employees
Description:	Finds the number of employees in the company on a daily basis.
Calculation schedule:	Calculate statistics
Measurements query:	<code>select 'Employees' as ElementName, count (*) as ElementValue from Person</code>

To display the statistics in the Manager in the **Employees > Info system** category, the following menu item is created:

Menu item:	Person.InfoSystem.CountEmployees
Item type:	Statistics
Sort order	1
Statistic:	Number of employees
Diagram type:	Thermometer

The menu item is assigned to the **Manager** program and an application role and can then be displayed in the Manager.

Figure 24: Displaying statistics in the Manager



Example 2:

The number of external employees in the company should be displayed in the statistics. This statistic should be calculated weekly. If more than 20% of employees in the company are external, the info system should display the state as acceptable instead of a correct. If more than 80% are external the state should be unacceptable.

Statistic:	CountExternalEmployees
Display name:	Number of external employees.
Description:	Find the number of external employees in the company on a weekly basis.
Calculation schedule:	Calculate weekly statistics
Measurements query:	<pre>Select 'Employees' as ElementName, Count (*) as ElementValue from Person where IsExternal = 1</pre>
Base measurements query:	<pre>Select 'Employees' as ElementName, Count (*) as ElementValue from Person</pre>
Threshold green:	0.2
Threshold red:	0.8

To display the statistics in the Manager in the **Employees > Info system** category, the following menu item is created:

Menu item:	Person.InfoSystem.CountExternalEmployees
Item type:	Statistics
Sort order	2
Statistic:	Number of external employees.
Diagram type:	Traffic light

The menu item is assigned to the **Manager** program and an application role and can then be displayed in the Manager.

Example 3:

The number of employees, for which the current user is entered directly as manager, should be represented in a statistic. Restrictions to the values for the current user are made through a condition.

Statistic:	CountEmployeesPersonHead
Display name:	Supervised employees
Description:	Finds the number of employees for which the manager is responsible on a daily basis.
Calculation schedule:	Calculate statistics
Measurements query:	<pre>select XObjectKey as ElementObjectKey, 'Employees' as ElementName, Count (*) as ElementValue from Person where IsExternal = 1 Group by XObjectKey</pre>
Condition:	<pre>ElementObjectKey in (select XObjectKey from Person where uid_PersonHead = '%userid%')</pre>

Configure the web project in the Web Portal, to display statistics in the Web Designer info system.

Example 4:

Internal and external employees, which the current user supervises as department manager, should be represented in a statistic. Departments are added here separately to determine clear results for displaying the measurement because a department manager might be responsible for more than one department.

Statistic:	PersonCountInternalExternal_By_Department
Display name:	Number of internal and external employees
Description:	Finds the number of internal and external employees per department on a daily basis.
Calculation schedule:	Calculate statistics
Measurements query:	<pre> select d.XObjectKey as ElementObjectKey, 'Internal' as ElementName, count(p.uid_person) as ElementValue from Department d Left Outer Join Person p on p.UID_Department = d.UID_Department and p.IsExternal = 0 Group By d.XObjectKey UNION ALL select d.XObjectKey as ElementObjectKey, 'External' as ElementName, count(p.uid_person) as ElementValue from Department d Left Outer Join Person p on p.UID_Department = d.UID_Department and p.IsExternal = 1 Group By d.XObjectKey </pre>
Condition:	<p>ElementObjectKey in</p> <pre> (select d.XObjectKey from Department d join helperheadorg hpo on d.UID_Department = hpo.UID_Org where hpo.UID_PersonHead = '%userid%') </pre>
Aggregate function	SUM

Configure the web project in the Web Portal, to display statistics in the Web Designer info system.

Example 5:

Ten employees with the highest risk index should be found and displayed in a statistic. They should be sorted by measurement unit.

Statistic:	Top10ActivePersonByRiskIndex
Display name:	Top 10 active employees by risk index
Description:	Find ten active employees with the highest risk indexes on a daily basis.
Calculation schedule:	Calculate statistics
Measurements	select top 10 p.InternalName as ElementName,

```

query:      Round(100 * IsNull(p.RiskIndexCalculated, 0), 0) as
            ElementValue,
            p.XObjectKey as ElementObjectKey,
            ROW_NUMBER() over (order by IsNull(p.RiskIndexCalculated, 0)
            desc, p.InternalName) as ElementOrder
            from Person p
            where p.IsInActive = 0
            order by ElementOrder

```

Configure the web project in the Web Portal, to display statistics in the Web Designer info system.

Extending the Launchpad

The Launchpad is the central tool for starting One Identity Manager administration tools and configuration tools. You can use the Launchpad to check the existing One Identity Manager installation and start One Identity Manager tools to run individual tasks.

The Launchpad can be customized. In the Designer, you can define your own menu items and actions for the Launchpad.

You can control how and where menu items are displayed in the Launchpad. You use the menu hierarchy and the different types of menu items to do this. For more information about the structure of a menu hierarchy and the individual menu items and their properties, see [User interface navigation](#) on page 112.

One Identity Manager supplies a number of Launchpad actions that you can use to start applications by using the Launchpad. You can also start your own applications over the Launchpad.

Detailed information about this topic

- [Recommendations for extending the Launchpad](#) on page 188
- [Actions for the Launchpad](#) on page 189
- [Creating new menu items and actions for the Launchpad](#) on page 190

Recommendations for extending the Launchpad

- To create a new category in the left-hand navigation area of the Launchpad, use menu items with the **Menu category** item type. The items are shown with their display text.
- To group together tasks in the main area of the Launchpad, use menu items with the **Task category** item type. The items are shown with their display text.
- For individual Launchpad tasks, use menu items with the **Task**, **Fixed menu item**, or **Data-dependent menu item** item types. The items are shown with their display text and description.
- Specify the order for displaying the menu items.
- To display the task status, enter an overlay icon definition on the menu item in VB.net syntax. Use the `NavigationNodeState` class.

Syntax:

```
public NavigationNodeState(string state, string imageUidOrName, string description)

public NavigationNodeState(string state, string imageUidOrName, string description, bool enabled, bool visible, int count)
```

Table 57: NavigationNodeState script parameters

Parameter	Description
State	Status returned, such as Info , Ok , Error , Warning .
ImageUidOrName	UID or name of the icon to be displayed.
description	Text displayed as tooltip.
enabled	Specifies whether the start button for the action is set.
visible	Specifies whether to display the task.
count	Number of affected objects.

Calling example:

```
Value = New NavigationNodeState("Ok", "QBM-33228392E9863141A9306B38ADF3D502", #LD("Project is completed.")#)
```

```
Value = New NavigationNodeState("Error", "QBM-a486f0eabf674392bbddf8572453258c", #LD("Project is not completed.")#)
```

- You can use the condition to specify whether the task is only available for a direct database connection or a connection over an application server. To do this, use the variable `SessionType`.

Condition examples:

Direct database connection only: '%SessionType%' = 'Direct'

Connect with the application server only: '%SessionType%' = 'AppServer'

If no condition is defined, the task is always available.

- If an action is going to be run from a task, link a Launchpad action to the menu item.

This displays the **Start** button for the task. The Launchpad action's description is displayed in the button's tooltip.

- If some actions in the Launchpad should not be made available to all users, you can manage the permissions by assigning Launchpad actions to program functions (QBMLaunchActionHasFeature table). Only tasks containing actions that the user's program function permissions permit him to run are shown in the Launchpad.

For more information about managing permissions and running Launchpad actions using program functions, see the *One Identity Manager Authorization and Authentication Guide*.

Related topics

- [Creating new menu items and actions for the Launchpad](#) on page 190
- [User interface navigation](#) on page 112
- [Actions for the Launchpad](#) on page 189

Actions for the Launchpad

One Identity Manager supplies a list of Launchpad actions that you can use to start applications. You can also start your own applications over the Launchpad.

At the start an application, you can pass calling parameters, tasks, and task parameters which the application can identify. Variable are permitted in this case. Supported are:

- Environment variables with the syntax %variable%
- Navigation variables with the syntax %variable%
- Columns of the object passed down in \$ notation.

To display Launchpad actions

1. In the Designer, select the **User interface > Launchpad Actions** category.
2. Select the Launchpad action The following main data is mapped for a Launchpad action.

Table 58: Action properties

Property	Description
Description	Detailed description of the action. The description is displayed in

Property	Description
	the tooltip.
.exe file	Full name of the binary file.
Calling parameter	Additional calling parameters for starting the application.
Action	Action identifier.
Administrative context	Specifies whether the application can only be started by an administrator. The application expects authentication as an administrator.
Method	Method that must also be transferred as a start parameter.
Task parameter	Additional parameters for the method.
Editor	Special editor, for completing tasks in the Launchpad.

Creating new menu items and actions for the Launchpad

In the Designer, you can define your own menu items and actions for the Launchpad.

To extend the Launchpad

1. Create new menu items for the Launchpad.
 - a. In the Designer, select **User interface > User interface navigation > Launchpad** category.
 - b. Start the User Interface Editor using the **Edit navigation for application 'Launchpad'** task.
 - c. Create the menu item.
 - d. Assign the menu items to the **Launchpad** application.
 - e. Assign the menu items from the permissions group to **QBM-LaunchPad**.
2. Assign the Launchpad actions to the menu items.
 - a. In the Designer, select the **User interface > Launchpad Actions** category.
 - b. Select the **View > Select table relations** menu item and enable the DialogTree table.
 - c. Select the Launchpad action and assign the menu item to the action using the **Menu items** tab.
3. (Optional) Assign a program function to the Launchpad action.
4. Select the **Database > Save to database** and click **Save**.

Related topics

- [Recommendations for extending the Launchpad](#) on page 188
- [Actions for the Launchpad](#) on page 189

Task definitions for the user interface

You use tasks to carry out specific actions on objects within One Identity Manager tools. Task definitions are created for object definitions. Depending on the selected objects, you have see different methods displayed in the user interface. These interface forms are made available to system users, taking into account their permissions group memberships, by the additional assignment of interface forms to permissions groups.

If a task definition is assigned a program function (QBMethodHasFeature table) users can only run this task if they have the necessary permissions groups. An error occurs if the user does not own this program function and tries to run it.

For more information about managing permissions and running methods using program functions, see the *One Identity Manager Authorization and Authentication Guide*.

Predefined configurations are maintained by the schema installation and cannot be edited apart from a few properties. You can disable individual predefined tasks to prevent them being shown in the user interface. They remain disabled even after schema installation.

NOTE: The tasks are displayed in alphabetical order in the task view of One Identity Manager.

Apart from these object dependent task definitions, form methods and customizer are provided that cannot be edited.

Detailed information about this topic

- [Disabling task definitions](#) on page 192
- [Creating and editing task definitions](#) on page 191
- [Script for conditional displaying of tasks](#) on page 193
- [Properties of task definitions](#) on page 193

Creating and editing task definitions

To create or edit a task definition

1. In the Designer, select **User Interface > Task definitions** category.
 2. Select the task.
- OR -

- To create a new task, select the **Object > New** menu item.
3. Select the **Change main data** task and edit the task's main data.
 4. Assign a permissions group to the task definition.
 - a. Select the **View > Select table relations** menu item and enable the `DialogGroupHasMethod` table.
 - b. In the edit view, select the **Permissions group** view and select the permissions groups.
 5. Assign the task definition to the object definition for which the task should be offered in the user interface.
 - a. Select the **View > Select table relations** menu item and enable the `DialogObjectHasMethod` table.
 - b. In the edit view, select the **Object** view and select the object definition.
 6. (Optional) Assign a program function to the task definition.
 - a. Select the **View > Select table relations** menu item and enable the `QBMethodHasFeature` table.
 - b. In the edit view, select the **Program function** view and select the program function.
 7. Select the **Database > Save to database** and click **Save**.

Related topics

- [Disabling task definitions](#)
- [Script for conditional displaying of tasks](#) on page 193
- [Properties of task definitions](#)

Disabling task definitions

You can disable individual predefined tasks to prevent them being shown in the user interface. They remain disabled even after schema installation.

To disable a task

1. In the Designer, select the task in the **User interface > Task definitions** category.
2. Select the **Change main data** task.
3. Set the **Disabled** option.
4. Select the **Database > Save to database** and click **Save**.

Related topics

- [Creating and editing task definitions](#) on page 191
- [Script for conditional displaying of tasks](#) on page 193

- [Properties of task definitions](#) on page 193

Script for conditional displaying of tasks

In principle, a user's permissions for displaying and editing tasks are controlled by permissions in permissions groups.

You can also use a script in the custom task definition to conditionally display a task. For example, this way you can control whether a task is only displayed in the Manager if a certain condition is met.

NOTE: The script does not change the user's permissions but simply the behavior if the object is loaded in one of the One Identity Manager tools. If you want to limit visibility and editability of a task, change the permissions of the permissions groups. For more information, see the *One Identity Manager Authorization and Authentication Guide*.

To specify a script for conditionally displaying a task

1. In the Designer, select the task in the **User interface > Task definitions** category.
2. Select the **Change main data** task.
3. Enter a **Visibility script** in VB.Net syntax. If the return value is **false**, task is not displayed in any of the One Identity Manager tools.
4. Select the **Database > Save to database** and click **Save**.

Related topics

- [Creating and editing task definitions](#) on page 191
- [Disabling task definitions](#) on page 192
- [Properties of task definitions](#) on page 193

Properties of task definitions

Table 59: Task properties

Property	Meaning
Task name	Name of the task.
Caption	The display name is used to display the task in the administration tool task view. Display names can be given in more than one language. NOTE: The tasks are displayed in alphabetical order in the task view of One Identity Manager.
Description	Description of the task. The description is shown as a tooltip in the user interface task view.

Property	Meaning
Enabled for	<p>This property specifies the intended use of the task.</p> <p>Permitted values are:</p> <ul style="list-style-type: none"> • Fat Client: You can display the task in the graphical user interface. • Web Client: You can display the task in web applications. • Fat Client + Web Client: You can display the task in both the graphical user interface and web applications.
Task behavior	<p>Sets the behavior of the task.</p> <p>The following entries are permitted:</p> <ul style="list-style-type: none"> • No data: Default. The task is available for single object and multiple object editing. Changes are run separately for each object, even if multiple edit is used. • Save required: The task saves data. A corresponding alert message is displayed. • Single objects only: This task is only permitted for single objects. • Save required + single objects only: The task saves data. A corresponding alert message is displayed. This task is only permitted for single objects. • Run on multiple objects: This task is available for multiple editing of objects. Changes are run for all objects together through a multi-object. • Save required + run on multiple objects: The task saves data. A corresponding alert message is displayed. This task is available for multiple editing of objects. Changes are run for all objects together through a multi-object.
Icon	Icon for displaying the task in the user interface.
Script	<p>Task script. You can use function calls or commando input in VB.Net statements for the task script. The Base. syntax always accesses the object that is currently loaded.</p> <p> NOTE: The database needs to be compiled after changing a task script.</p>
Visibility script	Script for conditional displaying of tasks in One Identity Manager tools. For more information, see Script for conditional displaying of tasks on page 193.
Disabled	Specifies whether the task is displayed in the user interface. Disabled tasks are never displayed in the user interface. Predefined system users are not effected by this limitation. This modification is also permitted for predefined default user interface tasks and is not overwritten when the schema is installed.

Property	Meaning
Processing status	The processing status is used for creating custom configuration packages.
Object	Assignment to object definitions (DialogObjectHasMethod table) for which the task will be shown in the user interface.
Permissions group	Assignment of permissions groups (DialogGroupHasMethod table), whose users can apply this task.
Program function	Program function, which is linked to the task definition. If a task definition is assigned a program function (QBMMethodHasFeature table) users can only run this task if they have the necessary permissions groups. For more information about managing permissions and running methods using program functions, see the <i>One Identity Manager Authorization and Authentication Guide</i> .

Related topics

- [Creating and editing task definitions](#) on page 191
- [Task definitions for the user interface](#) on page 191
- [Visual Basic .NET scripts usage](#) on page 341
- [Using #LD-notation](#) on page 354
- [Language-dependent data representation](#) on page 210

Applications for configuring the user interface

In the default version of One Identity Manager, the applications and the predefined navigation for the One Identity Manager tools, the Manager, the Designer, and the Launchpad are also supplied. Predefined configurations are maintained by the schema installation and cannot be edited apart from a few properties. It is not usually necessary to define your own applications. You might possibly need your own applications for a customer specific web interface.

In the Designer, the available programs are shown in the **Base data > Security settings > Programs** category.

Table 60: Predefined programs

Product	Meaning
Default	Default for front-ends without special usage, for example Job Queue Info or Report Editor. Required to

Product	Meaning
	determine the authentication module.
Designer	Application for the Designer.
Manager	Application for the Manager.
Launchpad	Application for the Launchpad.
WebDesigner	Application for installing the Web Portal.
WebDesignerEditor	Program for the Web Designer to configure and extend the Web Portal.
Application server	Program for installing the application server.
SOAP Service	Application for installing the SOAP Web Service.
SPML Service	Program for installing the SPML Web service.
API Designer	Application for the Web Portal configuration.
OperationsSupportWebPortal	Application for installing the Operations Support Web Portal.
PasswordReset	Application for installing the Password Reset Portal.

Program properties

Table 61: Program properties

Property	Meaning
Program	Name of the program.
Remarks	Comments about the program.
Start menu item	If the given start menu item is available to a system user in a program's navigation menu, the program navigates straight to this position in the menu when it starts up. You can specify, for example, a home page for a system user with this feature. This function is mainly used by web interfaces.
Configuration data	Configuration data is used to determine a system user by the dynamic authentication module. You can also adapt the configuration data for the default programs that are supplied. For more information, see the <i>One Identity Manager Authorization and Authentication Guide</i> .
Minimum Version	Minimum version of the program that can run with the version of the database in use. This input is used solely as information, the version number is not verified.
Engine based	Specifies whether menu navigation and forms can be assigned to the

Property	Meaning
	program.
Processing status	The processing status is used for creating custom configuration packages.
Authentication module	Authentication module used by the program. For more information about One Identity Manager authentication modules, see the <i>One Identity Manager Authorization and Authentication Guide</i> . To display the authentication module for a program <ol style="list-style-type: none"> 1. In the Designer, select the Base data > Security settings > Programs category. 2. Select the View > Select table relations menu item and enable the DialogProductHasAuthentifier table. 3. Click the Authentication module tab.
Form	Forms displayed in the program. To display forms for a program <ol style="list-style-type: none"> 1. In the Designer, select the Base data > Security settings > Programs category. 2. Select the View > Select table relations menu item and enable the DialogProductHasSheet table. 3. Select the Form assignment tab.
Menu	Menus displayed in the program. To display menu items for a program <ol style="list-style-type: none"> 1. In the Designer, select the Base data > Security settings > Programs category. 2. Select the View > Select table relations menu item and enable the DialogTreeInDialogProduct table. 3. Select the Menu assignment tab.
System users	System users that use the program. To display system users for a program <ol style="list-style-type: none"> 1. In the Designer, select the Base data > Security settings > Programs category. 2. Select the View > Select table relations menu item and enable the DialogUserConfiguration table. 3. Select the System user tab.

Property	Meaning
Permissions group	Permissions groups whose permissions are also valid for this program. For more information about permissions groups in One Identity Manager, see the <i>One Identity Manager Authorization and Authentication Guide</i> .

To display permissions groups for a program

1. In the Designer, select the **Base data > Security settings > Programs** category.
2. Select **View > Select table relations** and enable the DialogGroupInProductLimited table.
3. Select the **Permissions group** tab.

Related topics

- [Applications for configuring the user interface](#) on page 195

Icons and images for configuring the user interface

When you are configuring the One Identity Manager tools' user interfaces you can add icons and images for displaying in different parts of them. The default version of One Identity Manager supplies the icons and images that you can use for configuring the user interface and to create reports. Predefined configurations are maintained by the schema installation and cannot be edited apart from a few properties.


Icons are required to be in PNG format with sizes of 16x16 pixels, 24x24 and 32x32 pixels for the graphical interface.

Icons are required for the following use cases.



Table 62: Meaning of the icons

State	Usage
Normal	Icons showing an enabled state. These icons must exist in the One Identity Manager database.
Inverted	Icons that show an enabled state on a black background. These icons can be converted automatically.
Disabled	Icons showing a disabled state. These icons must exist in the One Identity Manager database.

To add an icon

1. In the Designer, select the **Base data > General > Icons** category.
2. Select the **Object > New** menu item.
3. Give the icon a name.
4. Upload the icon using .
5. Select the **Database > Save to database** and click **Save**.

To add images for reports

1. In the Designer, select the **Base data > General > Large images** category.
2. Select the **Object > New** menu item.
3. Give the image a name.
4. Load the image using the  button.
5. Save the image with the  button.
The image is displayed with certain important image properties.
6. Select the **Database > Save to database** and click **Save**.

NOTE: If you edit icons or images that are used in a web application, you must compile the API and the web application afterward with the Database Compiler.

Using predefined database queries

Direct database queries cannot be carried out from front-ends and web application when an application server is implemented due to security issues. Database queries, which are required on forms for example, must be formulated in One Identity Manager as predefined database queries. Database queries are always run with the permissions of the current user. Predefined database queries must be assigned to a permissions group.

A wizard in the Web Designer helps you to create database queries for the Web Portal and to link it with at least one permissions group. You can enter more predefined database queries in the Designer.

To create predefined database queries

1. In the Designer, select the **Base data > Advanced > Predefined SQL** category.
2. Select the **Object > New** menu item.
3. Edit the main data.

Table 63: Properties of predefined database queries

Property	Description
Processing status	Object processing status. The processing status is used for creating custom configuration packages.
Description	Text field for additional explanation.
Identifier for SQL code	A unique identifier that is used to identify the query
SQL type	Type of SQL query. Permitted values are: <ul style="list-style-type: none">• Statement: This is a complete SQL query.• WHERE clause: This is where clause part of the SQL query.
Code	Full database query SQL syntax. You can also use SQL parameters in the query.

4. Assign permissions groups.
 - a. Select **View > Select table relations** and enable the QBMGroupHasLimitedSQL table.
 - b. In the edit view, select the **Permissions group** view and select the permissions groups.
5. Select the **Database > Save to database** and click **Save**.

Localization in One Identity Manager

One Identity Manager requires country information at different stages, for example, employee country and state assignments are accessed when email notifications are created or IT Shop workflows are being determined. Language, time zones, public holidays, and working hours are mapped as well as countries and states. The basis data is loaded into the database during schema installation.

One Identity Manager supports language-dependent representation of data. You can use this feature to edit display text in different languages for the One Identity Manager tool user interfaces. You can also create multi-language text for process information output, script processing as well as processing messages.

The default One Identity Manager installation is supplied in the **English - United States [en-US]** and **German - Germany [de-DE]** language. You can add other languages to the user interface and display text if required. In this instance, you must translate the text before One Identity Manager goes live. There is a Language Editor in the Designer to help you do this. A special control is provided in the One Identity Manager tools that aids multi-language input.

To help you translate One Identity Manager, other languages are made available with the Web Portal Language Pack.

NOTE: You will find the One Identity Manager Language Pack in the Support Portal under <https://support.oneidentity.com/>.

Detailed information about this topic

- [Language settings for displaying and maintaining the data](#) on page 38
- [Working in different time zones](#) on page 202
- [Determining working hours](#) on page 202
- [Editing country information](#) on page 203
- [Language-dependent data representation](#) on page 210
- [Importing translations from the language pack](#) on page 219

Working in different time zones

Time stamps, for example, insert dates or modification dates, are stored in One Identity Manager with the respective UTC. The object layer transforms this time data into the currently valid time zone data when an object is loaded. The user, therefore, sees all the values in local time. When an object is saved the current time zone data is transformed into UTC data.

Countries and time zones are linked to another in the One Identity Manager schema. This makes it easier to find out the time zones when web fronts such as the Web Portal are in use.

Related topics

- [Setting countries and states](#) on page 203
- [Editing countries](#) on page 205
- [Editing states](#) on page 206

Determining working hours

Working hours are calculated for various function in One Identity Manager. For example, to determine working hours in the IT Shop or for determining reaction and solution times for calls in the Helpdesk Module. Weekends and public holidays are taken into account when calculating working hours.

To determine the working hours

- Ensure that a state and/or county is entered into the employee's main data.
- Public holiday are entered by state (county) in One Identity Manager. You can add separate public holidays for states.
- To exclude public holidays from the working hours calculation, in the Designer set the **QBM | WorkingHours | IgnoreHoliday** configuration parameter.
- To exclude weekends from the working hours calculation, in the Designer, set the **QBM | WorkingHours | IgnoreWeekend** configuration parameter.

Related topics

- [Specifying the standard hours for countries and states/provinces/counties](#) on page 204
- [Displaying public holidays for countries and states](#) on page 204
- [Editing countries](#) on page 205
- [Editing states](#) on page 206

Editing country information

One Identity Manager requires country information at different stages, for example, employee country and state assignments are accessed when email notifications are created or IT Shop workflows are being determined.

Language, time zones, public holidays and working hours are mapped as well as countries and states. The basis data is loaded into the database during schema installation.

Detailed information about this topic

- [Setting countries and states](#) on page 203
- [Specifying the standard hours for countries and states/provinces/counties](#) on page 204
- [Displaying public holidays for countries and states](#) on page 204
- [Editing countries](#) on page 205
- [Editing states](#) on page 206
- [Country properties](#) on page 207
- [State properties](#) on page 208
- [Public holiday properties](#) on page 209

Setting countries and states

To enable a country

1. In the Designer, select the **Base data > Localization > Country > Disabled** category.
2. Select a country.
3. Set **Enabled**.

To enable a state

1. In the Designer, select the **Base data > Localization > Country > <country name> > States** category.
2. Select a state.
3. Set **Enabled**.

Related topics

- [Working in different time zones](#) on page 202
- [Editing countries](#) on page 205
- [Editing states](#) on page 206

Specifying the standard hours for countries and states/provinces/counties

Specify the working hours which apply for the countries and states. Working hours are taken into account when calculating time periods, for example in the IT Shop.

To edit the working hours of a country

1. In the Designer, select the **Base data > Localization > Country** category.
2. Select a country.
3. Under **Hours (default)** specify the default working hours.

To edit the default working hours for a state

1. In the Designer, select the **Base data > Localization > Country > <country name> > States** category.
2. Select a state.
3. Under **Hours (default)** specify the default working hours.

Related topics

- [Determining working hours](#) on page 202

Displaying public holidays for countries and states

The holidays are loaded into the database during the schema installation and do not normally have to be customized.

To display the public holidays of a country

- In the Designer, select the **Base data > Localization > Country > <country name> > Public holidays** category.

To display the public holidays of a state

- In the Designer, select the **Base data > Localization > Country > <country name> > States > <state> > Public holidays** category.

Related topics

- [Public holiday properties](#) on page 209
- [Determining working hours](#) on page 202

Editing countries

The countries are loaded into the database during the schema installation and do not normally have to be customized.

NOTE: For enabled countries, an entry with the country name is displayed in the Designer in the **Base data > Localization > Country** category. Countries that are not enabled are displayed in **Base data > Localization > Country > Disabled**.

To edit a country

1. In the Designer, select the **Base data > Localization > Country** category.
2. Select a country.
3. Edit the main data.
4. (Optional) Assign the language to the country.
 - a. Select **View > Select table relations** and enable the DialogCountryHasCulture table.
 - b. On the **Languages** tab, select the languages.
5. (Optional) Assign the time zones to the country.
 - a. Select **View > Select table relations** and enable the DialogCountryHasTimeZone table.
 - b. On the **Time zones** tab, select the time zones.
6. (Optional) Assign the public holidays to the country.
 - a. Select **View > Select table relations** and enable the DialogCountryHoliday table.
 - b. On the **Holidays** tab, select the public holidays.
7. (Optional) Assign the states/provinces/counties to the country.
 - a. Select **View > Select table relations** and enable the DialogState table.
 - b. On the **States** tab, select the states.

Related topics

- [Working in different time zones](#) on page 202
- [Setting countries and states](#) on page 203
- [Specifying the standard hours for countries and states/provinces/counties](#) on page 204
- [Displaying public holidays for countries and states](#) on page 204
- [Editing states](#) on page 206
- [Country properties](#) on page 207

Editing states

The states are loaded into the database during the schema installation and do not normally have to be customized.

To edit a state

1. In the Designer, select the **Base data > Localization > Country > <country name> > States** category.
2. Select a state.
3. Edit the main data.
4. (Optional) Assign languages to the state.
 - a. Select **View > Select table relations** and enable the `DialogStateHasCulture` table.
 - b. On the **Languages** tab, select the languages.
1. (Optional) Assign time zones to the state.
 - a. Select **View > Select table relations** and enable the `DialogStateHasTimeZone` table.
 - b. On the **Time zones** tab, select the time zones.
2. (Optional) Assign public holidays to the state.
 - a. Select **View Select table relations** and enable the `DialogStateHoliday` table.
 - b. On the **Holidays** tab, select the public holidays.

Related topics

- [Working in different time zones](#) on page 202
- [Setting countries and states](#) on page 203
- [Specifying the standard hours for countries and states/provinces/counties](#) on page 204
- [Displaying public holidays for countries and states](#) on page 204
- [Editing countries](#) on page 205
- [State properties](#) on page 208

Country properties

Table 64: Country properties

Property	Description
Country name	Name of the country.
Description	Description of the country.
Enabled	If this option is set, this country is can be selected from the list in the administration tools. This helps to limit the selection of time zones and languages.
Daylight saving time	Specifies whether daylight saving time is taken into account when the difference to UTC time is calculated.
Hours (default)	Specify the working hours which apply across the country. Working hours are taken into account when calculating time periods, for example in the IT Shop.
Country name (national language)	Name of the country in the national language using the national script.
Capital city (national language)	Name of the capital city in the national language using the national script.
Country code	International telephone code for the country.
International vehicle reg. ID	International identifier for vehicle license plates.
ISO code (2-letter)	Two letter country code for this country. This data has to comply with ISO 3166, a standard for coding geographical units.
ISO code (3-letter)	Three letter country code for this country. This data has to comply with ISO 3166, a standard for coding geographical units.
ISO code (numeric)	Numeric country code for this country. This data has to comply with ISO 3166, a standard for coding geographical units.
Object class	Object class for mapping country data in an LDAP schema.
Search mask	Search mask for mapping country data in an LDAP schema.
UTC Offset (average)	Average time difference between country and UTC time. This value is calculated by the DBQueue Processor based on the country's time zones.
Language	Language and language code of the country. The language specifies the language for email notification sent to users.
Time zones	The country's time zone. The calculation of processes that are time

Property	Description
	dependent, such as in the IT Shop, is taken in account by specifying a time zone.
Holidays	National holidays.
States	States within this country.

Related topics

- [Working in different time zones](#) on page 202
- [Determining working hours](#) on page 202
- [Setting countries and states](#) on page 203
- [Specifying the standard hours for countries and states/provinces/counties](#) on page 204
- [Displaying public holidays for countries and states](#) on page 204
- [Editing countries](#) on page 205
- [Editing states](#) on page 206

State properties

Table 65: State properties

Property	Description
State	Name of the state.
State name (national language)	Name of the state in the national language using the national script.
Country	Enter the country that the state belongs to.
Enabled	Use this option to mark the states that your system uses.
Daylight saving time	Specifies whether daylight saving time is taken into account when the difference to UTC time is calculated.
Hours (default)	Specify the working hours which apply across the state. Working hours are taken into account when calculating time periods, for example in the IT Shop.
Capital city	Name of the state's capital.
Capital city (national language)	Name of the capital city in the national language using the national script.

Property	Description
Short name	Code according to ISO 3166-2 for the state, such as CA for California or SN for Saxony.
UTC Offset (average)	Average time difference between country and UTC time. This value is calculated by the DBQueue Processor based on the state's time zones.
Language	Language and language code of the country. The language specifies the language for email notification sent to users.
Time zones	The country's time zone. The calculation of processes that are time dependent, such as in the IT Shop, is taken in account by specifying a time zone.
Holidays	Public holidays of the state.

Related topics

- [Working in different time zones](#) on page 202
- [Determining working hours](#) on page 202
- [Setting countries and states](#) on page 203
- [Specifying the standard hours for countries and states/provinces/counties](#) on page 204
- [Displaying public holidays for countries and states](#) on page 204
- [Editing countries](#) on page 205
- [Editing states](#) on page 206

Public holiday properties

Table 66: Public holiday properties

Property	Description
Date (ISO Format)	The date of the public holiday is entered in ISO format, for example, yyyy-mm-dd where: yyyy - year, four digits mm - month, two digits dd - day, two digits
Public holiday name	Name of the holiday.
Public holiday name (national language)	Name of the holiday in the national language using the national script.

Property	Description
Country/State	Name of the country/state for the public holiday.
Processing status	The processing status is used for creating custom configuration packages.
Disabled	Specifies whether the public holiday is disabled.

Related topics

- [Determining working hours](#) on page 202
- [Displaying public holidays for countries and states](#) on page 204
- [Editing countries](#) on page 205
- [Editing states](#) on page 206

Language-dependent data representation

One Identity Manager supports language-dependent representation of data. You can use this feature to edit display text in different languages for the One Identity Manager tool user interfaces. You can also create multi-language text for process information output, script processing as well as processing messages.

Detailed information about this topic

- [Basic rules for using language-dependent data](#) on page 211
- [Flagging columns for translation](#) on page 212
- [Using the text memory for translation](#) on page 214
- [Displaying translations in the Language Editor](#) on page 214
- [Showing usage of a translation](#) on page 215
- [Editing translations of a single table](#) on page 216
- [Editing all translations](#) on page 217
- [Changing translation keys](#) on page 218
- [Importing translations from the language pack](#) on page 219

Related topics

- [Language settings for displaying and maintaining the data](#) on page 38

Basic rules for using language-dependent data

In order to use multi-language data representation in One Identity Manager, the following prerequisites need to be fulfilled:

- The language is set up in the database and labeled with the **Select in front-end** option.
- A fallback language for the database is set. In default installation of One Identity Manager, the language used is **English - United States [en-US]**. This language is used if there is no translation available for a language-dependent data break down in the user's requested language.
- The **Multilingual** option has to be set on the column definitions in order to use multi-language display text.
- Source and target of the translation are known.
- #LD notation is used for outputting language-dependent data from within Visual Basic .NET expressions. #LD text is automatically extracted for translation. To do this, a column must be labeled as #LD content.

The translations are stored in the DialogMultiLanguage table. A key, the language and the translation are entered into the table.

Example:

The **QERResource.Ident_QERResource** column has the Resource text displayed as its column name in the login language English - United States [en-US] (**DialogColumn.Caption**). The **Ressource** column name should be used for the login language **German - Germany [de-DE]**.

The **QERResource.Ident_QERResource** column contains the value **Car**. A user with the login language **English - United States [en-US]** is shown the **Car** value. A user with the login language **German - Germany [de-DE]** is shown the **Auto** value.

Table 67: Example of language-dependent entries in the DialogMultiLanguage table

Column Name	Key	language	Value
DialogColumn.Caption	Resource	English - United States [en-US]	Resource
	Resource	German - Germany [de-DE]	Resource

Column Name	Key	language	Value
QERResource.Ident_ QERResource	Car	English - United States [en-US]	Car
	Car	German - Germany [de- DE]	Auto

Related topics

- [Flagging columns for translation](#) on page 212
- [Using the text memory for translation](#) on page 214
- [Displaying translations in the Language Editor](#) on page 214
- [Language settings for displaying and maintaining the data](#) on page 38
- [Using #LD-notation](#) on page 354

Flagging columns for translation

Columns must be marked for translation in order to enter multilingual captions.

To label a column for translation

1. In the Designer, select the **One Identity Manager schema** category.
2. Select the table and start the Schema Editor with **Show table definition**.
3. Select the column and then the **Column properties** view.
4. Select the **Column** tab and edit the **Multilingual** property. Specify the following settings:
 - **Translation target:** The column content is displayed in translation.
 - **Translation source:** The column supplies the translation.
 - **#LD content:** The column has contents in #LD notation. The contents are extracted for translation.
 - **Without text memory fallback:** The text store is not used as fallback for the column.

You can combine the values. The combination of values determines the resulting translation.

5. A translation target is normally the same as the translation source. If the translation, however, is taken from another translation source, enter this additionally as a language dependency.

- Switch to the **Language dependencies** tab. Under **Translation source**, select the column that is to be used as the translation source.

NOTE: Ensure that the column used as a translation source has been labeled with **Translation source**.

Example: A column is translation target and source

The contents of the QERResource.Ident_QERResource column are to be translated. The Ident_QERResource column contains the value **Car**. A user with the login language **English - United States [en-US]** should be shown the value **Car**. A user with the login language **German - Germany [de-DE]** should be shown the value **Auto**. The actual translation should be maintained in the QERResource.Ident_QERResource column.

- Label the QERResource.Ident_QERResource column in the **Multilingual** property with **Translation target**.
- Label the QERResource.Ident_QERResource column in the **Multilingual** property with **Translation source**.
- In the Language Editor, translate the entries for the Ident_QERResource column of the QERResource table.

When the column is loaded, it is determined that QERResource.Ident_QERResource should be translated. For translation, the relevant key for the QERResource.Ident_QERResource column is determined from the DialogMultiLanguage table and the value saved for the user's login language is displayed.

Example: A column is translation target and takes its translation from another translation source

The action is displayed in the Manager process view in the current user's login language. The contents of the column DialogProcess.DisplayName are taken from the column JobEventGen.ProcessDisplay. The column JobEventGen.ProcessDisplay may use #LD notation to create the display string.

- Label the JobEventGen.ProcessDisplay column in the **Multilingual** property with the values **Translation source** and **#LD content**.
- Label the DialogProcess.DisplayName column in the **Multilingual** property with **Translation target** and as the **Language dependency**, enter the JobEventGen.ProcessDisplay column.
- In the Language Editor, translate the entries for the ProcessDisplay column of the JobEventGen table.

When the column is loaded, it is determined that `DialogProcess.DisplayName` should be translated. For translation, the relevant key for the `JobEventGen.ProcessDisplay` column is determined from the `DialogMultiLanguage` table and the value saved for the user's login language is displayed.

Related topics

- [Using the text memory for translation](#) on page 214
- [Displaying translations in the Language Editor](#) on page 214
- [Column definition properties](#) on page 91
- [Using #LD-notation](#) on page 354

Using the text memory for translation

Translations, which occur frequently or cannot be associated with a particular database column, can be stored in a text memory (`QBMTTranslationAddOnSource` table). The Web Portal, for example, takes its translations from the text store. In the same way, output text from database triggers is found in the text store. You can reference the text store as a translation source.

The text store is also used as a fallback when a fitting translation cannot be found through other translation sources.

TIP: To deactivate use of the text memory as a fallback, flag the column in the **Multilingual** property with **Without fallback translation source**.

To enter an item in the text store

1. In the Designer, select the **Base Data > Localization > Translatable texts** category.
2. Select the **Object > New** menu item and enter the translation key.
3. In the Language Editor, translate the entries for `QBMTTranslationAddOnSource.Entrykey`.

Related topics

- [Flagging columns for translation](#) on page 212

Displaying translations in the Language Editor

With the Language Editor you can carry out translations for:

- The content of column labeled for multi-language input
- #LD expressions from columns containing VB.Net code
- Text stored in the text store (table QBMTTranslationAddOnSource)

All translatable entries that are shown with their translation status in the Language Editor translation table.

To display the translations

1. In the Designer, select the **Base data > Localization** category.
2. Start the Language Editor using **Edit translation in database**.

The following information is displayed.

Table 68: Information in the translation table

Properties	Meaning
State	Current state of the item.
Table	Translation source table.
Column	Translation source column.
Source	Specifies where the key comes from. Permitted values are Data , Bitmask , List of permitted values , Part of a multi-value column , DBQueue Processor , LD notation , Web , and External .
Usage	Number of time the translation is used.
Key	Key value to be translated.
Checked	Specifies whether the translation has been tested.
Language	Translation in the selected language.

TIP: Click with the mouse in a column header to sort by the selected column.

Related topics

- [Showing usage of a translation](#) on page 215
- [Editing translations of a single table](#) on page 216
- [Editing all translations](#) on page 217
- [Changing translation keys](#) on page 218

Showing usage of a translation

You can edit translations in different places. When you change a translation, all those places are shown. Before changing a translation, check in the Designer about how it is

used.

To display usage of a key

1. In the Designer, select the **Base data > Localization** category.
2. Start the Language Editor using **Edit translation in database**.
3. In the **Usage** column for the required entry, select the **Show usage** context menu.
This opens a dialog showing all the occurrences of where the key is used. Double-click an entry to view the advanced properties of an object.

Related topics


- [Editing translations of a single table](#) on page 216
- [Editing all translations](#) on page 217
- [Changing translation keys](#) on page 218

Editing translations of a single table


NOTE:

- You can edit translations in different places. When you change a translation, all those places are shown. Before changing a translation, check in the Designer about how it is used.
- Users can only edit object and column translations for which they have permissions.
- To edit all translations, users require the **Allow translation of text data regardless of the edit permissions for the base object** program function (Common_Translation).

To translate the contents of a single table

1. In the Designer, select the **One Identity Manager schema** category.
2. Select the table and start the Language Editor using **Edit translation in table**.
The Language Editor shows all available translations from the columns of the selected table that are labeled for translation.
3. Select the languages of the translations you want to edit under **Select languages** in the toolbar.
4. To identify entries for which no translation is currently available, click .
5. Edit the translations.
 - a. Double-click the input field to unlock it and enter the translation.
 - b. Once you have checked a translation, set the translation to **Checked**.

TIP:

- Use **Ctrl + K** to transfer the key value to the translation.
- Use  to run a grammar check.
- Use the **Edit** context menu to:
 - Enter the data source of a key.
 - Add a comment.
 - Delete a translation for a language.

After the changes have been committed to the main database, the system data must be recalculated by the DBQueue Processor in order make the new multi-language data available to all system users.

For detailed information on translating the permitted values of a column, see [Permitted column values](#) on page 84.

Related topics


- [Showing usage of a translation](#) on page 215
- [Editing all translations](#) on page 217
- [Changing translation keys](#) on page 218





Editing all translations

NOTE:

- You can edit translations in different places. When you change a translation, all those places are shown. Before changing a translation, check in the Designer about how it is used.
- Users can only edit objects and columns for which they have permissions.
- To edit all translations, users require the **Allow translation of text data regardless of the edit permissions for the base object** program function (Common_Translation).

To display and edit all translations in the Language Editor

1. In the Designer, select the **Base data > Localization** category.
2. Start the Language Editor using **Edit translation in database**.
3. Select the languages of the translations you want to edit under **Select languages** in the toolbar.
4. To identify entries for which no translation is currently available, click .
5. Use filters to limit how much data is displayed, if necessary.


- Click in a column header to sort by the selected column.
- To limit column entries, click the arrow in the column header. This opens a field into which you can enter filter text. If a filter is defined, the column header displays an  icon. To delete the filter, click the arrow in the column header and select **Remove filter**.
- You can also enter add a filter to the toolbar using **Filter**. The  icon applies the filter to the key and the translations. To reset the filter, click the . The icon  can also be used to create and save additional filter queries with wildcards, full-text search or SQL queries.

When you double-click an entry in a translation field to unlock and highlight it, the selected text is automatically copied to **Filter** in the display.

6. Edit the translations.

- Double-click the input field to unlock it and enter the translation.
- Once you have checked a translation, set the translation to **Checked**.

TIP:

- Use **Ctrl + K** to transfer the key value to the translation.
- Use  to run a grammar check.
- Use the **Edit** context menu to:
 - Enter the data source of a key.
 - Add a comment.
 - Delete a translation for a language.

After the changes have been committed to the main database, the system data must be recalculated by the DBQueue Processor in order make the new multi-language data available to all system users.

Related topics

- [Showing usage of a translation on page 215](#)
- [Editing translations of a single table on page 216](#)
- [Changing translation keys on page 218](#)

Changing translation keys

Use this task to change the key of a translation (Entrykey) in the DialogMultiLanguage table and the source text of all objects that use this key, such as a column name or description.

IMPORTANT: Before changing a key, check in the Designer about how it is used.

To change a key

1. In the Designer, select the **Base data > Localization** category.
2. Start the Language Editor using **Edit translation in database**.
3. In the **Key** column for the required entry, select the **Edit key** context menu item.

Related topics

- [Showing usage of a translation](#) on page 215

Importing translations from the language pack


The default One Identity Manager installation is supplied in **English - United States [en-US]** and **German - Germany [de-DE]**. To translate the Web Portal there are other languages available. These are provided in the form of One Identity Manager Language Pack CSV files.

NOTE: You will find the One Identity Manager Language Pack in the Support Portal under <https://support.oneidentity.com/>.

The import:

- Creates the translations in the DialogMultiLanguage table.
- Updates currently existing entries based on the key, the table, and the column.
- Deletes the entries.

To import the language files

1. In the Designer, select the **Base data > Localization** category.
2. Start the Language Editor using the **Edit translation in database** task.
3. In the editor toolbar, click .
4. Select *.CSV files with the required language and click **Open**.

This starts the import. The process may take some time.

5. Commit the changes to the main database. Use the **Database > Save to database** menu item.

After the changes have been committed to the main database, the system data must be recalculated by the DBQueue Processor in order make the new multi-language data available to all system users.

Process orchestration in One Identity Manager

One Identity Manager uses so called 'processes' for mapping business processes. A process consists of process steps, which represent processing tasks and are joined by predecessor/successor relations. This functionality allows flexibility when linking actions and sequences to object events. Processes are modeled using process templates. A process generator (Jobgenerator) is responsible for converting script templates in processes and process steps into a concrete process in the 'Job queue'.

The One Identity Manager Service handles defined processes. The service has to be installed on the One Identity Manager network server to run the processes. The server must be declared as a Job server in the One Identity Manager database.

The One Identity Manager Service is the only One Identity Manager component authorized to make changes in the target system.

To monitor the process handling, use the Job Queue Info program. For more information, see the *One Identity Manager Process Monitoring and Troubleshooting Guide*.

Detailed information about this topic

- [Setting up Job servers](#) on page 261
- [Mapping processes in One Identity Manager](#) on page 220
- [The One Identity Manager Service functionality](#) on page 276

Mapping processes in One Identity Manager

One Identity Manager uses so called 'processes' for mapping business processes. A process consists of process steps, which represent processing tasks and are joined by predecessor/successor relations. This functionality allows flexibility when linking actions and sequences to object events.

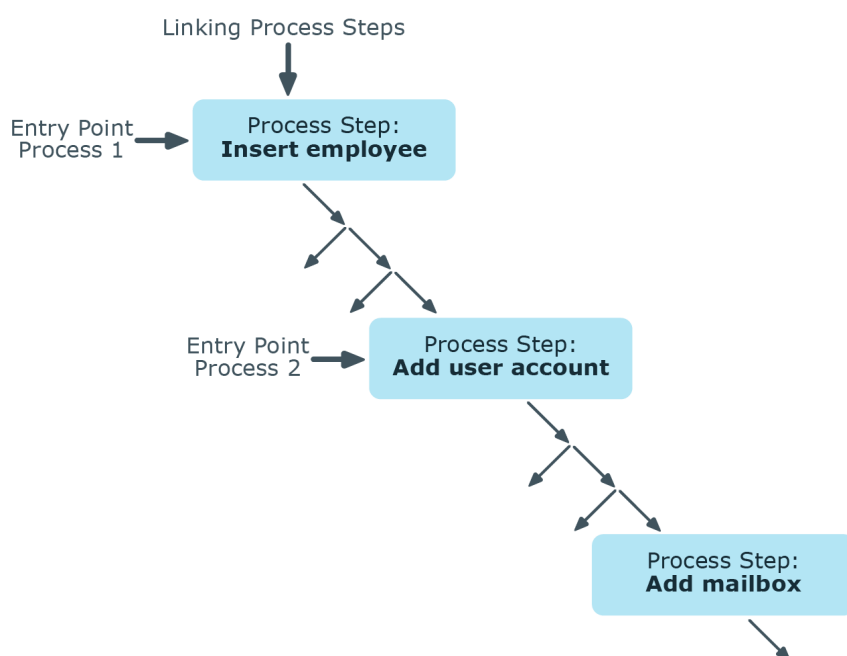
So-called process tasks are used to perform single elementary tasks at system level, for example, adding a directory. A process component consists of one or more process tasks and its parameters. Process components are defined in the tables Jobcomponent, Jobtask and Jobparameter along with their process tasks and parameters. Predefined configurations are maintained by the schema installation and cannot be edited apart from a few properties.

Processes are modeled using process templates. A process generator (Jobgenerator) is responsible for converting script templates in processes and process steps into a concrete process in the 'Job queue'.

The following illustration shows a chain of process steps with which you can add an employee, set up an Active Directory user account for him or her and finally add a mailbox.

You can reproduce this sequence in a process. However, you can also define entry points for other processes. The entry point of process 1 results in the creation of an employee with an Active Directory user account and mailbox. The entry point of process 2 only results in the creation of an Active Directory user account with a mailbox.

Figure 25: Creating a single process by linking process steps



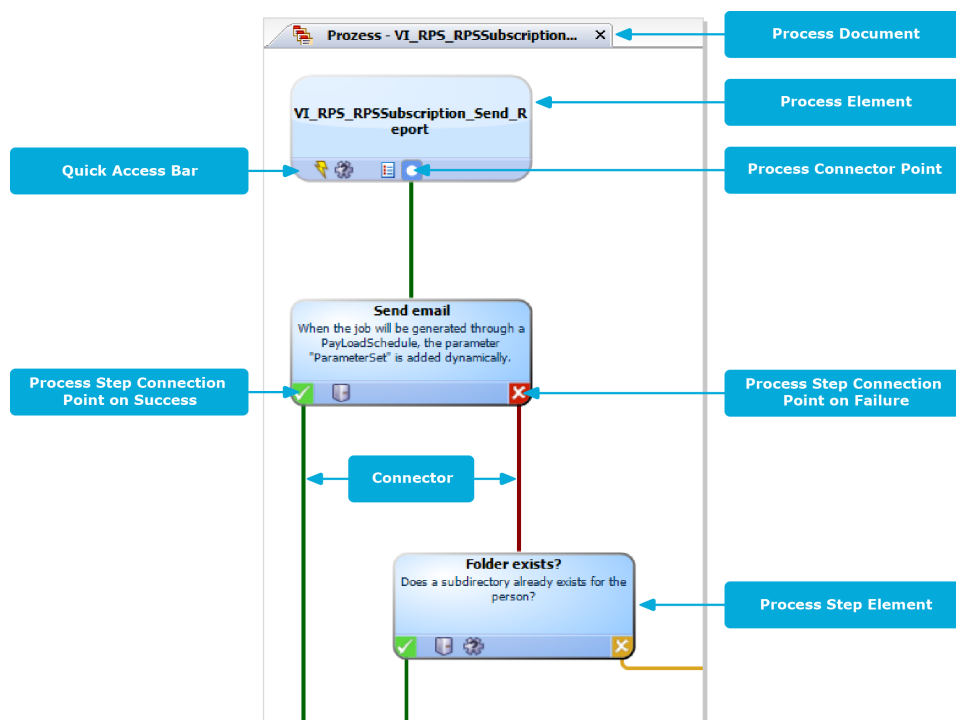
Related topics

- [Editing processes with the Process Editor](#) on page 222
- [Defining processes](#) on page 224
- [Running processes automatically](#) on page 251
- [Overview of process components](#) on page 254

Editing processes with the Process Editor

You can edit processes in the Designer using the Process Editor. In the Process Editor, a process is combined with its process steps in a process document. The process is displayed and controlled by means of special control elements.

Figure 26: Illustrating a process in the Process Editor



When you add a new process, an initial process document with one process element is created. When you add a process step, the associated process step element is created.

Individual elements are linked to each other with a connector. Activate the connection points with the mouse.

- To create a connection, click on a connection point, hold down the left mouse button and pull a connector to the second connection point.
- To delete a connection, select a connection end-point again by clicking with the mouse. Confirm the security prompt with **OK**.












Double-click on the process or process step element to open the respective edit view, where you can make your changes.

Each element has a tooltip. A process element's tooltip displays the name and description of the process. A process step element's tooltip displays the name and description of the process step as well as the description of the process task used.

Each element contains a quick access menu bar. The icons represent special properties of processes or process steps. The icon's tooltip shows more detailed information about a

property. Double-click on a icon to open the edit view of the process or process step and jump to the corresponding property.

Table 69: Quick access icons

Icon	Meaning
	Events are defined.
	Process is not generated.
	Process in wait mode on error.
	Processing is split. The connection point on error and the connector to the subsequent process step are colored yellow.
	Runtime errors are ignored. The connection point is colored gray on error. No process step is possible on error.
	If an error occurs, no more process steps are handled for this process.
	A generating condition exists.
	Process information is enabled.
	A script for selecting a server or server mask is entered.
	Messaging on error and on success is enabled.
	The process or process steps are customized. More information about the customizations is shown in a tooltip.

Some important properties are shown by the color of the element.

Table 70: Colors of elements

Color	Meaning
Blue	Default.
Yellow	The verification test resulted in a warning or information.
Red	The verification test failed.
Gray	The process is disabled.

You can drag and drop elements in the process document. Use **Arrange** in the context menu to reset the elements to their default positions. The position of each element is transferred to the One Identity Manager database when the entire process is saved. The layout is therefore available to all users when you restart the Designer.

Defining processes

IMPORTANT: The process and process steps are not created until the entire process is saved in the One Identity Manager database. After this, other users can use the Process Editor to make changes to the process. However, it cannot be generated yet. The process has to be compiled before it can be generated.

You can modify default processes to meet your requirements, if necessary. To add further process step to a process, create a custom process.

The following steps are required to set up a process

1. Create up a process.
2. Specify which events to trigger.
3. Create the process steps.
4. Edit the parameters.
5. Test the process.
6. Compiles the process.

Related topics

- [Editing processes with the Process Editor](#) on page 222
- [Creating and editing processes](#) on page 224
- [Copying processes](#) on page 225
- [Creating and editing process steps](#) on page 226
- [Copying single process steps](#) on page 227
- [Copying process steps within a process](#) on page 227
- [Searching for entries within processes](#) on page 228
- [Process step parameters](#) on page 234
- [Events for processes](#) on page 238
- [Permissions for triggering processes](#) on page 240
- [Simulating process generation](#) on page 241
- [Checking the validity of a process](#) on page 242
- [Compiling processes](#) on page 244

Creating and editing processes

To edit an existing process

1. In the Designer, select the process in the **Process Orchestration** category.
2. Start the Process Editor with the **Edit process** task.

The process is opened in the Process Editor.

To create a new process

1. In the Designer, select the **Process Orchestration** category.
2. Start Process Editor using the **Create a new process** task.

This makes a new element for the process and opens it in the Process Editor.

Related topics

- [Copying processes](#) on page 225
- [Process properties](#) on page 230
- [Comparing processes](#) on page 229
- [Exporting and importing processes](#) on page 229

Copying processes

To copy a process, a wizard is provided.

To copy a process

1. In the Designer, select the process in the **Process Orchestration** category.
2. Start the Process Editor with the **Edit process** task.
3. Select **Process > Copy** item from the menu.
4. Specify the copy options on the home page of the wizard.

Table 71: Copy options

Option	Meaning
Rename process steps	If you set this option, the wizard allows you to rename the individual process steps.
Copy events	Enable this option so that events assigned to this process are also copied.
Disable source process	Use this option to specify whether to disable the source process after it is copied. If you enable this option, the Do not generate option is enabled for the source process.
Disable copied process	Use this option to specify whether to disable the process after it is copied. If you enable this option, the Do not generate option is enabled for the copied process.

5. On the **Copy options** page, specify the name of the new process.

6. (Optional) On the **Define process step name** page, name the individual process steps.

You can change these by clicking on the new process step name.

NOTE: This step is only available if you have set the **Rename process steps** copy option.

7. To start the copying process, click **Finish** on the last page of the wizard.
The process is opened in Process Editor and can be further edited.

Related topics

- [Creating and editing processes](#) on page 224
- [Comparing processes](#) on page 229
- [Exporting and importing processes](#) on page 229

Creating and editing process steps


To create a new process step

1. In the Designer, select the process in the **Process Orchestration** category.
2. Start the Process Editor with the **Edit process** task.
3. Select the **Process step > New** menu item.
This makes a new element for the process step and displays it in the Process Editor.
4. In the **Process step properties** view, edit the main data of the process step.
5. Link the process step with the process.

To edit an existing process step

1. In the Designer, select the process in the **Process Orchestration** category.
2. Start the Process Editor with the **Edit process** task.
3. Click on the element for the process step in the process document.

NOTE: To edit several process steps, hold down the **Ctrl** key and click the process steps.

Input fields with entries that have different input are labeled with the  icon in the edit view for process steps. When an input field is edited and saved, the value is copied to all selected the process steps.

Related topics

- [Copying single process steps](#) on page 227
- [Copying process steps within a process](#) on page 227
- [Process step properties](#) on page 231

Copying single process steps

To copy a process step

1. In the Designer, select the process in the **Process Orchestration** category.
2. Start the Process Editor with the **Edit process** task.
3. Select the process step to copy and use **Copy** in the context menu or **Ctrl + C** to copy the process step to the clipboard,

NOTE: To copy several process steps, hold down the **Ctrl** key and click the process steps.


4. Insert the process step using **Paste** in the context menu or **Ctrl + V**.
The process step is given a new UID and all the process steps are copied.
5. Edit the process step's main data.
6. Link the process step with the process.

Related topics

- [Creating and editing process steps](#) on page 226
- [Copying process steps within a process](#) on page 227
- [Process step properties](#) on page 231

Copying process steps within a process

To import a process step



1. In the Designer, select the process in the **Process Orchestration** category.
2. Start the Process Editor with the **Edit process** task.
3. Select **Process step > Import** from the menu.
4. In the edit view, select **Search and import process steps**.
5. Enter the search text in the input field.
6. Use  to specify the search options in which objects are to be searched.

The specified objects are searched for internally by a WHERE clause. If several objects are specified, they are appended internally with JOIN conditions.

Table 72: Searchable objects and properties

Find options	Properties to be Searched
Process	Name

Find options	Properties to be Searched
Process step	Name, description, generating condition, server selection script
Parameter	Name, value
Process components	Component class, component assembly
Process task	Name
Parameter template	Name, value template

7. To starting searching, click .
- The process steps that are found are displayed in the result list.
8. In the result list of the search, select the required process step and click .
- The process step is imported into the process document.
9. Edit the process step's main data.
10. Link the process step with the process.

Searching for entries within processes

To search for an entry within a process

1. In the Designer, select the process in the **Process Orchestration** category.
2. Start the Process Editor with the **Edit process** task.
3. Open the search dialog box using **Ctrl + F**.
4. Enter the search text under **Text**.
5. Start the search using the **Search** button.
6. Use **F3** to search next.




This searches for the text in the process and process steps.

Table 73: Objects and properties to be searched

Search in Objects	Properties to be Searched
Process	Name
Process step	Name, description, generating condition, server selection script

Comparing processes

To find differences between two processes

1. In the Designer, select the process in the **Process Orchestration** category.
 2. Start the Process Editor with the **Edit process** task.
 3. Select the **Process > Compare processes** menu item.
The current process is already selected as **Process A**.
 4. Select the process to compare it with in the **Process B** menu.
 5. (Optional) use the  button to specify which process properties you want to include in the comparison. By default, all the properties of the processes, process steps and events are compared.
 6. Start the comparison with .
- Differences in the processes are highlighted in the output text.
- | **TIP:** Mark the text and click the  button to copy the text to the clipboard.

Exporting and importing processes

Exporting and importing processes is implemented through XML files.

To export a process to an XML file

1. In the Designer, select the process in the **Process Orchestration** category.
2. Start the Process Editor with the **Edit process** task.
3. Select the **Process > Export** menu item.
4. Enter the file name and click **Save**.

To import a process from an XML file

1. In the Designer, select the **Process Orchestration** category.
2. Start the import with the **Import process** task.
3. Select the XML file and click **Next**.
The process is opened in the Process Editor.

Related topics

- [Creating and editing processes](#) on page 224

Process properties

Table 74: Properties of a process

Property	Meaning
Name	Name of the process. The name of the process must be unique. Label custom processes with the CCC_ prefix.
Table	The process is generated on the event from this base object (table).
Description	Additional description of the process.
Remarks	Additional remarks about the process.
Process	Process UID. These cannot be edited.
Process information	<p>Specifies whether this process is logged. Logging is performed depending on the Common ProcessState ProgressView configuration parameter.</p> <p>Permitted values are:</p> <ul style="list-style-type: none">• None: The process information is not logged.• Full process tracking: The process information is recorded and displayed in the Manager.• Web Portal tracking: The process information is logged and displayed in the Manager and in the Web Portal.
Process information term	VB.Net expression for displaying the display name in the process view.
Pre-script for generating	<p>The pre-script is run before other scripts are run. You can find global variables with a pre-script or define process specific variables that can then be used within the process and process steps, for example, in generating conditions, sever selection scripts or parameters.</p> <p>NOTE: When a process is being handled, the generating pre-script is run first and then the generating condition is evaluated.</p>
Generating condition	<p>Define a condition in VB.Net syntax for the process step, which is used to decide whether the process is generated. If a generating condition is given, the process is only generated if the condition is fulfilled.</p> <p>You can find an example scripts on the installation medium in QBM\dvd\AddOn\SDK\ScriptSamples.</p>
Do not generate	<p>Use this option to decide whether a process will be generated. If the option is set, the process will not be generated and cannot be compiled.</p> <p>NOTE: If the option for processes is activated, this option also remains activated during a schema update and is not reset.</p>
Preprocessor	You can specify a preprocessor condition for a process for conditional

Property	Meaning
condition	compiling. A process is only available, therefore, if the preprocessor condition is fulfilled.
Disabled by preprocessor	If a process step is disabled by a preprocessor condition, the option is set by the Database Compiler.
Threshold (warning)	Maximum number of processes for a queue that can be present at the same time. A warning is sent if the number is exceeded. The One Identity Manager Service continues handling processes all the same.
Threshold (disable)	Maximum number of processes for a queue that can be present at the same time. If this number is exceeded, other processes are set to the Overlimit status and are not processed by the One Identity Manager Service.


Related topics

- [Using process-specific and global variables for the process definition](#) on page 245
- [Thresholds for handling processes](#) on page 247
- [Logging process information during process handling](#) on page 320
- [Conditional compilation using preprocessor conditions](#) on page 335
- [Visual Basic .NET scripts usage](#) on page 341

Process step properties

Table 75: General process step properties

Property	Meaning
Name	Name of the process step.
Process task	Process task to run for the process component. When you select a process task you define which action is run by the process step. The process task parameter templates are copied to the process step as parameters. This means that every process step that uses this process task can pass other parameter values. The original is not altered.
Description	Additional description of a process step.
Priority	The priority sets the precedence in the Job queue for adding and processing the process step. The values 1 to 15 are allowed. The higher the value, the sooner the process step will be processed.
Priority definition	VB.Net expression for determining the priority depending on the contents of the process. If a process step contains a script for dynamically determining the

Property	Meaning
	<p>priority, the script is used. Otherwise, a predefined priority is set.</p> <p>Example:</p> <p>Password changes to a user account should be run with a higher priority (in the example 7), changes to other main data with priority 3.</p> <pre>If \$UserPassword[o]\$ <> \$UserPassword\$ Then Value = 7 Else Value = 3</pre> <p>NOTE: The field is not visible when you open the process step in the Process Editor. Click  next to Priority to show the field.</p>
Process information	<p>Specifies whether this process is logged. Logging is performed depending on the Common ProcessState ProgressView configuration parameter.</p> <p>Permitted values are:</p> <ul style="list-style-type: none"> • None: The process information is not logged. • Full process tracking: The process information is recorded and displayed in the Manager. • Web Portal tracking: The process information is logged and displayed in the Manager and in the Web Portal.
Process information term	VB.Net expression for displaying the display name in the process view.
Depth of detail	Severity level for mapping process information.
Notification (success)	Specifies whether notification is sent on success.
Notification (error)	Specifies whether notification is sent on error.
Pre-script for generating	The pre-script is run before other scripts are run. You can find global variables with a pre-script or define process specific variables that can then be used within the process, for example, in generating conditions, sever selection scripts or parameters.
Generating condition	Define a condition in VB.Net syntax for the process step, which is used to decide whether the process step is generated. If a generating condition is given, the process step is only generated if the condition is fulfilled.
Preprocessor condition	You can specify a preprocessor condition for a process step for conditional compiling. A process step is, therefore, only available if the preprocessor condition is fulfilled.

Property	Meaning
Disabled by preprocessor	If a process step is disabled by a preprocessor condition, the option is set by the Database Compiler.
Server function	Specifies the server types for this process step. Specifies the permitted server types for this process step. The selection must lead to a unique result, for example SQL processing Server.
Script for server selection	If it is not possible for the Job Generator to decide which server to use based on the server function, you can use a selection script in VB.net syntax for more a detailed evaluation.
Wait mode on error	If a specific condition is not fulfilled at a particular point in the process step, One Identity Manager Service can repeat the process step. Setting this option results in the process step being re-run depending on latency and retries.
Latency (mins)	Latency period in minutes. Number of minutes a process step, if it has failed, is deferred until the next retry.
Retries	Number of retries.
Split processing	Process steps that are only required for branching the process are labeled with this option. An example could be a process step that checks for the existence of a directory. Depending on the result returned, the next step to be processed is either the next step on success or the next step on error, without generating an error message.
Ignore errors	Specifies whether runtime errors are ignored. In this case the following process step is still carried out despite the previous step not being correctly processed.
Stop on error	<p>If an error occurs when a process step is processed, the process step remains in the job queue and is given the Frozen status. In this case, no more process steps are collected for processing and they remain in the Job queue. You can re-enable the process steps that have the Frozen status in Job Queue Info program. For more information, see the <i>One Identity Manager Process Monitoring and Troubleshooting Guide</i>.</p> <p>If the Common MailNotification NotifyAboutWaitingJobs configuration parameter is enabled, an email notification sent is sent in addition if processes with the Frozen status occur, and a corresponding entry is generated in the event log of the update server. Prerequisites for using the notification system is an SMTP host set up for sending mail and activation of the configuration parameter for mail notification.</p> <p>Process steps that are generated by SQL from the database, can always be labeled with the Stop on error option. You can configure this behavior in the Common DBJobCreateWithFreeze configuration parameter.</p>

Property	Meaning
Log errors to journal	If this option is set, the error message from process handling is logged to the system journal. Error messages from process handling can be recorded in the process history.
Log mode	<p>You can enable an extended logging mode for process step messages in Job Queue Info.</p> <p>Use this logging mode to provide individual processing steps with continuous extended logging. Use the Always value to log the messages of the process step on success and on failure. Use the value Error to log the messages of the process step on failure only.</p>
Process History	Specifies whether process step notification is written to the process history.
DBQueue does not wait	Specifies whether or not to wait until the process step has been processed before continuing to process DBQueue Processor tasks. It is only necessary to wait for process steps if a process step could change data that is relevant to the DBQueue Processor tasks.

Related topics

- [Checking the validity of a process](#) on page 242
- [Specifying the executing server](#) on page 248
- [Notifications about process step handling](#) on page 249
- [Using process-specific and global variables for the process definition](#) on page 245
- [Overview of process components](#) on page 254
- [Logging process information during process handling](#) on page 320
- [Conditional compilation using preprocessor conditions](#) on page 335
- [Visual Basic .NET scripts usage](#) on page 341

Process step parameters

When you select a process task you specify which action will be run by the process step. The process task parameter templates are copied to the process step as parameters. This means that every process step that uses this process task can pass other parameter values. The original is not altered.

Compulsory parameters are immediately entered into the process step when the process task is selected. Then, you can enter any optional parameters individually. When a parameter is added, the value template is copied from the parameter template. Templates for parameter values are mostly predefined, for example, procedures that evaluate object UIDs and note them accordingly.

Detailed information about this topic




- [Editing process step parameters](#) on page 235
- [Properties of process step parameters](#) on page 235
- [Allocating parameter values](#) on page 236

Editing process step parameters

To edit process step parameters

1. In the Designer, select the process in the **Process Orchestration** category.
2. Start the Process Editor with the **Edit process** task.
3. Click on the element for the process step in the process document.
4. Select the **Parameter** view.
This displays all the parameters defined for the process.
5. Check whether the required parameters are assigned and edit the parameters.
You can add, delete, or edit parameters from the toolbar.
TIP: Click an entry to edit the parameter value directly.

Table 76: Meaning of icon used

Icon	Meaning
	Mandatory process task parameter
	Optional process task parameter, which is assigned to a process step.
	Optional process task parameter, which is not assigned to a process step.


Related topics

- [Editing process step parameters](#) on page 235
- [Allocating parameter values](#) on page 236

Properties of process step parameters

Table 77: Properties for parameters

Property	Meaning
Name	Name of the parameter. NOTE: You should not change the name of a parameter. The special parameters of the HandleObjectComponent process component are an exception to this rule.

Property	Meaning
Hidden	<p>Specifies whether the parameter is shown in the One Identity Manager Service log file and in the Job Queue Info program. Values for hidden parameters are shown as <HIDDEN>.</p> <p>NOTE: Users with the program function Option to see the values of hidden parameters in Job Queue Info (JobQueue_ShowHiddenParameters) can view the hidden parameters in the Job Queue Info. Assign the appropriate permissions group to the program function.</p>
Encrypted	<p>Specifies whether the parameter is encrypted when it is passed if the database is encrypted. Encrypted parameters are shown as <hidden> in the One Identity Manager Service log file and in the Job Queue Info program.</p> <p>NOTE: If the Encrypted option is already set in the parameter template, the parameter must also be encrypted when it is passed.</p>
Contains encrypted components	<p>Specifies whether encrypted sequences are contained in this value. Use this option, if partially encrypted sequences such as passwords are to be passed in complex parameters, for example Windows PowerShell scripts. Encrypted parts of a parameter are shown as <Hidden> in the One Identity Manager Service log file and in the Job Queue Info program.</p>
Value template	<p>Define value templates in VB.Net syntax. When a parameter is added, the value template is copied from the parameter template.</p> <p>TIP: To restore the default value template, select the  button in View > Parameter and click the Template button in the Edit parameters view.</p>
Type	<p>Type of parameter. The IN, OUT and INOUT values are permitted.</p> <p>Parameters of the OUT or INOUT type are parameters that a process component can use to output a value. This value is then available in all subsequent process steps in the process and can be used as a value for parameters of the IN type.</p>

Related topics

- [Allocating parameter values](#) on page 236
- [Visual Basic .NET scripts usage](#) on page 341

Allocating parameter values

Define value templates in VB.Net syntax. The following statements can be used for allocating values:

- None
- Columns of an object or columns of an object connected by a relation

Syntax:

Value = \$<column name>:<data type>\$

Value = \${FK(<foreign key column>).}column name:<data type>\$

Example:

Value = \$Lastname\$

Value = \$PasswordNeverExpires:bool\$

Value = \$FK(Ident_Domain).Description\$

- Parameter from the optional parameter collection

Syntax:

Value = \$PC(<parameter name>)\$

Example:

Value = \$PC(SRCUID_Application)\$

- Out-Parameter

Parameters of the **OUT** or **INOUT** type are parameters that a process component can use to output a value. This value is then available in all subsequent process steps in the process and can be used as a value for parameters of the **IN** type.

When you use **OUT** parameters, you need to ensure that they contain data at runtime. Alternatively, when the text is processed "&OUT(<parameter name>)" is entered, which means that the variable will not be replaced.

Syntax:

Value = "&OUT(<parameter name>)"

Example:

Value = "&Out(FileSize)"

- Global variables allocated by the set-up program

Syntax:

Value = Variables("<variable name>")

Example:

Value = Variables("GENPROCID")

Value = Variables("FULLSYNC")

- The local variables of the process step or of the process generated by the pre-script

Syntax:

Value = values("Name")

Example:

Value = Values("FirstHomeServer")

- Querying configuration parameters

The full path for the configuration parameter must always be entered.

Syntax:

```
Value = Session.Config().GetConfigParm("<full path>")
```

Example:

```
Value = Session.Config().GetConfigParm("TargetSystem | ADS |  
PersonAutoDefault")
```

- VB.Net

Enter any statements in VB.NET syntax.

- Querying environment variables

Syntax:

```
&ENV(Variablename)&
```

Example:

```
Value = "&ENV(COMPUTERNAME)&"
```

- Querying secrets

In the One Identity Manager Service configuration, the SecretAllowList and SecretsFolder parameters must be configured.

Syntax:

```
&SECRET(Name)&
```

Example:

```
Value = "&SECRET(API_KEY)&"
```

Related topics

- [Properties of process step parameters](#) on page 235
- [Visual Basic .NET scripts usage](#) on page 341
- [Configuring the One Identity Manager Service](#) on page 279

Events for processes

Events are defined to assign processes to objects. Processes cannot be generated until a link has been created between object, event, and process. The following predefined events are available. These are described in the following table.

Table 78: Predefined events

Event	Comment
Insert	Event created when an object is created. Available for all objects.
Update	Event created when an object is changed. Available for all objects.

Event	Comment
Delete	Event created when an object is deleted. Available for all objects.
Execute	The event is triggered by the DBQueue Processor when the activation time of a deferred operation is reached.
Assign	The event is triggered when many-to-many assignments are added.
Remove	The event is triggered when many-to-many assignments are removed.

Other events are provided by the Customizer. You can define other custom events to trigger processes.

Detailed information about this topic

- [Creating events for processes](#) on page 239
- [Permissions for triggering processes](#) on page 240

Creating events for processes

If the object event is assigned a program function, users that own this program function by permissions group, can trigger the object event and therefore the process, irrespective of their permissions. Detailed information about managing permissions and running processes with program functions can be found in the *One Identity Manager Authorization and Authentication Guide*.

To create an event



1. In the Designer, select the process in the **Process Orchestration** category.
2. Start the Process Editor with the **Edit process** task.
3. Click on the element for the process in the process document.
4. Select the **Events** view and click .
5. Enter the following information.

Table 79: Event properties

Property	Description
Object event	<p>Name of the object event.</p> <p>The Object event menu displays the object events of the table specified in the process.</p> <ol style="list-style-type: none"> 1. Select an existing object event. - OR - 2. Click  and enter the name of the new object event.

Property	Description
Sort order	Specifies the sort order in which the processes are generated if multiple processes refer to the same event of the base object. Processes with a lower sort order are generated before processes with a higher sort order.
Event process information	VB.Net expression for displaying the display name in the process view.

6. (Optional) Assign a program function to the object event.
 - a. In the Designer, select the event in the **Process Orchestration > Object events**.
 - b. Select the **View > Select table relations** menu item and enable the QBMEventHasFeature table.
 - c. In the edit view, select the **Program function** view and select the program function.

Related topics

- [Logging process information during process handling](#) on page 320
- [Permissions for triggering processes](#) on page 240

Permissions for triggering processes

The basic permissions for triggering processes are granted to the logged in user by the **Allow to trigger any events from the frontend** program feature (Common_TriggerEvents).

In One Identity Manager, triggering of events on stored processes is linked to the permissions concept. Users can only trigger events on objects like this if they own edit permissions for them. This can lead to table users who only have viewing permissions not being able to trigger additional events for processes.

In this case, it is possible to connect the object events (QBMEvent table) with a program function (QBFeature table). An event (JobEventGen table), which is defined for a process, is linked with an object event (JobEventGen.UID_QBMEvent column). The object events are linked to a program function (QBEventHasFeature table). Users with this program function can trigger the object event and therefore the process too independent of their permissions.

Detailed information about managing permissions and running processes with program functions can be found in the *One Identity Manager Authorization and Authentication Guide*.

Related topics

- [Creating events for processes](#) on page 239

Simulating process generation

You can use simulation to test whether a selected process can be generated successfully or whether the syntax for passing parameters is correct. This makes it easier to alter processes if necessary.

NOTE:

- The **Do not generate** option is taken into account when you simulate a process. Disable this option for process simulation.
- The assemblies generated are saved locally on the workstation on which the simulation is run. A simulation does not, therefore, have any effect on other users.

To generate a process for testing

1. In the Designer, select the process in the **Process Orchestration** category.
2. Start the Process Editor with the **Edit process** task.
3. Start the simulation wizard with the **Process > View > Simulation view** menu item.
4. On the start page of the wizard, click **Next**.
5. On the **Select event** page, select the event for which the process is to be generated and specify the database connection for the simulation. Select **Designer Database** or **Main database**.
6. On the **Select object** page, select the object for which the event is to be simulated.
7. (Optional) On the **Change object properties** page, change the object properties.
8. (Optional) On the **Define parameter list** page, enter the parameters for the parameter collection. You can run the following actions:
 - **Load process steps:** For the selected process, all elements of the parameter collections for all process steps are loaded.
 - **Insert:** Insert individual parameters for the simulation. Enter the parameter name and parameter value.
 - **Delete:** Delete individual parameters for the simulation.

NOTE: For processes generated with parameter collections, you must specify the parameters and the values to be passed (for example, the `SourceDir` parameter for copies of profiles or the `ConfigName` parameter for loading a target system). No parameter collection is used for processes generated for the default events (insert, update, delete).

9. (Optional) On the **Define connection variables** page, specify the session object global variables to use for the simulation. Click **Insert** and enter the variable name

and the value.

10. (Optional) On the **Define preprocessor directives** page, select the preprocessor conditions to be taken into account when the process is generated.
11. To start the simulation, click **Finish** on the last page of the wizard.

The simulation process can take some time. After the simulation is complete the generated process is shown in the Process Editor. The process steps are shown in color depending on the generation result.

Table 80: Simulation color code


Color	Meaning
Light gray	Process step not generated.
Blue	Process step successfully generated.

NOTE:

Double-click on a successfully generated process step in the edit window to display properties and parameters with their specific values.

You can copy parameters values to the clipboard with **Ctrl + C**.

TIP:

- You can swap between the edit view and the simulation view using the **Process > View** menu to post-process the processes.
- For every simulation, an entry is created in the  toolbar of the Process Editor, which you can use to rerun the simulation without having to specify the simulation data again.
- To display the process generation protocol, select the **Process > View > Process generator log** menu item.

Checking the validity of a process




NOTE: Before you compile a process, you should carry out a validity check of the process and process steps.

To check a process

1. In the Designer, select the process in the **Process Orchestration** category.
2. Start the Process Editor with the **Edit process** task.
3. Select the **Process > Error checking** menu item.

The result of the check is displayed in the **Validity check** view and is retained until the next check.

Table 81: Icons used in the validity check

Icon	Meaning
	No errors found.
	Errors.
	Warning, Information.

TIP:

- Process or process step controls are highlighted in yellow to indicate a warning or information. If errors occur, the process or process step controls are highlighted in red.
- Double-click an error message in the **Validity check** view to jump to the corresponding entry in the process.

Table 82: Possible reasons for process failure

Error category	Possible cause
Errors	<p>The process does not have a name.</p> <p>No base object given.</p> <p>The generating condition does not correspond to required notation (value =).</p>
Warning	<p>The process does not have a base process step.</p> <p>The process has no event.</p>
Information	The option Do not generate is set.

Table 83: Possible reasons for process step failure

Error Category	Possible Cause
Errors	<p>The process step does not have a name.</p> <p>No process task assigned.</p> <p>The given generating condition does not correspond to required notation (value =).</p> <p>No executing server specified (server selection script or server mask).</p> <p>Process step name not unique.</p> <p>Process step has no parameters.</p> <p>The given parameter value does not correspond to required notation (value =).</p> <p>The two process step properties Ignore errors and Wait mode on error</p>

Error Category	Possible Cause
	<p>are set. The process step is never repeated because all errors are ignored.</p> <p>The two process step properties Ignore errors and Stop on error are set. The process step never goes into a FROZEN state because all errors are ignored.</p> <p>The two process step properties Ignore errors and Split processing are set. The split error is never processed.</p>
Warning	<p>Process step not linked into the process.</p> <p>The two process step properties Split processing and Wait mode on error are set and no retries are defined. If errors occur, the process step is stopped immediately and therefore processing does not split off.</p>

Related topics

- [Compiling processes](#) on page 244

Compiling processes

Once you have created, imported, or made changes to a process, you need to compile it. The process cannot be generated until it has been compiled.

NOTE: Before you compile a process, you should carry out a validity check of the process and process steps.

Compiling takes place for each base object, that means that all processes that belong to a base object are translated. The assemblies are created and placed on the workstation where generating will take place. During translation, the source is checked for errors. This process may required some time.

There are two methods for compiling a process in the Process Editor:

Local compiling

Use this method to compile a process for testing.

To compile a process for testing

1. In the Designer, select the process in the **Process Orchestration** category.
2. Start the Process Editor with the **Edit process** task.
3. Select **Process > Compile** from the menu.

Compiling and saving assemblies to the main database

If the process has been test compiled, use this method to add assemblies that are generated into the main database after compiling the process. Once the changes have be

integrated the altered processes are immediately available in the system.

To compile a process and save the assemblies to the main database

1. In the Designer, select the process in the **Process Orchestration** category.
2. Start the Process Editor with the **Edit process** task.
3. Select **Process > Compile and save to database** from the menu.

Displaying errors

Error messages during compiling are displayed in **Compiler errors**. The source code is displayed if errors occur during compilation. This view is only for displaying the source code. It cannot be edited here.

NOTE:

- If several users edit processes of the same base object, any error messages are also sent to other users. However, these cannot be changed by the current user.
- Double-clicking the error message in the **Compiler errors** view takes you straight to the corresponding line in the process. Here, you can edit it.
- Double-clicking the error message in the **Compiler errors** view takes you straight to the corresponding row if the source code view is activated.

Related topics

- [Checking the validity of a process](#) on page 242

Using process-specific and global variables for the process definition

Process specific variables are local data spaces when a process is generated. They are used for determining values on a once-off basis within a pre-script, which can then be made further use of within the processes and their processes steps, for example, in generating conditions, server selection scripts or in the parameters.

NOTE: It is recommended only to set process specific variables in the pre-script and to have read access to them during further usage.

Pre-script syntax

```
values("Name") = "value"
```

Usage in the process and process step code sections

```
Value = values("Name")
```

Process generation can be controlled using additional global variables, which are made available through the Session object. These variables are valid as long as the session is active. All environment variable and custom variables defined for the Session object can be used in addition to predefined variables. Custom global variables can be defined through scripts, methods, or customizers, for example, and used in the processes.

NOTE:

- Global variables should only be used with read access in processes.
- When a process is being handled, the generating pre-script is run first and then the generating condition is evaluated. It is recommended to evaluate global variables that are used in the generating condition in the pre-script as well. This can prevent unnecessary data access.

If a custom session variable is defined, it must be removed again afterward. Otherwise it remains for the rest of the session and, in certain circumstances, the wrong processes can be generated.

Example:

The process should only be generated for a full synchronization. The Session variable **FullSync** is used for this. This variable can take the **True** and **False** values. The variable is available to all processes that are generated within full synchronization.

The variable is queried in the pre-script for generating and the generating condition. This way, loading of unnecessary objects is already prevented by running the pre-script.

Generating pre-script:

```
If CBool(Session.Variables("FULLSYNC")) Then
```

```
    values("Name1") = "value1"
```

```
    values("Name2") = "value2"
```

```
    ...
```

```
End If
```

Generating condition:

```
Value = CBool(Session.Variables("FULLSYNC"))
```

Related topics

- [Pre-scripts for use in processes and process steps](#) on page 351
- [Querying session object global variables](#) on page 353

Thresholds for handling processes

In order to prevent bulk modifications, you can specify how long each process can remain in the Job queue.

Prerequisites

- If the warning threshold is exceeded, a message is sent by email to a specified recipient. The prerequisites for using the notification system are a SMTP host set up for sending mail and the activation of the configuration parameter for mail notification. For detailed information about configuring the email notification, see the *One Identity Manager Installation Guide*.
- In the Designer, check the **Common | MailNotification | NotifyAboutWaitingJobs** configuration parameter and enable this configuration parameter if necessary. If the configuration parameter is enabled, an email notification is sent if processes with the **Overlimit** status occur and a corresponding entry is created in the update server's event log.

To define thresholds

1. In the Designer, select the process in the **Process Orchestration** category.
2. Start the Process Editor with the **Edit process** task.
3. Click on the element for the process in the process document.
4. In the **Process properties** view on the in the **General** tab, edit the following information.
 - **Threshold value (warning):** Enter the maximum number of these processes for a queue that can be present at the same time. A warning is sent if the number is exceeded. The One Identity Manager Service continues handling processes all the same.
 - **Threshold value (disable):** Enter the maximum number of these processes for a queue that can be present at the same time. If the disable threshold is exceeded, the affected processes in the Job queue are set to the **Overlimit** status. These processes are no longer collected by the One Identity Manager Service for processing and remain in the Job queue.

You can re-enable these processes in the Job Queue Info. For more information, see the *One Identity Manager Process Monitoring and Troubleshooting Guide*.

TIP: You can use the `SDK_SetLimitationCount_in_Jobchain` database script to initially fill the threshold for the lock. You can find an example of a configuration file on the installation medium in directory `QBM\dvd\AddOn\SDK\SQLSamples`.

Related topics

- [Process properties](#) on page 230

Specifying the executing server

You specify which server should handle each process step. You can select the executing server using the server function or a selection script. Server selection should always end with a unique result. The selection script is evaluated first to determine the server. If a server cannot be determined in this way, the server function is analyzed. The first server that is found is used for running the process step.

Detailed information about this topic

- [Selecting servers with server functions](#) on page 248
- [Selecting servers with selection scripts](#) on page 248

Selecting servers with server functions

The most common server functions are predefined, for example, domain controller or SQL processing server. Enter a server function directly if you can determine the server uniquely.

To specify a server using a server function

1. In the Designer, select the process in the **Process Orchestration** category.
2. Start the Process Editor with the **Edit process** task.
3. Click on the element for the process step in the process document.
4. In the **Process step properties** view, on the **Generation** tab in the **Server function** menu, select the server function.

Related topics

- [Overview of server functions](#) on page 266
- [Process step properties](#) on page 231

Selecting servers with selection scripts

If it is not possible to decide which server should be used based on the server function (for example, because several SMTP servers exist), you can use a server script for more a detailed evaluation.

To find the server with a selection script, use a VB.Net expression, which:

- Returns a string with the Job server UID
- Returns a string with data for a WHERE clause for database queries. The selection must return a string, which begins with WHERE and contains a logical condition. The WHERE clause is applied to the QBMServer table.

Alternatively, you can enter the queue to be handled by the process step directly into the selection script. Each One Identity Manager Service within the network has a unique queue name. Only process steps that have this exact queue name are requested from the Job queue.

Syntax for direct queue input:

DIRECT:<queue>

Example:

Value = "DIRECT:\Server01"

To specify a server using a selection script

1. In the Designer, select the process in the **Process Orchestration** category.
2. Start the Process Editor with the **Edit process** task.
3. Click on the element for the process step in the process document.
4. In the **Process step properties** view on the **Generation** tab in the **Script for server selection** property, enter the selection script.

Related topics

- [Visual Basic .NET scripts usage](#) on page 341
- [Selecting servers with server functions](#) on page 248
- [Process step properties](#) on page 231

Notifications about process step handling

You have the possibility to send a message when a process step has succeeded or when it has failed. Prerequisite for using the notification system is an SMTP host, set up for sending mail and activation of the configuration parameter for mail notification. Use the various configuration parameters for mail notifications for setting up notifications. For detailed information about configuring the email notification, see the *One Identity Manager Installation Guide*.

To configure mail notification for a process step

1. In the Designer, select the process in the **Process Orchestration** category.
2. Start the Process Editor with the **Edit process** task.
3. Click on the element for the process step in the process document.
4. Select the **Process step properties** view.

5. On the **General** tab, enable the **Notification (success)** and **Notification (error)** options.
6. Enter the data for sending notifications on the **Notification on success** and **Notification on error** tabs.

NOTE: You must enter all data in VB.NET syntax. Use #LD notation for language-dependent formatting of the information.

Table 84: Properties for notifications

Property	Meaning
Sender email address	Email address of the notification sender.
Recipient email address	Email address of the notification recipient.
Subject	Subject line.
Message	The message to be sent.

NOTE: Email notifications are only sent during processing if all the data is entered for a case (failure or success).

Example for configuring an email message

Sender email address	Value = Connection.GetConfigParm ("Common\MailNotification\DefaultSender")
Recipient email address	Value = Connection.GetConfigParm ("Common\MailNotification\DefaultAddress")
Subject	Value = #LD("Error updating the Active Directory user account {0}.", \$CanonicalName\$)#
Message	Value = #LD("The user account {0} could not be updated.)#

The process VID_SendMail (DialogDatabase table) is used to send email notifications from the process handling. This process uses the parameters of the vid_InsertForSendMail database procedure. To customize this process, create a copy of the process and edit it.

TIP:

To send the error messages logged by the One Identity Manager Service in case of an error by email notification, the vid_InsertForSendMail database procedure supplies the pcAdditionalMessage parameter.

To access this functionality, use the variable [AdditionalMessage] when you set up your failure notification message.

Example of a message:

```
Value = "Process failed." & vbCrLf _
```

```
& vbCrLf _  
& "-----"  
" & vbCrLf _  
& "[AdditionalMessage]"
```

Related topics

- [Visual Basic .NET scripts usage](#) on page 341
- [Using #LD-notation](#) on page 354
- [Process step properties](#) on page 231

Running processes automatically

Set up process plans to run cyclical processes to put into effect, for example, regular synchronization with a target system environment. Process plans are connected to schedules and can therefore be run at regular intervals.

The following steps are necessary to run processes automatically:

1. Creating a process plan
A process plan contains the basic configuration for automatically running a process.
2. Setting up and configuring a schedule
A schedule includes the configuration of activation times for running processes regularly. For detailed information about schedules, see the *One Identity Manager Operational Guide*.

Detailed information about this topic





- [Displaying process plan status](#) on page 251
- [Starting a process plan immediately](#) on page 252
- [Creating and editing process plans](#) on page 252
- [Process plan properties](#) on page 253

Displaying process plan status

To display the status of process plans

1. In the Designer, select the **Process Orchestration > Process automation** category.
2. Start the editor using the **Edit process plans** task.
The list view of the process plan editor shows all process plans with their status.

Table 85: Meaning of list view icons

Icon	Meaning
	The process plan schedule is not enabled.
	The process plan was run according to plan.
	The process plan was not run. This state can occur if the task could not be run to plan or if the schedule was re-enabled and the time had not been reached for the initial run.
	The current time (server time) does not lie between the start and end times of the schedule.

TIP: To reload the start times of all process plans, use the **Process plan > Refresh** menu item.

Related topics

- [Starting a process plan immediately](#) on page 252
- [Creating and editing process plans](#) on page 252

Starting a process plan immediately

Process plans are connected to schedules and can therefore be run at regular intervals. If necessary, you can start a process plan immediately.

To start a process plan immediately

1. In the Designer, select the **Process Orchestration > Process automation** category.
2. Start the editor using the **Edit process plans** task.
3. Select the process plan and select the **Run** context menu.

Related topics

- [Displaying process plan status](#) on page 251
- [Creating and editing process plans](#) on page 252

Creating and editing process plans

A process plan contains the basic configuration for automatically running a process.

To create or edit a process plan

1. In the Designer, select the **Process Orchestration > Process automation** category.
2. Start the editor using the **Edit process plans** task.
3. Use the **Process plan > New** menu item to create a new process plan.
- OR -
Select an existing process plan.
4. In the **Configure process plan** view, edit the process plan main data.



TIP: You can see which process is triggered by a process plan from the **Edit process** context menu.

Related topics

- [Displaying process plan status](#) on page 251
- [Starting a process plan immediately](#) on page 252
- [Process plan properties](#) on page 253

Process plan properties

Table 86: Process plan properties

Property	Meaning
Name	Name of the process plan. Translate the given text using the  button.
Table	Base object (table) for which the process plan will run.
Event	Event to be run. All base object events are listed for new process plans.
Activation schedule	Schedule that contains the activation times for the process plan. NOTE: Create a new schedule using  next to the menu. For detailed information about schedules, see the <i>One Identity Manager Operational Guide</i> .
Max. processing time	Enter the number of hours after which the process plan should automatically quit.
Description	Enter a detailed description of the process plan.
Condition	Limiting condition for elements to which the scheduled task will be applied. The input must satisfy the WHERE clause database query syntax.
Parameters	List of parameters of a parameter collection that are set when the process is generated from this process plan.

Related topics

- [Displaying process plan status](#) on page 251
- [Starting a process plan immediately](#) on page 252
- [Creating and editing process plans](#) on page 252

Overview of process components

Process components and their process tasks form a framework that all process steps can be based on. The tables Jobcomponent, JobTask, and Jobparameter define the complete range of One Identity Manager's own process components and process task with the associated parameters.

Process tasks are used to carry out single basic jobs at system level, for example, adding directories. A process component consists of one or more process tasks and its parameters.

When a process is created, the parameter templates for the process task are copied and entered in the process step. This means that every process step that uses this process task can pass other parameter values. The original is not altered.

NOTE: The information available for the process components and their process tasks is added when the schema is installed and cannot be edited.

To obtain a complete overview of process components and their process tasks and parameters

- In the Designer, select the **Documentation > System configuration reports** category and the **Process components** report.

To display individual process components and their process tasks and parameters

- In the Designer, select the **Process Orchestration > Process components** category.

The following table contains short descriptions of the process components.

NOTE: Additional process components may be available depending on which modules are installed.

Table 87: Short descriptions of process components

Component	Description
AutoUpdateComponent	This process component maps the One Identity Manager Service built-in-tasks.
CommandComponent	This process component runs any command.
DelayComponent	This process component controls the start time of the following process steps.

Component	Description
FileComponent	<p>This process component creates, deletes, copies, and modifies file and directories and also their access permissions.</p> <p>The RSync program is a prerequisite for using the process component on Linux operating systems.</p> <p>Under Windows, some of the process components' process functions required the program XCalcs to edit permissions. You can find this in the your server installation resource kit.</p>
FtpComponent	This process component can transfer file by FTP.
HandleObjectComponent	This process component runs default and custom events for database objects. Each assigned default process is generated as in the front-ends. The component also makes it possible to initiate so called CustomEvents for triggering object related generation of a special process.
LogComponent	This process component is used to log messages, for example, in the result log.
MailComponent	This process component can send emails.
PowerShellComponent	This process is used for calling Windows PowerShell. Version 2.0 of Windows PowerShell must be installed.
PowershellComponentNet4	This process is used for calling a .NET 4 Windows PowerShell. A version of Windows PowerShell later than 2.0 must be installed.
ProjectorComponent	This process component contains tasks for synchronizing and provisioning data with the One Identity Manager database.
ReportComponent	This process component can create reports and export them in various file formats.
ScriptComponent	This process component run the scripts from the assemblies.
SFtpComponent	This process component can transfer files by SFTP.
SQLComponent	This process component runs SQL queries and can be used to determine the number of data records and the existence of data records.
ZipComponent	This process component creates or unpacks ZIP files.

Detailed information about this topic

- [Displaying and editing process task exe types](#) on page 256
- [Changing the maximum number of instances for process tasks and process components](#) on page 257
- [Properties of process components, process tasks, and parameter templates](#) on page 258

Displaying and editing process task exe types

The exe type of a process task defines whether processing takes place within the One Identity Manager Service or in an external process.

To display process task exe types of a process component

- In the Designer, select the **Process Orchestration > Process components > <process component>** category.

On the process component's overview form, you can see all the process tasks with their exe types.

To change the exe type of a process task

1. In the Designer, select the **Process Orchestration > Process components** category.
2. In the Process Component Editor, select the process component.
3. Select a process task and in the edit view on the **Properties** tab, adjust the exe type value.

Permitted values are:

- **Internal:** Runs internally in the One Identity Manager Service.
- **External:** Runs externally as its own process.
- **External32:** Runs externally as its own 32-bit process.

4. Save the changes.

Related topics

- [Overview of process components](#) on page 254
- [Properties of process components, process tasks, and parameter templates](#) on page 258
- [The One Identity Manager Service functionality](#) on page 276
- [Running external processes with the StdioProcessor](#) on page 279

Changing the maximum number of instances for process tasks and process components

The One Identity Manager Service handles parallelization of processes by using the maximum number of instances allowed for process tasks and process components.

To change the maximum number of instances of a process task

1. In the Designer, select the **Process Orchestration > Process components** category.
2. In the Process Component Editor, select the process component.
3. Select a process task and in the edit view on the **Properties** tab, adjust the **Max instances** value to the maximum number instances.

Permitted values are:

- **-1**: All instances of this process task are processed sequentially. Other process task instances of the same process component are not run simultaneously.
- **0**: The maximum number of instances given for the process component is used.
- **1** or greater: The exact number of instances of a process task, which are processed simultaneously.

4. Save the changes.

To change the maximum number of instances of a process component

1. In the Designer, select the **Process Orchestration > Process components** category.
2. In the Process Component Editor, select a process component and in the edit view on the **Properties** tab, adjust the **Max instances** value to the maximum number instances.

Permitted values are:

- **-1**: All instances of this process component are processed sequentially.
It must be ensured that these components are run exclusively on one Job server, which means no other queue can exist to process these components.
- **0**: All instances of this process component can be processed simultaneously.
- **1** or greater: The exact number of instances of a process component, which are processed simultaneously.

NOTE: The value is only used if the maximum number of instances of a process task is set to **0**. Otherwise, the value applies that is set for the process task.

3. Save the changes.

Related topics

- [Overview of process components](#) on page 254
- [Properties of process components, process tasks, and parameter templates](#) on page 258
- [The One Identity Manager Service functionality](#) on page 276
- [Parallel processing of processes by the One Identity Manager Service](#) on page 278

Properties of process components, process tasks, and parameter templates

Table 88: Process component properties

Property	Meaning
Display name	Name of component for displaying.
Component class	Component class.
Assembly name	Name of the component.
Description	Description of component functionality.
Remarks	Additional remarks about the process component.
Max. instances	<p>This value specifies the maximum number of instances in which this process component is allowed to run in a queue in the Job server.</p> <p>Permitted values are:</p> <ul style="list-style-type: none">• -1: All instances of this process component are processed sequentially. It must be ensured that these components are run exclusively on one Job server, which means no other queue can exist to process these components.• 0: All instances of this process component can be processed simultaneously.• 1 or greater: The exact number of instances of a process component, which are processed simultaneously. <p>NOTE: The value is only used if the maximum number of instances of a process task is set to 0. Otherwise, the value applies that is set for the process task.</p>
Configuration	Definition of possible additional options for the component in XML syntax.

Table 89: Process task properties

Property	Meaning
Name	Name of the process task.
Operating system class	Specifies the operating system on which the process task can be run. The Win32 , Linux and ALL values are permitted, where the ALL value specifies that this process task is used on any operating system.
Exe type	Exe type for the process task. Permitted values are: <ul style="list-style-type: none"> • Internal: Runs internally in the One Identity Manager Service. • External: Runs externally as its own process. • External32: Runs externally as its own 32-bit process.
Description	Description of the process task.
Max. instances	This value specifies the maximum number of instances that can be run by One Identity Manager Service in parallel per process task. Permitted values are: <ul style="list-style-type: none"> • -1: All instances of this process task are processed sequentially. Other process task instances of the same process component are not run simultaneously. • 0: The maximum number of instances given for the process component is used. • 1 or greater: The exact number of instances of a process task, which are processed simultaneously.
Last step in the partial process tree	Specifies whether a process task is principally marks the end of a partial process tree.
Component	Process component to which the process task belongs.
Direct database connection required	Specifies whether a process task requires a direct database connection.
Exclusive per object	Specifies whether the process task is run exclusively per object. If this option is enabled in a process task, only one process step with this process task can be run for a specific object. There is no parallel processing.
DBQueue does not wait	Specifies whether or not to wait until the process step has been processed before continuing to process DBQueue Processor tasks. It is only necessary to wait for process steps if a process step could change data that is relevant to the DBQueue Processor tasks.

Table 90: Parameter template properties

Property	Meaning
Name	Name of the parameter.
Value template	Default template for finding values. When a parameter is added to a process step, the value template is taken from the parameter template. Define value templates in VB.Net syntax.
Value template (example)	Example of the value template.
Description	Description of the parameter.
Type	The IN , OUT and INOUT values are permitted.
Optional	Labels the parameter as a mandatory or optional parameter.
Hidden	<p>Specifies whether the parameter is shown in the One Identity Manager Service log file and in the Job Queue Info program. Values for hidden parameters are shown as <HIDDEN>.</p> <p>NOTE: The following users can view the hidden parameters in the Job Queue Info.</p> <ul style="list-style-type: none"> • Administrative users • In the Job Queue Info, users with the Option to see the values of hidden parameters in Job Queue Info program (JobQueue_ShowHiddenParameters) function
Encrypted	Specifies whether the parameter is encrypted when it is passed.
Contains encrypted components	Specifies whether encrypted sequences are contained in this value.
Process task	Process task to which the parameter belongs.

Related topics

- [Changing the maximum number of instances for process tasks and process components](#) on page 257
- [Displaying and editing process task exe types](#) on page 256
- [Parallel processing of processes by the One Identity Manager Service](#) on page 278
- [Running external processes with the StdioProcessor](#) on page 279

Setting up Job servers

The One Identity Manager Service handles defined processes. To run the processes, the One Identity Manager Service has to be installed on the One Identity Manager network server. For more information, see the *One Identity Manager Installation Guide*.

There are several methods for setting up a Job server:

- For the initial schema installation with the Configuration Wizard, you already set up a Job server with the **SQL processing server** and **Update server** server functions. Use the Configuration Wizard to configure the service and install the service remotely on a server.
- To configure further Job servers, use the Server Installer program.
Using the Server Installer, you create the Job server with its machine roles and server functions in the database. Use the Server Installer to configure the service and install the service remotely on a server.
- You can create Job servers in the Designer.
Use the Designer, to create a Job server with the machine roles and server functions, configure the service on the server and install the service remotely.
- If a remote installation is not possible, you can install and configure the service locally on a server.
 - Install the service components on the server using the installation wizard.
 - Configure the service using the Job Service Configuration program.
 - If the **Common | Jobservice | AutoCreateServerFromQueues** configuration parameter is enabled, in response to queries from the One Identity Manager Service for unknown queues, new Job servers are created in the database. Information about machine roles and server functions is transferred to the database.

Setting up a Job server requires the following steps:

- Create an entry for the Job server in the One Identity Manager database.
- Specify the machine roles and server functions for the Job server.
Installation packages to be installed on the Job server are found, depending on the selected machine roles. The server function defines the functionality of a server in One Identity Manager. One Identity Manager processes are handled with respect to the server function.
- Install the One Identity Manager Service.
- Configure the One Identity Manager Service.
- Start the One Identity Manager Service.

Each Job server within the network must have a unique queue identifier. The process steps are requested by the Job queue using exactly this queue name:

- A Job server must be known in the One Identity Manager database for each queue.
- Enter this queue name in the One Identity Manager Service configuration file.

Detailed information about this topic

- [Editing the Job server](#) on page 262
- [Machine roles and server functions](#) on page 265
- [Job server statistics](#) on page 269
- [Connection data for process generation](#) on page 270
- [Installing the One Identity Manager Service on a Job server remotely](#) on page 272
- [Configuring the Job server for connecting to the application server](#) on page 274
- [Customizing the One Identity Manager Service configuration for a Job server](#) on page 284
- [The One Identity Manager Service functionality](#) on page 276
- [Configuring the One Identity Manager Service](#) on page 279

Editing the Job server

To edit a Job server

1. In the Designer, select the **Base Data > Installation > Job server** category.
2. Enter a new Job server using the **Job servers > New** menu item.
- OR -
Select the Job server to be edited in the Job server overview.
3. Edit the Job server's main data.
4. Select the **View > Server functions** menu item and specify the server functionality.
5. Select the **View > Machine roles** menu item and assign roles to the server.
The machine roles expected by a server function, are already assigned.

Detailed information about this topic

- [Job server properties](#) on page 263
- [Machine roles and server functions](#) on page 265
- [Overview of server functions](#) on page 266
- [Overview of machine roles](#) on page 267
- [Job server statistics](#) on page 269
- [Connection data for process generation](#) on page 270

Job server properties

NOTE: More properties may be available depending on which modules are installed.

Table 91: Job server properties

Property	Meaning
Server	Job server name.
Full server name	Full server name in accordance with DNS syntax. Syntax: <Name of servers>.<Fully qualified domain name>
Server is cluster	Specifies whether the server maps a cluster.
Server belongs to cluster	Cluster to which the server belongs. NOTE: The Server is cluster and Server belongs to cluster properties are mutually exclusive.
IP address (IPv6)	Internet protocol version 6 (IPv6) server address.
IP address (IPv4)	Internet protocol version 4 (IPv4) server address.
Coding	Character set coding that is used to write files to the server.
Parent Job server	Name of the parent Job server.
Executing server	Name of the executing server. The name of the server that exists physically and where the processes are handled. This input is evaluated when the One Identity Manager Service is automatically updated. If the server is handling several queues, the process steps are not supplied until all the queues that are being processed on the same server have completed their automatic update.
Queue	Name of the queue to handle the process steps. The process steps are requested by the Job queue using this queue identifier. The queue identifier is entered in the One Identity Manager Service configuration file.
Server operating system	Operating system of the server. This input is required to resolve the path name for replicating software profiles. The values Win32 , Windows , Linux , and Unix are permitted. If no value is specified, Win32 is used.
Service account data	One Identity Manager Service user account information. In order to replicate between non-trusted systems (non-trusted domains, Linux server), the One Identity Manager Service user information has to be declared for the servers in the database. This means that the service

Property	Meaning
	account, the service account domain, and the service account password have to be entered for the server.
One Identity Manager Service installed	<p>Specifies whether a One Identity Manager Service is installed on this server. This option is enabled by the QBM_PJobQueueLoad procedure the moment the queue is called for the first time.</p> <p>The option is not automatically removed. If necessary, you can reset this option manually for servers whose queue is no longer enabled.</p>
Stop One Identity Manager Service	<p>Specifies whether the One Identity Manager Service has stopped. If this option is set for the Job server, the One Identity Manager Service does not process any more tasks.</p> <p>You can make the service start and stop with the appropriate administrative permissions in the Job Queue Info program. For more information, see the <i>One Identity Manager Process Monitoring and Troubleshooting Guide</i>.</p>
No automatic software update	<p>Specifies whether to exclude the server from automatic software updating.</p> <p> NOTE: Servers must be manually updated if this option is set.</p>
Software update running	Specifies whether a software update is currently running.
Port	Port for showing the One Identity Manager Service log file in a browser.
No direct database connection	<p>Specifies whether the Job server has a direct connection to the database. Enable this option if the Job server receives its processes through an application server.</p>
No process assignment	Specifies whether the Job server load balances.
Connection data	<p>If the Job server has no direct connection to the database, enter the connection data for the application service.</p> <p>You can enter the connection data in the Designer, in the Base data > Security settings > Connection data category.</p>
Extended properties	Additional information about Job servers. The UID of the Job server and the details of creation and change (user, date) are displayed. These cannot be edited.
Last fetch time	Last time the process was collected.
Last timeout check	The time of the last check for loaded process steps with a dispatch value that exceeds the one in the Common Jobservice LoadedJob-sTimeout configuration parameter.

Property	Meaning
External port	(For docker containers) Custom port for showing the One Identity Manager Service log file in a browser.
Full server name external	(For docker containers) Custom full server name complying with DNS syntax. Syntax: <Name of servers>.<Fully qualified domain name>
Server function	Server functionality in One Identity Manager. One Identity Manager processes are handled with respect to the server function.
Machine role	Role of the Job server in One Identity Manager. Installation packages to be installed on the Job server are found depending on the selected machine role.

Related topics

- [Overview of server functions](#) on page 266
- [Overview of machine roles](#) on page 267
- [Job server statistics](#) on page 269
- [Connection data for process generation](#) on page 270
- [JobServiceDestination](#) on page 295

Machine roles and server functions

A machine role describes the role a computer or server assumes in a One Identity Manager system. You can give each computer or server several roles. This means, one, or more machine roles can be assigned. You select machine roles when One Identity Manager components are installed.

Machine roles are structured hierarchically. If you select a machine role at installation, all parent machine are also assigned.

Example: Machine role structure

Server

Job server

Active Directory

If you select the **Active Directory** machine role during the installation, the **Job server** and **Server** machine roles are also assigned.

Some machine roles such as **Web** cannot be actively selected during the installation. These machine roles are automatically assigned when different web applications are installed with the Web Installer.

Machine roles for installing the One Identity Manager Service are linked with server functions. The server function defines the functionality of a server in One Identity Manager. One Identity Manager processes are handled with respect to the server function. The server functions available are predefined when a server is installed, based on the selected machine role.

Example: Connection between machine roles and server functions.

The **Active Directory** machine role is connected to the **Active Directory Connector** server function. Therefore, when you set up a One Identity Manager synchronization project after the machine role is installed, the server is available as synchronization server in Active Directory.

The installation packages and files to be installed on the computer or server are specified in a machine role. The information about the machine role, the installation package and the files is saved in the file `InstallState.config` during installation and are thus available for automatic software update.

NOTE: If you use the Software Loader to import new files into the One Identity Manager database, you should assign the files to a machine role. This ensures that the file are distributed by automatic software update. For detailed information about automatic software updates, see the *One Identity Manager Installation Guide*.

Related topics

- [Overview of server functions](#) on page 266
- [Overview of machine roles](#) on page 267
- [Editing the Job server](#) on page 262

Overview of server functions

To display information about server functions

- In the Designer, select the **Base data > Installation > Server functions** category.

The server function defines the functionality of a server in One Identity Manager. One Identity Manager processes are handled with respect to the server function.

NOTE: More server functions may be available depending on which modules are installed.

Table 92: Permitted server functions

Server function	Remark
Update server	<p>This server automatically updates the software on all the other servers. The server requires a direct connection to the database server that One Identity Manager database is installed on. It can run SQL tasks.</p> <p>The server with the One Identity Manager database installed on it is labeled with this functionality during initial installation of the schema.</p>
SQL processing server	<p>It can run SQL tasks. The server requires a direct connection to the database server that One Identity Manager database is installed on.</p> <p>Several SQL processing servers can be set up to spread the load of SQL processes. The system distributes the generated SQL processes throughout all the Job servers with this server function.</p>
CSV script server	This server can process CSV files using the ScriptComponent process component.
One Identity Manager Service installed	Server on which a One Identity Manager Service is installed.
SMTP host	Server from which One Identity Manager Service sends email notifications. Prerequisite for sending mails using One Identity Manager Service is SMTP host configuration.
Default report server	Server on which reports are generated.

Related topics

- [Machine roles and server functions](#) on page 265
- [Editing the Job server](#) on page 262
- [Overview of machine roles](#) on page 267

Overview of machine roles

To display information about machine roles

- In the Designer, select the **Base data > Installation > Machine roles** category.

Installation packages to be installed on the Job server are found depending on the selected machine role.

Table 93: Machine role and installation package options

Machine role	Description of the installation package
Workstation	Contains all basic components for installing tools on an administrative workstation.
Administration	Contains One Identity Manager administration tools required by default users to fulfill their tasks with One Identity Manager. In addition to the tools that ensure basic functionality for working with One Identity Manager, the administration machine role includes the Manager as a main administration tool.
Configuration	Contains all One Identity Manager tools for the default user and additional programs for configuring the system. These include, for example, the Configuration Wizard, Database Compiler, Database Transporter, Crypto Configuration, Designer, Web Designer, and configuration tools for the One Identity Manager Service.
Development & Testing	Contains the One Identity Manager tools for developing and testing custom scripts and forms, for example, the System Debugger.
Monitoring	Contains One Identity Manager programs for monitoring the system status, for example, the Job Queue Info program.
Documentation	Contains One Identity Manager documentation in different languages.
Server	Contains all the basic components for setting up a server.
Job server	Contains the One Identity Manager Service and basic processing components. Additional machine roles contain connectors for synchronizing individual target systems.

NOTE: The **Base data > Installation > Machine roles** category also displays the **API** and **Web** machine roles. These are reserved for internal user and cannot be changed or assigned.

Related topics

- [Machine roles and server functions](#) on page 265
- [Overview of server functions](#) on page 266
- [Editing the Job server](#) on page 262

Job server statistics

This Job server statistical data is evaluated and creates a basis for configuration recommendations for Job server load intervals. The data for the last 100 days is included in the calculation of the configuration recommendations. You should take these configuration suggestions into account when configuring the One Identity Manager Service.

To calculate statistics

- In the Designer, set the **Common | JobQueueStats** configuration parameter. If the configuration parameter is enabled, the One Identity Manager Service statistics are written to the JobQueueStats table.
- In the Designer, set the **Common | JobQueueStats | MaxAge** configuration parameter and enter the retention period for the statistics in days.

For every action in the Job queue, such as inserting, changing, or deleting processes, new statistic entries are created for the Job server. The DBQueue Processor task QBMJobQueueStatsShrink compresses the statistics. The compression takes place for every hour prior to the current hour.

To display Job server statistics

1. In the Designer, select the **Base Data > Installation > Job server** category.
2. Start the Job Server Editor using the **Edit Job server** task.
3. Select the Job server to be edited in the Job server overview.
4. Use the **Select columns** context menu to select the columns with statistics.

These columns are highlighted in the color in the Job server view.

Table 94: Columns for mapping statistics

Column	Name	Meaning
AverageLoad	Average processes/hour	Average number of processes per hour.
MaxLoad	Maximum processes/hour	Maximum number of processes per hour.
LoadDuration	Recommended load interval (secs)	Configuration suggestion for the Process request interval (StartInterval) parameter in the One Identity Manager Service configuration.
StatisticsDuration	Recommended statistic interval (secs)	Configuration suggestion for the Time interval for statistics parameter (StatisticInterval) parameter in the One Identity Manager Service configuration.

Related topics

- [Configuring the One Identity Manager Service](#) on page 279
- [JobServiceDestination](#) on page 295

Connection data for process generation

To generate processes for the Job server, you need the provider, connection parameters and the authentication data. In the default case, this information is determined from the database connection data.

If a Job server has no direct connection to the database, but works with an application server:

- Enter the connection data for the application server.
- Label the Job server with the **No direct database connection** option and assign the connection data to the application server.

TIP: Label one set of connection data for the application server as a **Fallback connection**. This connection data is used if you do not enter any reference to concrete connection data on the Job server.

Determining the connection data during process generation

- The connection data from the database information is used for all Job servers with a direct data connection.
- Connection data for Job servers without a direct database connection is determined as follows:
 1. Connection data that is entered on the Job server.
 2. Connection data that is labeled as a fallback connection.
 3. Connection data that is entered in the database information.

Detailed information about this topic

- [Changing database connection data](#) on page 36
- [Entering connection data for the application server](#) on page 270
- [Entering Job server connection data](#) on page 272

Entering connection data for the application server

Enter the connection data for the application server.

To enter connection data for the application server

1. In the Designer, select the **Base data > Security settings > Connection data** category.
2. Using the **Object > New** menu item, enter new connection data.
3. Enter the following information.

Table 95: Properties of connection data

Property	Description
Display name	Display name of the connection data. Using this display name, you can select the connection data at the Job server entry.
Fallback connection	Label one of the sets of connection data for the application server as a Fallback connection . This connection data is used if you do not enter any reference to concrete connection data on the Job server.
Provider	For connection data for the application server, select Application Server .
Connection parameter	Web address (URL) for the application server. Use the ... button to open the default connection dialog box, from which you can specify other options and test the connection.
Authentication data	Enter the authentication data Syntax: Module=<name>;<property1>=<value1>;<property2>=<value2> Example: Module=DialogUserAccountBased Use the ... button to open a dialog box from which you can select the authentication module directly. The authentication data is transferred when the dialog is closed. For more information about One Identity Manager authentication modules, see the <i>One Identity Manager Authorization and Authentication Guide</i> .

Related topics

- [Connection data for process generation](#) on page 270
- [Entering Job server connection data](#) on page 272

Entering Job server connection data

Label the Job server with the **No direct database connection** option and assign the connection data to the application server.

To declare the connection data on the Job server

1. In the Designer, select the **Base Data > Installation > Job server** category.
2. Select the Job server to be edited in the Job server overview.
3. Edit the following data on the **Properties** tab.
 - Enable the **No direct database connection** option for the Job server.
 - Under **Connection data**, select the connection data for the application server.

Related topics

- [Connection data for process generation](#) on page 270
- [Entering connection data for the application server](#) on page 270

Installing the One Identity Manager Service on a Job server remotely

You have the option to install certain Job servers remotely in the Job Server Editor. The remote installation wizard runs the following steps:

- Installs One Identity Manager Service components.
- Configures the One Identity Manager Service.
- Starts the One Identity Manager Service.

NOTE: To generate processes for the Job server, you need the provider, connection parameters and the authentication data. In the default case, this information is determined from the database connection data. If the Job server runs through an application server, you must configure extra connection data in the Designer. For more information, see [Configuring the Job server for connecting to the application server](#) on page 274.

Prerequisites for remote installation

- The Job server is entered in the database
- There is a user account with sufficient permissions for installing the One Identity Manager Service.
- Remote installation is only supported within a domain or a trusted domain.

NOTE: If you are working with an encrypted One Identity Manager database, see the notes on working with an encrypted database in the *One Identity Manager Installation*

To install the One Identity Manager Service remotely

1. In the Designer, select the **Base Data > Installation > Job server** category.
2. Start the Job Server Editor using the **Edit Job server** task.
3. Select the Job server to be edited in the Job server overview.
4. Select the **Job server > Install service** menu item.

This starts the One Identity Manager Service remote installation wizard.

5. On the start page of the wizard, click **Next**.
6. On the **Configure service** page, enter the One Identity Manager Service configuration settings.

Initial configuration of the service is already predefined for the database connection. To use this template, enter the connection data for process collection. In order to extend the configuration, each configuration section of the One Identity Manager Service is listed in the module list.

- For a direct connection to the database:
 1. Select **Process collection > sqlprovider**.
 2. Click the **Connection parameter** entry, then click the **Edit** button.
 3. Enter the connection data for the One Identity Manager database.
 - For a connection to the application server:
 1. Select **Process collection**, click the **Insert** button and select **AppServerJobProvider**.
 2. Click the **Connection parameter** entry and click the **Edit** button.
 3. Enter the connection data for the application server.
 4. Click the **Authentication data** entry and click the **Edit** button.
 5. Select the authentication module. Depending on the authentication module, other data may be required, for example, user, and password. For more information about One Identity Manager authentication modules, see the *One Identity Manager Authorization and Authentication Guide*.
7. On the **Installation source and destination** page, enter the following information.
 - a. General information:
 - **Installation directory:** Select the directory containing the installation files.
 - **Private key:** If the database is encrypted, select the file with the private key.
 - b. Click **Next**.
 - c. Enter the service's installation data.

- **Computer:** Enter the name or IP address of the server that the service is installed and started on.
- **Service account:** Enter the details of the user account that the One Identity Manager Service is running under. Enter the user account, the user account's password and password confirmation.

The service is installed using the user account with which you are logged in to the administrative workstation. If you want to use another user account for installing the service, you can enter it in the advanced options. You can also change the One Identity Manager Service details, such as the installation directory, name, display name, and the One Identity Manager Service description, using the advanced options.

8. Click **Next** to start installing the service.

Installation of the service occurs automatically and may take some time.

9. Click **Close** to end the workflow wizard.

NOTE: In a default installation, the service is entered in the server's service management with the name **One Identity Manager Service**.

TIP: Use the **Job server > Start HTTP request** menu item to address the HTTP server of the One Identity Manager Service for a Job server and display the different services of the One Identity Manager Service.

Related topics

- [Setting up Job servers](#) on page 261
- [The One Identity Manager Service functionality](#) on page 276
- [Configuring the One Identity Manager Service](#) on page 279

Configuring the Job server for connecting to the application server

To configure the Job server for connecting to the application server

- Declare the Job server in the One Identity Manager database.
- Install the One Identity Manager Service and configure the **AppServerJobProvider** for process collection.
- To generate processes through an application server, enter the connection data in the Job server.
 - Enter the connection data for the application server.
 - Label the Job server with the **No direct database connection** option and assign the connection data to the application server.

Related topics

- [Editing the Job server on page 262](#)
- [Installing the One Identity Manager Service on a Job server remotely on page 272](#)
- [AppServerJobProvider on page 294](#)
- [Connection data for process generation on page 270](#)

The One Identity Manager Service functionality

The One Identity Manager Service enables the distribution throughout the network of information that is administrated in the One Identity Manager database. The One Identity Manager Service performs data synchronization between the database and any connected target systems and runs actions at the database and file level.

Process steps are run by process components. The One Identity Manager Service also creates an instance of the required process component and transfers the process step parameters. Decision logic monitors the performance of the process steps and determines how processing should continue depending on the results of the run process components.

For more information about installing and updating the One Identity Manager Service, see the *One Identity Manager Installation Guide*.

For more information about logging and monitoring process handling and support for troubleshooting, see the *One Identity Manager Process Monitoring and Troubleshooting Guide*.

Related topics

- [Handling processes with the One Identity Manager Service](#) on page 277
- [Parallel processing of processes by the One Identity Manager Service](#) on page 278
- [Running external processes with the StdioProcessor](#) on page 279
- [Configuring the One Identity Manager Service](#) on page 279
- [Setting up Job servers](#) on page 261
- [Process orchestration in One Identity Manager](#) on page 220

Handling processes with the One Identity Manager Service

The processing tasks for the One Identity Manager Service are saved in the Job queue (JobQueue table) for a defined queue. A One Identity Manager Service can process several queues. The queues that a One Identity Manager Service can process are declared in the One Identity Manager Service's configuration. A Job server must be known in the One Identity Manager database for each queue. The One Identity Manager Service queries the Job queue to see which processes are waiting for its own queue. A queue is initialized when the One Identity Manager Service starts.

The process requests and processing results are cached internally in a request queue (RequestQueue) and a result queue (ResultQueue). The request queue is processed in parallel to the result queue. The following requests use the two internal queues:

- Request queue
 - Request for pending process steps
 - Request for statistics data
- Results queue
 - Process results request
 - Processing status request
 - Request for events
 - Initial request of the pending process step when the One Identity Manager Service starts up. This ensures that any missing processing results in the database have arrived before enabling process steps that still have **Loaded** status.

When the One Identity Manager Service is downloaded, any requests that may still be in the results queue are serialized in a file that is processed the next time the One Identity Manager Service starts. This should ensure that no processing results go missing. The backup files are kept in the local program data directory (%APPDATA%\One Identity\One Identity Manager\JobService).

Related topics

- [Configuring the One Identity Manager Service](#) on page 279
- [Setting up Job servers](#) on page 261

Parallel processing of processes by the One Identity Manager Service

The One Identity Manager Service enables parallel processing of process steps because it can create several instances of process components. You specify for each process component and its process tasks on an individual basis whether parallel processing of process steps is possible.

Parallelization is affected by the following properties of the process components and process tasks.

- Maximum number of valid instances of one process task (`JobTask.MaxInstance`)

Permitted values are:

- **-1**: All instances of this process task are processed sequentially. Other process task instances of the same process component are not run simultaneously.
- **0**: The maximum number of instances given for the process component is used.
- **1** or greater: The exact number of instances of a process task, which are processed simultaneously.

- Maximum number of valid instances of one process component (`JobComponent.MaxInstance`)

The value is only used if the maximum number of instances of a process task is set to **0**. Otherwise, the value applies that is set for the process task.

Permitted values are:

- **-1**: All instances of this process component are processed sequentially.
It must be ensured that these components are run exclusively on one Job server, which means no other queue can exist to process these components.
- **0**: All instances of this process component can be processed simultaneously.
- **1** or greater: The exact number of instances of a process component, which are processed simultaneously.

- Specifies whether it is necessary to run one process task per object (`JobTask.IsExclusivePerObject`)

If this option is enabled in a process task, only one process step with this process task can be run for a specific object. There is no parallel processing.

Related topics

- [Overview of process components](#) on page 254
- [Properties of process components, process tasks, and parameter templates](#) on page 258
- [Changing the maximum number of instances for process tasks and process components](#) on page 257

Running external processes with the StdioProcessor

The exe type of a process task defines whether processing takes place within the One Identity Manager Service or in an external process. If a process task is going to run in an external process, the StdioProcessor (StdioProcessor) is started on an external slot. After the external process has run, it remains available for further runs until one of the following conditions occurs:

- No further process step has been started for the past 30 seconds.
- The maximum reuse count has been reached according to the One Identity Manager Service configuration.

The One Identity Manager Service configuration uses the value in the **Max. external processor reuse count** (MaxExternalSlotReuse) as the maximum reuse count. The default value is **100**. For more information, see [JobServiceDestination](#) on page 295.

- The process step has the return code ErrorAndTerminate.

Related topics

- [Overview of process components](#) on page 254
- [Properties of process components, process tasks, and parameter templates](#) on page 258
- [Displaying and editing process task exe types](#) on page 256
- [Configuring the One Identity Manager Service](#) on page 279

Configuring the One Identity Manager Service

A Job provider function makes a Job destination process step available within the One Identity Manager Service. The Job destination function handles the process steps and returns a result to the Job provider. The Job provider evaluates the result.

The combination of a Job provider on one server and a Job destination on another server is called a "Job gate". The Job provider and Job destination are configured within the Jobgate such that they can communicate with each other.

Figure 27: Example of the One Identity Manager Service configuration

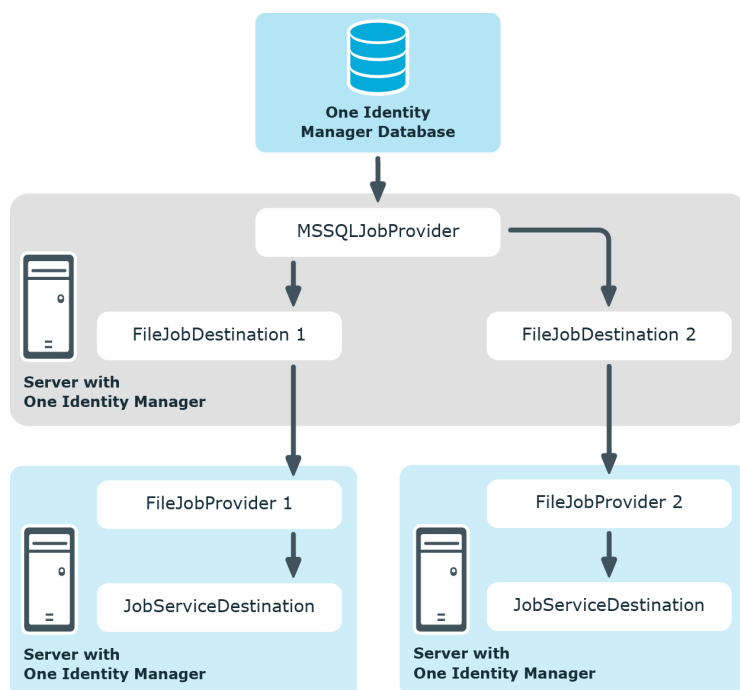


Table 96: One Identity Manager Service provider

Provider	Description
MSSQLJobProvider	The MSSQLJobProvider retrieves the process steps from the One Identity Manager database under SQL Server and sends them to a Job destination.
FileJobProvider	In the FileJobProvider , process requests and results are read from and written to files. These files can be processed by the FileJobGate (FileJobDestination or FTPJobDestination) . The data is transferred using these files.
FTPJobProvider	The FTPJobProvider is based on the function of the FileJobProvider . In the FTPJobProvider , process requests and results are read from and written to files. After the files have been created in the local directory, the FTPJobProvider connects to the FTP server and transfers the files to the server. A connection is also made to the FTP Server when it gets a signal and the data is collected.
HTTPJobProvider	The HTTPJobProvider receives process steps from a parent Job server. The data transfer is carried out by HTTP.
AppServerJobProvider	The AppServerJobProvider retrieves the process steps from the application server and sends them to a Job destination.

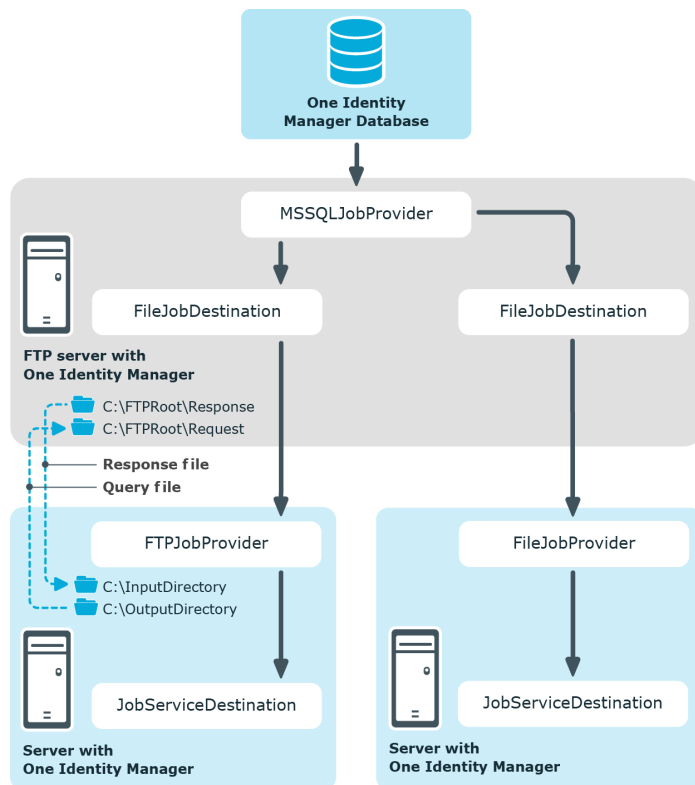
Table 97: One Identity Manager Service Job destinations

JobDestination	Description
JobServiceDestination	The JobServiceDestination is the One Identity Manager Service component that performs the actual handling of process steps. It requests the process steps from the Job provider, processes them with the process component and returns the result.
FileJobDestination	The FileJobDestination handles the process steps provided by the FileJobGate (FileJobProvider or FTPJobProvider) and returns the results to the Job provider.
FTPJobDestination	The FTPJobDestination handles the process steps provided by the FileJobGate (FileJobProvider or FTPJobProvider) and returns the results to the Job provider.
HTTPJobDestination	The HTTPJobDestination sends process steps to a child Job server. The data transfer is carried out by HTTP.

Table 98: One Identity Manager Service Jobgates

Jobgate	Description
HTTPJobGate	Consisting of HTTPJobProvider and HTTPJobDestination .
FileJobGate	Consisting of FileJobProvider , FileJobDestination , FTPJobProvider and FTPJobDestination . JobProvider and JobDestinations can be combined with each other.

Figure 28: Example FileJobGate configuration



Detailed information about this topic

- [One Identity Manager Service configuration files](#) on page 283
- [Customizing the One Identity Manager Service configuration for a Job server](#) on page 284
- [Process collection module](#) on page 288
- [Job destination module](#) on page 294
- [Configuration module](#) on page 302
- [Logwriter module](#) on page 305
- [Dispatcher module](#) on page 308
- [Connection module](#) on page 309
- [HTTP authentication module](#) on page 310
- [Module plugins](#) on page 310
- [File module with private key](#) on page 314

One Identity Manager Service configuration files

You configure One Identity Manager Service and its plugins using a configuration file. The file has to reside in the same directory as the file `viNetworkService`. The configuration file is necessary both for One Identity Manager Service on a windows based operating system and for the Linux daemon.

Two configuration file formats are supported:

- `Jobservice.cfg`

`Jobservice.cfg` is an XML configuration file with its own format. The advantage of this file is that run-time loading is supported.

- `viNetworkService.exe.config`

The `viNetworkService.exe.config` file is the default configuration file for .NET exes and has the specified format.

The system initially searches for the parameter in the configuration file `Jobservice.cfg` in order to determine the setups. If the parameter is not found, the file `viNetworkService.exe` is automatically used. Thus the One Identity Manager Service can only work with the configuration file `viNetworkService.exe.config`.

In the Designer, configure the One Identity Manager Service in the **Base data > Installation > Job server** category or by using the Job Service Configuration program.

There is one unique section in the file for each of the different modules in One Identity Manager Service.

Table 99: One Identity Manager Service modules

Module	Description
Process collection	Specify the Job provider in this module.
JobDestination	In this module, you specify the job destination.
Configuration	Standard configuration settings for One Identity Manager Service are in this module.
LogWriter	This module writes One Identity Manager Service messages to a log file.
Request dispatcher	Use this module to configure the One Identity Manager Service as a dispatcher. The process requests from the child Job server are buffered, processed, and forwarded.
Connection	With this module you can set special configuration settings for the behavior of the One Identity Manager Service.
HTTP authen-	Use this module to specify how authentication works on an HTTP server

Module	Description
tication module	so that extended services can be accessed, for example, displaying the log file or the status display.
Plugins	Specify which plugins should be installed in this module.
File with the private key.	In this module, you provide the data for files with a private key. Use this module if you are working with more than one private key.

Detailed information about this topic

- [Configuring the One Identity Manager Service](#) on page 279
- [Customizing the One Identity Manager Service configuration for a Job server](#) on page 284
- [Process collection module](#) on page 288
- [Job destination module](#) on page 294
- [Configuration module](#) on page 302
- [Logwriter module](#) on page 305
- [Dispatcher module](#) on page 308
- [Connection module](#) on page 309
- [HTTP authentication module](#) on page 310
- [Module plugins](#) on page 310
- [File module with private key](#) on page 314
- [One Identity Manager Service configuration files](#) on page 513





Customizing the One Identity Manager Service configuration for a Job server

This configuration is already created when the One Identity Manager Service is installed. Use the Job Server Editor to modify each configuration setting. You can also customize all configuration settings in the Job Service Configuration program.

NOTE: Before changing the One Identity Manager Service configuration, make sure that the configuration is imported into the database. In the Designer, configure, and enable the **Get configuration file from the Job server and write in the Job server configuration** schedule.

To modify the One Identity Manager Service configuration on a Job server

1. In the Designer, select the **Base Data > Installation > Job server** category.
2. Start the Job Server Editor using the **Edit Job server** task.

3. Enable the **Configure One Identity Manager Service** view.
4. Select the Job server to be edited in the Job server overview.
5. Edit the configuration settings.
| **TIP:** Use the  and  buttons to change the configuration data.
6. Save the configuration using .
7. Use the  button to test the configuration.
8. Deploy the modified configuration to the Job server using **Job server > Deploy Job server configuration** from menu.

This generates a process, which updates the configuration file on the Job server.

| **TIP:** Use the **Job server > Start HTTP request** menu item to address the HTTP server of the One Identity Manager Service for a Job server and display the different services of the One Identity Manager Service.

Related topics

- [Configuring the One Identity Manager Service](#) on page 279
- [Template for the configuration file](#) on page 285
- [Selecting module types and editing parameters](#) on page 286
- [Validating the configuration file](#) on page 287

Template for the configuration file

| **NOTE:** The template is only available in the Job Service Configuration program.

The **SQL server direct** template is supplied for configuring the One Identity Manager Service with a direct database connection.

This template already contains the most important modules with settings for a simple One Identity Manager Service configuration with a direct connection to a SQL Server. You can load the template using the menu item **Templates > SQL server direct**. After loading, the configuration the template needs to be modified as required.

Related topics

- [Customizing the One Identity Manager Service configuration for a Job server](#) on page 284
- [Selecting module types and editing parameters](#) on page 286
- [Validating the configuration file](#) on page 287

Selecting module types and editing parameters

The module list of the One Identity Manager Service configuration gives an overview of the individual configuration sections. A selection of module types is available for certain modules.

To select a module type

1. Click on the module in the module list.
2. Use **Insert** to open the module type menu.
3. Select the module type you want from the list and insert it with the **OK** button.

To change the name of a module type

1. Click on the module in the module list.
2. Select the module type and click **Rename**.
3. Change the name of the module type.
4. Press return.

To delete a module type

1. Click on the module in the module list.
2. Select the module type and click **Delete**.

To edit a parameter value





1. Select the parameter in the **Properties** column.
2. Click **Edit**.



When a item is selected in the module list, all possible parameters and their values are displayed. You can change some values by clicking in input field or on the option button in the **Value** column.

NOTE: The parameter description in each module contains the parameter name, in brackets, which is used in the configuration file.

The following icons are used:

Table 100: Meaning of the icons for the module parameters

Icon	Meaning
	The value is passed as a string.
	Compulsory input. The parameter must be altered as required. The value is passed as a string.
	The value is passed as an integer.
	The parameter can be enabled or disabled.

Icon	Meaning
	This parameter is added during run-time. The One Identity Manager Service does not need to be restarted.
	The parameter takes effect after the One Identity Manager Service is restarted.

Related topics

- [Customizing the One Identity Manager Service configuration for a Job server on page 284](#)
- [Template for the configuration file on page 285](#)
- [Validating the configuration file on page 287](#)

Validating the configuration file

The verification test ensures that the minimum requirements for a configuration file are met.

To start the validity check in the One Identity Manager Service configuration


- Use the  button to test the configuration.
Errors and warnings are sent to a message window.

Table 101: Verification test error output

Errors	Output
No Job provider found.	Errors
No Logwriter found.	Errors
No input in compulsory field.	Errors
No Job destination found.	Warning
No plugins found.	Warning

Related topics

- [Customizing the One Identity Manager Service configuration for a Job server on page 284](#)
- [Template for the configuration file on page 285](#)
- [Selecting module types and editing parameters on page 286](#)

Process collection module

In this module you define the Job providers. The Job provider supplies a Job destination process step and evaluates the result. The following module types may be selected:

- [MSSQLJobProvider](#)
- [FileJobProvider](#)
- [FTPJobProvider](#)
- [HTTPJobProvider](#)
- [AppServerJobProvider](#)

You can configure any number of Job providers in one configuration file. The associated configuration sections are determined by name. Therefore, you should rename Job providers that are added.

MSSQLJobProvider

The **MSSQLJobProvider** handles One Identity Manager database process requests on a SQL Server.

Table 102: MSSQLJobProvider parameters

Parameters	Description
Connection parameter (ConnectionString)	<p>Access data for the database server and the database to be used At least one user with the configuration user access level is required for process collection. Some functions may require an administrative user.</p> <ul style="list-style-type: none">• Server: Database server.• (Optional) Windows Authentication: Specifies whether the integrated Windows authentication is used. This type of authentication is not recommended. If you decide to use it anyway, ensure that your environment supports Windows authentication.• User: The user's SQL Server login name.• Password: Password for the user's SQL Server login.• Database: Select the database.
Max. number of pending requests (RequestQueueLimit)	<p>The process requests are internally cached. This parameter defines the maximum number of cache entries. The default value is 1000.</p>
Max. number of pending results	<p>The process results are internally cached. This parameter defines the maximum number of cache entries. The default value</p>

Parameters	Description
(RequestQueueLimit)	is 10000 .
Results flush timeout on shutdown in seconds (FlushTimeoutSeconds)	The service can continue to write results to the database until this timeout is reached. Input in seconds. The default value is 10 seconds.

FileJobProvider

Data transfer takes place in the **FileJobProvider** by means of files. Process requests and results are written to file or read from file. These files can be processed by the **FileJobDestination**.

Table 103: FileJobProvider parameters

Parameters	Description
Backup of transferred files (BackupFiles)	If this option is enabled, all files (with or without errors) are moved to a Backup subdirectory. In the default case (not set) only files with errors are saved.
Check file index (CheckInputIndex)	If this option is set, the file name index is checked to see if has increased in size. Files with the same or a lower index are not processed. This option is not set by default.
Max. number of process trees in a transfer file (MaxListCount)	Maximum number of process steps that can be grouped together as a file. This allows limiting of the file size.
Use encryption (UseEncryption)	Specifies whether the data is to be written to the files in encrypted form. NOTE: The setting for encryption must be identically configured in the Job provider and the related Job destination.
Notification methods (EventTypes)	<p>The Job provider supports three different methods for providing notification about new data.</p> <ul style="list-style-type: none"> • Timer: Newly stored data is queried at defined intervals. • HTTP: The provider queries the parent Job server through HTTP and processes the stored data once the server replies. • FSEvent: Newly stored data is queried after a file system event. <p>The notification methods can be combined when separated by commas.</p> <p>Example:</p>

Parameters	Description
	TIMER, FSEVENT
Remote host for HTTP notification (HostName)	If using the HTTP notification method, enter the name of the remote host here to which the queries are transferred.
HTTP notification port (Port)	If using the HTTP notification method, enter the port for transfer here.
File lookup timer interval (ms) (TimerInterval)	If using the TIMER notification method, enter the interval in milliseconds here.
Input directory (InputDirectory)	<p>The module reads and processes the process files (*.fjg) in this directory.</p> <p>NOTE: Ensure that the Job provider and related Job destination use the same directory. Input directory and output directory are then reversed accordingly.</p>
Output directory (OutputDirectory)	Directory to which the processed files are written.
Subdirectories (SubDirectories)	<p>You can enter a list of directory names separated by a pipe character () here. All the directories are then monitored and processed correspondingly. The following directory structure is expected:</p> <pre>SubDirectories = "ServerA ServerB" ... Request ServerA ServerB Response ServerA ServerB</pre> <p>where Request and Response are the directories specified in the Input directory (InputDirectory) and Output directory (OutputDirectory) parameters.</p> <p>NOTE: You can only use the Timer notification method. The HTTP and FSEvent notification methods are not available.</p>
Automatic identification of subdirectories (AutoSubDirectories)	If this option is enabled, the module automatically processes all the files in the subdirectories. Processing is not recursive.

Related topics

- [FTPJobProvider](#) on page 291
- [FileJobDestination](#) on page 297
- [FTPJobDestination](#) on page 299

FTPJobProvider

After the files have been created in the local directory, the **FTPJobProvider** connects to the FTP server and transfers the files to the server. After a signal, a connection is set up to the FTP Server and the data is transferred. The directories Request and Response are expected to be found on the FTP Server. The names of these directories are fixed and cannot be changed. The software components (Job provider/Job destination) deposit or collect the files from here. The FTP user requires the necessary permissions to create, rename, and delete files.

Table 104: FTPJobProvider parameters

Parameters	Description
Backup of transferred files (BackupFiles)	If this option is enabled, all files (with or without errors) are moved to a Backup subdirectory. In the default case (not set) only files with errors are saved.
Check file index (CheckInputIndex)	If this option is set, the file name index is checked to see if has increased in size. Files with the same or a lower index are not processed. This option is not set by default.
Max. number of process trees in a transfer file (MaxListCount)	Maximum number of process steps that can be grouped together as a file. This allows limiting of the file size.
Use encryption (UseEncryption)	Specifies whether the data is to be written to the files in encrypted form. NOTE: The setting for encryption must be identically configured in the Job provider and the related Job destination.
Notification methods (EventTypes)	The Job provider supports three different methods for providing notification about new data. <ul style="list-style-type: none">• Timer: Newly stored data is queried at defined intervals.• HTTP: The provider queries the parent Job server through HTTP and processes the stored data once the server replies.• FSEvent: Newly stored data is queried after a file system event.

The notification methods can be combined when separated by

Parameters	Description
	<p>commas.</p> <p>Example:</p> <p>TIMER, FSEVENT</p>
Remote host for HTTP notification (HostName)	If using the HTTP notification method, enter the name of the remote host here to which the queries are transferred.
HTTP notification port (Port)	If using the HTTP notification method, enter the port for transfer here.
File lookup timer interval (ms) (TimerInterval)	If using the TIMER notification method, enter the interval in milliseconds here.
Input directory (InputDirectory)	<p>The module reads and processes the process files (*.fjg) in this directory.</p> <p>NOTE: Ensure that the Job provider and related Job destination use the same directory. Input directory and output directory are then reversed accordingly.</p>
Output directory (OutputDirectory)	Directory to which the processed files are written.
Subdirectories (SubDirectories)	<p>You can enter a list of directory names separated by a pipe character () here. All the directories are then monitored and processed correspondingly. The following directory structure is expected:</p> <pre>SubDirectories = "ServerA ServerB" ... Request ServerA ServerB Response ServerA ServerB</pre> <p>where Request and Response are the directories specified in the Input directory (InputDirectory) and Output directory (OutputDirectory) parameters.</p> <p>NOTE: You can only use the Timer notification method. The HTTP and FSEvent notification methods are not available.</p>
Automatic identi-	If this option is enabled, the module automatically processes all

Parameters	Description
ification of subdirectories (AutoSubDirectories)	the files in the subdirectories. Processing is not recursive.
FTP Server (FTPServer)	Name or IP address of the FTP server.
FTP port (FTPPort)	Port for FTP transfer The default port is port 21.
FTP user account (FTPUser)	User account for FTP login.
FTP password (FTPPassword)	Password for the user account for FTP login.

Related topics

- [FileJobProvider](#) on page 289
- [FileJobDestination](#) on page 297
- [FTPJobDestination](#) on page 299

HTTPJobProvider

The **HTTPJobProvider** receives process steps from a parent Job server. The data transfer is carried out by HTTP.

Table 105: HTTPJobProvider parameters

Parameters	Description
Receiver port (ParentPort)	HTTP port of the parent Job server.
Receiver server (ParentServer)	DNS name or IP address of the parent Job server.
Retries	Number of retries performed by the module if the data transfer fails.
RetryDelay	This defines how long the module will wait after a failed process step transfer before retrying. Timeout format: day.hour:minutes:seconds
Remote domain (RemoteDomain)	User account domain on the remote HTTP server.

Parameters	Description
Remote user account (RemoteUser)	User account for logging onto the HTTP server.
RemotePassword	Password for the user account for logging onto the HTTP server.

Related topics

- [HTTPJobDestination](#) on page 301

AppServerJobProvider

The **AppServerJobProvider** retrieves the process steps from the application server and sends them to a job destination.

Table 106: AppServerJobProvider parameters

Parameters	Description
Authentication data (AuthenticationString)	Select the authentication module. Depending on the authentication module, other data may be required, for example, user, and password. For detailed information about One Identity Manager authentication modules, see the <i>One Identity Manager Authorization and Authentication Guide</i> .
Max. number of pending requests (RequestQueueLimit)	The process requests are internally cached. This parameter defines the maximum number of cache entries. The default value is 1000 .
Max. number of pending results (ResultQueueLimit)	The process results are internally cached. This parameter defines the maximum number of cache entries. The default value is 10000 .
Connection parameter (ConnectionString)	Web address (URL) of the application server.

Job destination module

In this module, you specify the job destination. This handles the process steps and returns an result to the Job provider. The following module types may be selected:

- [JobServiceDestination](#)
- [FileJobDestination](#)
- [FTPJobDestination](#)
- [HTTPJobDestination](#)

NOTE: You can configure any number of job destinations in one configuration file. The associated configuration sections are determined by name. Therefore the Job destinations that are added can be renamed.

JobServiceDestination

The **JobServiceDestination** module of the One Identity Manager Service performs the actual handling of process steps. A **JobServiceDestination** requests the process steps from the job provider, processes them using process components and returns the result.

Table 107: JobServiceDestination parameters

Parameters	Description
Number of external slots (ExternalSlots)	Maximum number of external processes (StdioProcessor.exe) opened by the One Identity Manager Service for handling process components.
Environment variables for external slots (ExternalSlotEnvironment)	List of environment variables to set for external slot processes. Enter the variables in a pipe () delimited list. Syntax: Variable1=value1 Variable2=value2...
Number of external 32-bit slots (ExternalSlots32)	Maximum number of external processes in the 32-bit memory (StdioProcessor32.exe) opened by the One Identity Manager Service for handling process components.
Environment variables for external 32-bit slots (ExternalSlotEnvironment32)	List of environment variables to set for external 32-bit slot processes. Enter the variables in a pipe () delimited list. Syntax: Variable1=value1 Variable2=value2...
Number of internal slots (InternalSlots)	Number of internal process provided by the One Identity Manager Service for the internal handling of process components.
File with private key (PrivateKey)	File with encryption information. The default file is private.key . The encryption file has to be in the installation directory of all servers with One Identity Manager Service. If the One Identity Manager Service finds a private key on start up, it places it in the user-specific key container and deletes the file from the hard drive. To create a key file and encrypt database information, use the Crypto Configuration program. NOTE: If you are working with an encrypted One

Parameters	Description
	Identity Manager database, see the notes on working with an encrypted database in the <i>One Identity Manager Installation Guide</i> .
Encryption method (EncryptionScheme)	<p>Encryption method used</p> <p>Permitted values are:</p> <ul style="list-style-type: none"> • RSA: RSA encryption with AES for large data (default). • FIPSCompliantRSA: FIPS certified RSA with AES for large data. This method is used if encryption must match the FIPS 104-2 standard. The local security policy Use FIPS compliant algorithms for encryption, hashing, and signing must be enabled.
ProviderID	if more than one Job provider is being processed by the One Identity Manager Service, enter the name of the Job provider to be used. If this is empty the first Job provider is used.
Private key identifier (PrivateKeyId)	<p>Identifier of the private key. If no ID is specified, a search is performed for the private.key file.</p> <p>Use this parameter if you work with several private keys, for example, if One Identity Manager Service data must be exchanged between two encrypted One Identity Manager databases. Enter the private keys in the File with private key module. If One Identity Manager only uses an encrypted database, you can alternatively enter the key file in the File with private key parameter (PrivateKey).</p>
Queue	Queue identifier The process steps are requested by the Job queue using this queue identifier. A Job server must be known in the One Identity Manager database for each queue.
RequestTimeout	<p>Specifies when a process request has failed and is resent.</p> <p>Timeout format:</p> <p>day.hour:minutes:seconds</p>
Process request interval (StartInterval)	Interval in seconds after which the One Identity Manager Service requests new process steps The default value is 90 seconds. Suggestions for configuring the time interval are calculated from Job server statistical data.
Interval for calculating statist-	Interval in seconds in which the One Identity Manager

Parameters	Description
ics (StatisticInterval)	Service delivers statistic information on processing speed to the database. The default value is set to 4 times the process request interval. Suggestions for configuring the time interval are calculated from Job server statistical data.
Max. external processor reuse count (MaxExternalSlotReuse)	Specifies how often an external processor can be reused before the process is unloaded and restarted. The value 0 indicates that the process is only unloaded when no longer in use. The default value is 100 .

Related topics

- [Setting up Job servers](#) on page 261
- [Job server properties](#) on page 263
- [Job server statistics](#) on page 269
- [File module with private key](#) on page 314
- [Running external processes with the StdioProcessor](#) on page 279

FileJobDestination

The **FileJobDestination** handles the process steps provided by the **FileJobGate** (**FileJobProvider** or **FTPJobProvider**) and returns the results to the job provider.

Table 108: FileJobDestination parameters

Parameters	Description
Backup of transferred files (BackupFiles)	If this option is enabled, all files (with or without errors) are moved to a Backup subdirectory. In the default case (not set) only files with errors are saved.
Check file index (CheckInputIndex)	If this option is set, the file name index is checked to see if has increased in size. Files with the same or a lower index are not processed. This option is not set by default.
Max. number of process trees in a transfer file (MaxListCount)	Maximum number of process steps that can be grouped together as a file. This allows limiting of the file size.
Use encryption (UseEncryption)	Specifies whether the data is to be written to the files in encrypted form. NOTE: The setting for encryption must be identically configured in the job provider and the related job destination.

Parameters	Description
Notification methods (EventTypes)	<p>The job provider supports three different methods for providing notification about new data.</p> <ul style="list-style-type: none"> • Timer: Newly stored data is queried at defined intervals. • HTTP: The provider queries the parent Job server through HTTP and processes the stored data once the server replies. • FSEvent: Newly stored data is queried after a file system event. <p>The notification methods can be combined when separated by commas.</p> <p>Example:</p> <p>TIMER,FSEVENT</p>
Remote host for HTTP notification (HostName)	If using the HTTP notification method, enter the name of the remote host here to which the queries are transferred.
HTTP notification port (Port)	If using the HTTP notification method, enter the port for transfer here.
File lookup timer interval (ms) (TimerInterval)	If using the TIMER notification method, enter the interval in milliseconds here.
Input directory (InputDirectory)	<p>The module reads and processes the process files (*.fjg) in this directory.</p> <p>NOTE: Ensure that the job provider and related job destination use the same directory. Input directory and output directory are then reversed accordingly.</p>
Output directory (OutputDirectory)	Directory to which the processed files are written.
Subdirectories (SubDirectories)	<p>You can enter a list of directory names separated by a pipe character () here. All the directories are then monitored and processed correspondingly. The following directory structure is expected:</p> <pre>SubDirectories = "ServerA ServerB" ... Request ServerA ServerB Response</pre>

Parameters	Description
	ServerA ServerB where Request and Response are the directories specified in the Input directory (InputDirectory) and Output directory (OutputDirectory) parameters. NOTE: You can only use the Timer notification method. The HTTP and FSEvent notification methods are not available.
Automatic identification of subdirectories (AutoSubDirectories)	If this option is enabled, the module automatically processes all the files in the subdirectories. Processing is not recursive.
ProviderID	if more than one job provider is being processed by the One Identity Manager Service, enter the name of the job provider to be used. If this is empty the first Job provider is used.

Related topics

- [FileJobProvider](#) on page 289
- [FTPJobProvider](#) on page 291
- [FTPJobDestination](#) on page 299

FTPJobDestination

The **FTPJobDestination** handles the process steps provided in the **FileJobGate** (**FileJobProvider** or **FTPJobProvider**) and returns the results to the Job provider.

Table 109: FTPJobDestination parameters

Parameters	Description
Backup of transferred files (BackupFiles)	If this option is enabled, all files (with or without errors) are moved to a Backup subdirectory. In the default case (not set) only files with errors are saved.
Check file index (CheckInputIndex)	If this option is set, the file name index is checked to see if has increased in size. Files with the same or a lower index are not processed. This option is not set by default.
Max. number of process trees in a transfer file (MaxListCount)	Maximum number of process steps that can be grouped together as a file. This allows limiting of the file size.

Parameters	Description
Use encryption (UseEncryption)	Specifies whether the data is to be written to the files in encrypted form. NOTE: The setting for encryption must be identically configured in the Job provider and the related Job destination.
Notification methods (EventTypes)	<p>The Job provider supports three different methods for providing notification about new data.</p> <ul style="list-style-type: none"> • Timer: Newly stored data is queried at defined intervals. • HTTP: The provider queries the parent Job server through HTTP and processes the stored data once the server replies. • FSEvent: Newly stored data is queried after a file system event. <p>The notification methods can be combined when separated by commas.</p> <p>Example:</p> <p>TIMER,FSEVENT</p>
Remote host for HTTP notification (HostName)	If using the HTTP notification method, enter the name of the remote host here to which the queries are transferred.
HTTP notification port (Port)	If using the HTTP notification method, enter the port for transfer here.
Monitoring interval for input directory (TimerInterval)	If using the TIMER notification method, enter the interval in milliseconds here.
Input directory (InputDirectory)	<p>The module reads and processes the process files (*.fjg) in this directory.</p> <p>NOTE: Ensure that the Job provider and related Job destination use the same directory. Input directory and output directory are then reversed accordingly.</p>
Output directory (OutputDirectory)	Directory to which the processed files are written.
Subdirectories (SubDirectories)	<p>You can enter a list of directory names separated by a pipe character () here. All the directories are then monitored and processed correspondingly. The following directory structure is expected:</p> <pre>SubDirectories = "ServerA ServerB" ... Request</pre>

Parameters	Description
	ServerA
	ServerB
	Response
	ServerA
	ServerB
	where Request and Response are the directories specified in the Input directory (InputDirectory) and Output directory (OutputDirectory) parameters.
	NOTE: You can only use the Timer notification method. The HTTP and FSEvent notification methods are not available.
Automatic identification of subdirectories (AutoSubDirectories)	If this option is enabled, the module automatically processes all the files in the subdirectories. Processing is not recursive.
ProviderID	if more than one Job provider is being processed by the One Identity Manager Service, enter the name of the Job provider to be used. If this is empty the first Job provider is used.
FTP Server (FTPServer)	Name or IP address of the FTP server.
FTP port (FTPPort)	Port for FTP transfer The default port is port 21.
FTP user account (FTPUser)	User account for FTP login.
FTP password (FTPPassword)	Password for the user account for FTP login.

Related topics

- [FileJobProvider](#) on page 289
- [FTPJobProvider](#) on page 291
- [FileJobDestination](#) on page 297

HTTPJobDestination

A **HTTPJobDestination** sends process steps to a child Job server. The data transfer is carried out by HTTP.

Table 110: HTTPJobDestination parameters

Parameters	Description
Receiver port (ChildPort)	HTTP port of the child Job server.
ProviderID	Enter the name of the Job provider that will be used if more than one Job provider is being processed. If this is empty the first Job provider is used.
Retries	Number of retries performed by the module if the data transfer fails.
RetryDelay	Specifies how long the module will wait after a failed process step transfer before retrying. Timeout format: day.hour:minutes:seconds
Remote domain (RemoteDomain)	User account domain on the remote HTTP server.
Remote user account (RemoteUser)	User account for logging onto the HTTP server.
RemotePassword	Password for the user account for logging onto the HTTP server.

Related topics

- [HTTPJobProvider](#) on page 293

Configuration module

The standard One Identity Manager Service configuration settings are specified in this module.

Table 111: Configuration module parameters

Parameters	Description
VerboseLogging	Set the parameter to obtain more detailed messages on starting and stopping the One Identity Manager Service.
DebugMode	In DebugMode, One Identity Manager Service writes additional information to the log file. For example, all the parameters and results that are passed to a component are written to the log file. NOTE: This parameter is used for localizing errors. It

Parameters	Description
	is not recommended to set this parameter in normal working conditions on performance grounds.
ComponentDebugMode	<p>When set, individual One Identity Manager Service process components write additional process information to a log file.</p> <p>NOTE: This parameter is used for localizing errors. It is not recommended to set this parameter in normal working conditions on performance grounds.</p>
HTTP Header (HTTPHeader)	<p>HTTP header for status page. Pipe () delimited list of headers in the form: "name1: value1 name2: value2".</p> <p>Supported values are:</p> <ul style="list-style-type: none"> • X-Frame options: SAMEORIGIN • X-Content type options: nosniff • Content-Security-Policy: default-src 'self';script-src 'self' 'unsafe-inline';style-src 'self' 'unsafe-inline';img-src 'self' data:;font-src 'self' data: • X-XSS-Protection: 1; mode=block <p>Example:</p> <p>"X-Frame-Options: SAMEORIGIN X-Content-Type-Options:nosniff"</p>
HTTPAddress	<p>If One Identity Manager Service is running on a computer with several network cards, you can use this parameter to define which service should work over which IP address. If no IP address is entered, then all of them are used.</p>
HTTPPort	<p>Every One Identity Manager Service automatically works as an HTTP server. This parameter specifies the port that One Identity Manager Service works with. The default value is port 1880.</p> <p>The HTTP server is addressed by:</p> <p>http://<server name>:<port number></p>
Logging of Job provider and running instance (LogDestinationAndProviderId)	<p>Specifies whether the job provider ID and running instance are output in the log messages of the process step.</p>
Language	<p>Language used for error messages and outputs from the One Identity Manager Service. Permitted values are German and English. The default value is English.</p>

Parameters	Description
UseSSL	<p>Specifies whether the HTTP server is to provide secure connections. If this option is enabled, you can access the server from your browser using HTTPS.</p> <p>The One Identity Manager Service uses <code>System.Net.HttpListener</code> for the web interface. For detailed information on how to configure certificates, see How to: Configure a port with an SSL certificate.</p>
DoNotProtectCryptedValues	<p>Nomally, encrypted values from the <code>Jobservice.cfg</code> are additionally protected by the data protection API. This prevents use by other accounts or servers. This option switches of additional protection to use it on other cluster nodes, for example.</p> <p>NOTE: If you set this option, it causes problems if the database being synchronized against the One Identity Manager Service database is not encrypted. Therefore, ensure that database encryption is enabled.</p>
Timeout after failed start (WaitTimeOnFailedStart)	<p>The time to wait after a failed start before a retry is carried out. The default value is 90 seconds.</p> <p>Timeout format:</p> <p>hours:minutes:seconds</p>
Retries on failed start (RetriesOnFailedStart)	<p>Number of retries for the One Identity Manager Service to start up. The default value is 5.</p>
Do not protect private keys (DoNotProtectPrivateKeys)	<p>If the One Identity Manager Service finds a private key in the installation directory on startup, it places the key in the Windows internal key container of its service account and deletes the file from the hard drive. If this option is enabled, the key files are not moved to the key container.</p>
Do not write the configuration back to the database (DoNotWriteConfigBack)	<p>By default, the service configuration is written to the database. To prevent this, enable this option.</p>
Secrets allowed as replacements (SecretsAllowList)	<p>Comma-delimited list of secret names that are allowed as replacements in parameters. In the directory under <code>SecretsFolder</code>, there must be a file with the name of the secret that contains the value.</p> <p>Syntax:</p> <p><code>&SECRET(Name)&</code></p> <p>Example:</p>

Parameters	Description
	&SECRET(API_KEY)& In the %SECRETS% folder, there must be a API_KEY file containing the value.
Secrets folder (SecretsFolder)	Path the secret files' repository that can be used by the parameters. The path can take the form %Name%. Default value is %SECRETS% .

Logwriter module

This module writes the One Identity Manager Service messages. The following module types may be selected:

- [EventLogLogWriter](#)
- [FileLogWriter](#)

EventLogLogWriter

The **EventLogLogWriter** writes messages from the One Identity Manager Service to an event log. To view the event log, you can use the results display in the Microsoft Management Console, for example.

Table 112: EventLogLogWriter parameters

Parameters	Description
EventLog	Name of the event log to which the messages are written. The messages are written to the application log with Application as the default value. NOTE: If more than one One Identity Manager Service write event logs on a server, make sure that the first eight letters in the log name are unique on the server.
LogSeverity	Severity levels of the logged messages. Permitted values are: <ul style="list-style-type: none"> • Info: All messages are written to the event log. The event log quickly becomes large and confusing. • Warning: Only warnings and exception errors are written to the event log (default). • Serious: Only exception messages are written to the event log.
EventID	The ID of the messages written to the event log.

Parameters	Description
Category	The category of the messages written to the event log.
Source	The name of the source of the messages written to the event log.

By default, the One Identity Manager Service only logs messages in the event log **Application**.

To use an event log with a different name

1. On the Job server, manually add the file for the One Identity Manager Service to write to. You can use Windows PowerShell, for example, to do this.
 - a. Run Windows PowerShell as administrator on the Job server.
 - b. Run the following CmdLet:


```
New-EventLog -Source "Foobar" -LogName "<file name>"
```
2. Enter this file name in the One Identity Manager Service configuration file, in the module **EventLogWriter** as the name for the event log.
3. Restart the computer.
4. Restart the One Identity Manager Service.

Related topics

- [FileLogWriter](#) on page 306

FileLogWriter

The **FileLogWriter** writes messages from One Identity Manager Service to a log file. The log file can be displayed in a browser.

You call up the log file with the appropriate URL.

`http://<server name>:<port number>`

The default value is port 1880.

Table 113: FileLogWriter parameters

Parameters	Description
Log file (OutputFile)	<p>Name of the log file, including the directory name. Log information for the One Identity Manager Service is written to this file.</p> <p>IMPORTANT: The directory specified for the file must exist. If the file cannot be created, no error output is possible. Error messages then appear under Windows operating systems in the event log or under Linux operating systems in <code>/var/log/messages</code>.</p>

Parameters	Description
Log rename interval (LogLifeTime)	<p>In order to avoid unnecessarily large log files, the module supports the functionality of exchanging the log file with a history list. The LogLifeTime specifies the maximum life of a log file before it is renamed as backup. If the log file has reached its maximum age, the file is renamed (for example, as JobService.log_20040819-083554) and a new log file is started.</p> <p>Timeout format:</p> <p>day.hour:minutes:seconds</p>
Process step log lifetime (JobLogLifeTime)	<p>Retention time for process step logs. After this expires, the logs are deleted.</p> <p>Timeout format:</p> <p>day.hour:minutes:seconds</p> <p>For test purposes, you can enable logging of individual process steps in the Job Queue Info. The processing messages of the process step is written to a separate log with the Debug NLog severity. The files are stored in the log directory.</p> <p>Repository structure:</p> <p><Log directory>\JobLogs\<First 4 digits of the UID_Job>\Job_<UID_Job>_<yyyymmdd>_<Timestamp>.log</p>
Number of history logs (HistorySize)	Maximum number of log files. If several log files exist, the oldest backup file is deleted when a new log file is created so that the limit is not exceeded.
Max. log file size (MB) (MaxLogSize)	Maximum size in MB of the log file. Once the log file has reached the limit, it is renamed as a backup file and a new log file is created.
Max. length of parameters (ParamMaxLength)	Maximum number of characters allowed in a process step parameter so that they are written to the log file.
LogSeverity	<p>Severity levels of the logged messages.</p> <p>Permitted values are:</p> <ul style="list-style-type: none"> • Info: All messages are written to the event log. The event log quickly becomes large and confusing. • Warning: Only warnings and exception errors are written to the event log (default). • Serious: Only exception messages are written to the event log.
Add server name (AddServerName)	Specifies whether the server name is to be added to the log entries.

Dispatcher module

In a hierarchical server structure a server can be used as a proxy server for other servers. The proxy server makes requests at set time intervals for process steps to be processed on a server and sends them to the next server. If the request load needs to be minimized, a proxy server is recommended.

Table 114: Dispatcher module parameters

Parameters	Description
Acts as proxy for other servers (IsProxy)	Specifies whether the server is to act as a proxy server. Set this option if the server should be a proxy server.
ProxyInterval	Time interval in seconds, after which the proxy server acting as deputy for another server, should renew a request to the database.

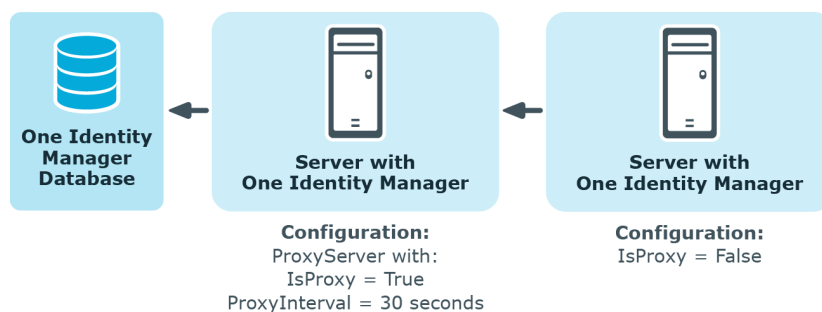
The following guidelines can be used as orientation for the configuration of One Identity Manager Service polling intervals in a cascading environment:

Table 115: Polling interval guidelines for One Identity Manager Service

Parameters	Root Server (direct connection to database)	Leaf server (connected by HTTP or file)
JobServiceDestination.StartInterval	90 seconds	600 seconds
JobServiceDestination.Statisticinterval	360 seconds	600 seconds
Dispatcher.ProxyInterval	180 seconds	
Dispatcher.IsProxy	True	False

The proxy mode of a root server ensures that, acting on behalf of the leaf server, process steps are queried in shorter proxy intervals. When the root server is restarted, it may take a while until all leaf servers have sent their first request (in this case a maximum of 600 seconds). However, the system then swings into action.

Figure 29: Dispatcher configuration example



Connection module

With this module you can set special configuration settings for the behavior of the One Identity Manager Service.

Table 116: Connection module parameters

Parameters	Description
Process generation log directory (JobGenLogDir)	Directory of log files in which the instructions for process generation generated by One Identity Manager Service are recorded.
Disable reload beep (NoReloadBeep)	When this parameter is set the beep is switched off that is made when buffered dialog data is loaded.
Log BLOB reads (LogBlobReads)	Specifies whether read operations on text and binary LOB (BLOB) should be written to the SQL log.
Cache type (CacheType)	Specifies how the data is cached. The default value is MultipleFiles .
Cache reload interval (CacheReloadInterval)	Time in seconds after which the local cache should be updated. This parameter overwrites the setting in the Common CacheReload Interval configuration parameter.
Regular expression for stack trace positions (ObjectDumpStackExpression)	<p>This expression specifies when an extra stack trace is written to the object log. If the current row in the object log matches the regular expression, the stack trace is written in the object log.</p> <p>Sample expression: "Lastname"</p> <p>If the current row contains the value "Lastname", the stack trace is also copied to the log.</p> <p>NOTE: This parameter is used for localizing errors. It is not recommended to set this parameter in normal working conditions on performance grounds.</p>
TokenCertificateThumbprint	Thumbprint of the certificate used to verify the security token.
TokenCertificateFile	Certificate file of the certificate to be used to verify the security token. The certificate must support RSA encryption with SHA1, SHA256, or SHA512 and contain the private key.
Supports read-only replicas in Azure (SupportReadScaleOut)	Specifies whether a second pool for read-only queries is supported in Azure. If the option is set, read-only queries are supported. This feature is available in Azure's Premium and Business Critical plans. For more

Parameters	Description
	information, see https://docs.microsoft.com/en-us/azure/azure-sql/database/read-scale-out .
Connect directly without availability check (DirectConnection)	Specifies whether to connect directly to the target database without testing availability or status first. This allows tools that do not allow database switching within the connection, to trace the connection. NOTE: This option can affect migration because the connection is always open.

HTTP authentication module

Every One Identity Manager Service automatically works as an HTTP server. Which services the One Identity Manager Service provides depends on the plugins configurations. Use this module to specify how authentication works on an HTTP server so that other services can be accessed, for example, displaying the log file or the status display.

The following module types may be selected:

- BasicHttpAuthentication
Use this authentication type to specify a user account and the corresponding password for accessing the HTTP server.
- WindowsHttpAuthentication
Use this authentication type to specify an Active Directory group, whose users can be authenticated on the HTTP server. A security ID (SID) or the Active Directory group name in the domain of the Job server can be specified. If Active Directory is not located in the domain of the Job server, the SID must be used.

NOTE: If a module is not specified, authentication is not required. In this case, all users can access the services.

Module plugins

Plugins are program classes that One Identity Manager Service loads and that extend the functionality of the service. The following plugins are available:

- [HTTPLogPlugin](#)
- [ScheduleCommandPlugin](#)
- [RequestWatchDogPlugin](#)
- [PerformanceCounterPlugin](#)
- [DebugMailPlugin](#)

- [ShareInfoPlugin](#)
- [RemoteConnectPlugin](#)
- [DatabaseAgentPlugin](#)

HTTPLogPlugin

The plugin writes a log file that records the One Identity Manager Service HTTP requests. Enter the following parameter:

- **Output file (LogFile)**
Enter the name of the file that is to record the messages. The file is written in Apache HTTP Server Combined Log Format.

ScheduleCommandPlugin

This plugin calls up an external program in regular intervals. This is useful, for example, when process steps need to be routed over their own transfer methods.

Table 117: ScheduleCommandPlugin parameters

Parameters	Description
Command to run (Command)	Command to be run including command line option This command is run as a <code>cmd</code> and therefore built-in commands are possible.
Service start command (StartCommand)	Command run when the One Identity Manager Service is started
Service start command (StopCommand)	Command run when the One Identity Manager Service is stopped
Interval between runs (Interval)	Interval (in seconds) at which the command should be called While the command is running, the timer is stopped so that the calls do not overlap. The default value is 60 .
Command output to log file (OutputToLog)	Specifies whether the command outputs are logged if successful. If this parameter is set, the command output is also written to the One Identity Manager Service's log file when successful. If the parameter is disabled, only errors are written to the log file.
Severity level (LogSeverity)	Message types used for messages that appear in the log file when the transaction is successful. Permitted values are Info , Warning , and Serious . The default value is Info .

RequestWatchDogPlugin

This plugin restarts One Identity Manager Service when less than a defined number of requests are made within a specified interval.

Table 118: RequestWatchDogPlugin parameters

Parameters	Description
Action	Action to be run when there is a break in the queries. Permitted values are: <ul style="list-style-type: none">• Restart (default): Messages are logged. This restarts the service.• Log: Messages are logged. The service is not restarted.
Monitoring interval Interval	Monitoring interval in seconds. The default value is 600 .
Minimum number of requests (MinRequests)	Minimum number of requests that must be made within the specified interval. The default value is 2 .

PerformanceCounterPlugin

This plugin exports the One Identity Manager Service status values as performance counter. Monitoring through a system monitor is thus enabled. The list of currently available performance counters is displayed under `http://<servername>:1880/PerfCounter`.

Table 119: PerformanceCounterPlugin parameters

Parameters	Description
Value types to specify (CounterType)	Value types provided as performance counters Int and Long values are directly output, while TimeSpan values are output as Long values (numbe of milliseconds).
Polling interval (PollingInterval)	Interval (in seconds) at which the performance counters are exported
Category	Category under which the performance counters of the One Identity Manager Service are displayed. This information is required if several One Identity Manager Services with this plugin are active on the server.

NOTE: If, after restarting the One Identity Manager Service, an error of the type At least one service could not be started occurs, enter the WMI Performance Adapter service as dependent on the One Identity Manager Service.

DebugMailPlugin

If this plugin is enabled, email notifications generated by the One Identity Manager Service are not sent but are kept in a drop folder. The file names contain the time stamp in this case. If a mail contains HTML text, a *.html file is saved with the same name as the descriptive *.txt file with the body. Email attachments are also saved in this way.

NOTE:

- The plugin only works for processes run internally in the One Identity Manager Service.
- If this plugin is enabled, no email notifications are sent through the One Identity Manager Service. This plugin is only used for localizing errors. It is not recommended to set this parameter in normal working conditions.

Enter the following parameter:

- Drop folder (DropFolder)
Directory for storing email notifications.

ShareInfoPlugin

This plugin is required for solving Samba shares (smb.conf) under the Linux operating system. The plugin solves UNC paths to local paths. This plugin does not required any parameters.

NOTE: Install the plugin if the One Identity Manager Service runs copy actions between servers with Linux operating system.

RemoteConnectPlugin

To configure synchronization with a target system, One Identity Manager must load the data from the target system. One Identity Manager communicates directly with the target system to do this. Sometimes direct access from the workstation, on which the Synchronization Editor is installed, is not possible. For example, because of the firewall configuration or the workstation does not fulfill the necessary hardware and software requirements. If direct access is not possible from the workstation, you can set up a remote connection. Prerequisite for this is that the **RemoteConnectPlugin** is installed on the Job server.

Table 120: RemoteConnectPlugin parameters

Parameters	Value	Description
Authentication method (AuthenticationMethod)	ADSGroup	Method with which incoming queries can be authenticated.

Parameters	Value	Description
Permitted AD group (ADGroupAuthPermittedGroup)		Permitted values: ADGroup Distinguished name or object SID of the Active Directory group whose members are permitted to use a remote connection. This parameter is only required for the ADGroup authentication method.
Port (Port)	2880	Port for reaching the server.

NOTE: Authentication of a remote connection can only be done through an Active Directory group.

DatabaseAgentPlugin

This One Identity Manager Service plugin deploys the Database Agent Service. The Database Agent Service controls processing of DBQueue Processor tasks. The plugin should be configured on the Job server that performs the **Update server** server function. An administrative user must be used for the database connection in the Job provider.

Enter the following parameter:

- Job provider IDs (ProviderIDs)
IDs of the Job providers to be used. Enter a list of job provider names separated by the pipe symbol (|). If this is empty the first Job provider is used. If * is specified, all Job providers are used.

Related topics

- [MSSQLJobProvider](#) on page 288
- [Processing DBQueue Processor tasks by the Database Agent Service](#) on page 506

File module with private key

In this module, you provide the data for files with a private key. Use this parameter if you work with several private keys, for example, if One Identity Manager Service data must be exchanged between two encrypted One Identity Manager databases.

If no key is entered here, the private key file from the **File with private key** (PrivateKey) parameter of the **JobServiceDestination** is used.

To enter a file with a private key

1. Click **New** and enter the following information:
 - **Property:** Enter the ID of the private key. The ID is expected in the **JobServiceDestination** in the **Private key identifier** parameter (PrivateKeyId). The default key has the ID **Default**.
 - **Value:** Enter the path of the private key file. You can enter the absolute or relative path to the One Identity Manager Service.

Example: Configuration in the file jobservice.cfg.

```
configuration>
  <category name="privatekeys">
    <value name="Default">private.key</value>
    <value name="Key2">key2.key</value>
    <value name="OtherKey">C:\Path\To\Other.key</value>
  </category>
</configuration>
```

Related topics

- [JobServiceDestination](#) on page 295

Tracking changes with process monitoring

With One Identity Manager, it is possible to create a change history for objects and their properties. This can be used to fulfill reporting duties for internal committees and legal obligations for providing documentary evidence. Different methods can be used to track changes within One Identity Manager. With this combination of methods, all changes that are made in the One Identity Manager system can be traced.

- Recording data modifications

Modifications to data can be recorded for add or delete operations on objects, and up to and including changes to individual object properties.

- Recording process information

Recording process information allows all processes and process steps to be tracked while being processed by One Identity Manager Service.

- Recording messages in the process history

In the process history, success, and error messages from handling each process step in the Job queues are recorded by the One Identity Manager Service.

All entries logged in One Identity Manager are initially saved in the One Identity Manager database. The proportion of historical data to total volume of a One Identity Manager database should not exceed 25 percent. Otherwise, performance problems may arise. You must ensure that log entries are regularly removed from the One Identity Manager database and archived. For more information about archiving data, see the *One Identity Manager Data Archiving Administration Guide*.

Detailed information about this topic

- [Basic rules for process monitoring](#) on page 317
- [Logging data changes](#) on page 318
- [Logging process information during process handling](#) on page 320
- [Recording messages in the process history](#) on page 324
- [Archiving and deleting records](#) on page 330

Basic rules for process monitoring

To use process monitoring in One Identity Manager.

1. In the Designer, check if the **Common | ProcessState** configuration parameter is set. If not, set the configuration parameter.

If the configuration parameter is set, you can configure process monitoring. In addition, the process view is enabled in the Manager.

2. You can control the extent of the logging using the configuration settings for each method.

The methods implemented by One Identity Manager allow monitoring of all modifications to the system that are triggered by a user action. Each action in One Identity Manager is labeled with a unique ID number. This ID number is called a GenProcID. All changes that can be traced back to the same cause are given the same GenProcID and are grouped in this way. If a previously stored action does not pass a GenProcID to the current action, a new ID is automatically created.

If an action is triggered from the One Identity Manager's object layer, the GenProcID is written to the context data of the database connection. The logged in user is also noted in the context data and is made available in this way.

A new GenProcID is generated by the trigger if an action takes place directly in the database or through an application that works without the One Identity Manager object layer. This GenProcID is valid for the duration of the database connect, which means that all changes belong to the same action and link to the same GenProcID. The user data is made up of the database user's name, the MAC address and the workstation name as well as the application name.

All actions (process triggers) that cause changes to the system, and their actual status information, are logged internally in the DialogProcess status table. Logging takes place independent of the chosen change history method. This log writing therefore provides a starting point for monitoring and allows the changes based on one action to be grouped together.

The following information is recorded for one action:

- ID number (GenprocID)
- Display name for the action
- Base object that the action is triggered for
- User that triggered the action
- Time of action
- Object key for selecting the process trigger
- Comment on the action
- Current process status

NOTE: The information is displayed in the Manager in the process view. For more information, see the *One Identity Manager Operational Guide*.

Detailed information about this topic

- [Logging data changes](#) on page 318
- [Logging process information during process handling](#) on page 320

Logging data changes

NOTE: The information is displayed in the Manager in the process view. For more information, see the *One Identity Manager Operational Guide*.

To log data changes

- In the Designer, check whether the **Common | ProcessState** configuration parameter is set. If not, set the configuration parameter.
- In the Designer, set the **Common | ProcessState | PropertyLog** configuration parameter and compile the database.

When this configuration parameter is set, changes to individual values are logged and shown in the process view in the Manager.

If you disable the configuration parameter at a later date, model components and scripts that are not longer required, are disabled. SQL procedures and triggers are still carried out. For more information about the behavior of preprocessor relevant configuration parameters and conditional compiling, see the [Conditional compilation using preprocessor conditions](#) on page 335.

- (Optional) To log changes for the user data part to properties that belong to an alternative key, in the Designer, set the **Common | ProcessState | PropertyLog | AutoTrackAlternatePK | Payload** configuration parameter.
- (Optional) To log changes for the user data part to properties that belong to an alternative key, in the Designer, set the **Common | ProcessState | PropertyLog | AutoTrackAlternatePK | Payload** configuration parameter.
- Label columns for which changes will be logged.
- Label columns to be logged when an object is deleted.

TIP: If you set the **Common | ProcessState | PropertyLog | AllDefaultPropertiesForModel** configuration parameter in the Designer, One Identity Manager schema columns are already labeled for logging changes and deletions. Define which columns are affected in the `QBMVDefaultHistoryColumns` table.

Add, change, and delete operations can be recorded for objects. The GenProcID trigger is also passed down so that the changes to one object can be grouped together. The data changes are stored in the `DialogWatchOperation` and `DialogWatchProperty` tables. An entry is also created in the status `DialogProcess` table for the triggering action.

The following information is collected for these operations:

- Adding an object

If a new object is added, the object key, object display name, date of insertion, and user are logged.

- Changing an object

If a column is changed the old value, change date, and user are logged. Depending on the **Common | ProcessState | PropertyLog | AutoTrackAlternatePK** and **Common | ProcessState | PropertyLog | AutoTrackAlternatePK | PayLoad** configuration parameters, changes to properties belonging to an alternative key are logged.

- Deleting an object

If an object is deleted, the columns to be logged are all primary key columns are logged. The value, deletion date and user are logged.

Related topics

- [Labeling columns for recording changes to data](#) on page 319
- [Basic rules for process monitoring](#) on page 317
- [Logging process information during process handling](#) on page 320

Labeling columns for recording changes to data

TIP: If you set the **Common | ProcessState | PropertyLog | AllDefaultPropertiesForModel** configuration parameter in the Designer, One Identity Manager schema columns are already labeled for logging changes and deletions. Define which columns are affected in the QBMVDefaultHistoryColumns database view.

To label a column for recording

1. In the Designer, select the **One Identity Manager schema** category.
2. Select the table and start the Schema Editor with **Show table definition**.
3. Select the column and then the **Column properties** view.
4. Select the **Miscellaneous** tab and edit the following properties.
 - **Log changes:** Set this option to log changes to data in the column.
 - **Log changes when deleting:** Set this option to record the column when the object is deleted.

Related topics

- [Logging data changes](#) on page 318
- [Column definition properties](#) on page 91

Logging process information during process handling

NOTE: The information is displayed in the Manager in the process view. For more information, see the *One Identity Manager Operational Guide*.

To log process information

- In the Designer, check if the **Common | ProcessState** configuration parameter is set. If not, set the configuration parameter.
- In the Designer, check if the **Common | ProcessState | ProgressView** configuration parameter is set. If not, set the configuration parameter. Select the scope of logging through the configuration parameter option.

Permitted values are:

- **1:** Full process tracking Process information from all processes marked for process tracking is logged.
- **2:** Web Portal tracking Only process information for process marked for process tracking the Web Portal is logged. (default)
- Label the process and process steps for process tracking and define templates for event, process, and process step process information.

In the Designer, use the Process Editor to set up templates for creating process information for processes, process steps, and events. Use #LD notation for language-dependent definition of process information.

If the **Common | ProcessState | ProgressView** configuration parameter is enabled, the Job generator creates entries in the status tables during process generation for processes, process steps, and events with process information.

Right at the start, the Job Generator uses the GenProcID for the generating operation. If there is no GenProcID passed at runtime, a new one is automatically created. This ID is written to the **GenProcID** global variable for the current database connection object before the process is generated. It can, therefore, be used by all processes. All partial steps that are triggered by a generating operation are grouped together in this way and logged. Bulk operations, such as synchronization and CSV import, are an exception. In this case, a new GenProcID is created for each individual step in tracking the object changes and not for the process as a whole.

An entry is set up in the DialogProcessStep status table for each process step that is marked for tracking. For each process that has at least one such process step, an entry is made in the DialogProcessChain status table. For each generating operation that has caused an entry in the DialogProcessChain status table, an entry is written to the DialogProcess status table. At the same time, the Job Generator creates the display name for the process view by running the given VB.Net expression for the process information.

The possible processing states and additional information available for the respective processing statuses are listed in the following tables.

Table 121: Possible process states

Process State	Description
Initial	<generated> ::= "G"
End of processing	<finalstate> ::= <ended> <failed> <not run> where: <ended> ::= "E" (processing successful) <failed> ::= "F" (processing unsuccessful) <not run> ::= "N" (no longer accessible during processing)
In progress	<workingstate> ::= <delayed> <processing> [<ProcessStateAddON>] where: <delayed> ::= "D" (processing delayed) <Long delayed> ::= "L" (processing was put on hold) <processing> ::= "P" (in progress) <ProcessStateAddON> (optional additional information)

Table 122: Possible additional information

Additional Information	Description
Processing deferred until	<datetime> ::= <YYYY> - <MM> - <DD> <HH> : <NN> : <SS> where: <YYYY> ::= 1980..9999 <MM> ::= 01..12 <DD> ::= 01..31 <HH> ::= 00..23 <NN> ::= 00..59 <SS> ::= 00..59
Retries	<retryinfo> ::= 1..99

Related topics

- [Editing process information for processes](#) on page 322
- [Editing process information for process steps](#) on page 322
- [Editing process information for events](#) on page 323
- [Basic rules for process monitoring](#) on page 317
- [Logging data changes](#) on page 318

Editing process information for processes

To edit process information for a process

1. In the Designer, select the process in the **Process Orchestration** category.
2. Start the Process Editor with the **Edit process** task.
3. Click on the element for the process in the process document.
4. On the **General** tab in the **Process properties** view, edit the following information.
 - **Process information:** Select how the process information should be logged.
Permitted values are:
 - **None:** The process information is not logged.
 - **Full process tracking:** The process information is logged and displayed in the Manager.
 - **Web Portal tracking:** The process information is logged and displayed in the Manager and the Web Portal.
5. Enter the following information in the **Process properties** view on the **Process tracking** tab.
 - **Process information:** Value template for the process information as VB.NET term. Use #LD notation for language-dependent definition of process information.

Related topics

- [Using #LD-notation](#) on page 354
- [Process properties](#) on page 230
- [Editing process information for process steps](#) on page 322
- [Editing process information for events](#) on page 323

Editing process information for process steps

To edit process information for a process

1. In the Designer, select the process in the **Process Orchestration** category.
2. Start the Process Editor with the **Edit process** task.
3. Click on the element for the process step in the process document.
4. In the **Process step properties** view on the **General** tab, edit the following information.

- **Process information:** Select how the process information should be logged.

Permitted values are:

- **None:** The process information is not logged.
- **Full process tracking:** The process information is logged and displayed in the Manager.
- **Web Portal tracking:** The process information is logged and displayed in the Manager and the Web Portal.

5. In the **Process step properties** view on the **Process tracking** tab, enter the following information.

- **Depth of detail:** Select the level of detail of the process information. You can choose from: **basic information**, **extended information** and **full information**.

You use depth of detail to control how process information is displayed in the Manager's process view. Depending on the Manager's program settings, differing levels of detail are offered to the user on views of the process information. For more information, see the *One Identity Manager Operational Guide*.

- **Process information:** Enter the value template for the process information as VB.NET term. Use #LD notation for language-dependent definition of process information.


Related topics

- [Using #LD-notation](#) on page 354
- [Process step properties](#) on page 231
- [Editing process information for processes](#) on page 322
- [Editing process information for events](#) on page 323

Editing process information for events

IMPORTANT: At least one event process must have process tracking enabled in order to generate process information for events.

To edit process information for events

1. In the Designer, select the process in the **Process Orchestration** category.
2. Start the Process Editor with the **Edit process** task.
3. Click on the element for the process in the process document.
4. In the **Events** view, select the event and click .
5. Enter the following information.

- **Event process information:** Value template for the process information as VB.NET term. Use #LD notation for language-dependent definition of process information.

If there no template is available, the information is evaluated as follows:

<table> - <event> - <object display name>

If several processes point to one event, the event with a process information template is found that has the lowest generating order specified in its process configuration. This template is evaluated and shown in the process view in the Manager. For more information, see the *One Identity Manager Operational Guide*.

Related topics

- [Using #LD-notation](#) on page 354
- [Creating events for processes](#) on page 239
- [Editing process information for processes](#) on page 322
- [Editing process information for process steps](#) on page 322

Recording messages in the process history

In the process history (JobHistory table), the processes being handled are logged. You can analyze the process history in Job Queue Info. For more information, see the *One Identity Manager Process Monitoring and Troubleshooting Guide*.

To log messages to the process history

- In the Designer, check if the **Common | ProcessState** configuration parameter is set. If not, set the configuration parameter.
- In the Designer, check if the **Common | ProcessState | ProgressView** configuration parameter is set. If not, set the configuration parameter. Select the scope of logging through the configuration parameter option.

Table 123: Permitted values of the “Common | ProcessState | JobHistory” configuration parameter

Value	Meaning
NO	No messages are logged in the process history.
ALL	All process steps being handled are logged in the process history.

Value	Meaning
ERROR	On failed process steps are logged in the process history.
ERRORorSELECTED	Failed process steps and process steps labeled with the Process history option are logged in the process history.
SELECTED	Only process steps labeled with the Process history option are logged in the process history.

- Use the **Common | ProcessState | JobHistory | TrimLongParameters** configuration parameter to specify the length of process parameter values that are logged in the process history.

Log entries in the process history are exported from the One Identity Manager database at regular intervals. One Identity Manager provides various methods to do this. For more information, see [Archiving and deleting records](#) on page 330.

Process tracking for DBQueue Processor operations

In order to track inherited calculations as a result of changes to the system, the GenProcID is always passed to the DBQueue Processor operation. There may only be one entry in the DBQueue for each operation and object in case of follow-on operations. To map such processes, a new GenProcID is issued and used in subsequent processes. The conflicting processes and their GenProcID's are saved in the DialogProcessSubstitute table.

When a new GenProcID is created for conflicting processes, the following rules apply:

- Several of the same DBQueue Processor operations on one object are merged into one process (one GenProcID). This uses existing substitute processes if the number is identical to the predecessor (with respect to the root processes).
- If further conflicts occur in the sequence, the GenProcIDs that have already been replaced are reset to the original and a new substitute is created.
- A substitute is only valid for one set of original processes.

The **QBM | DBQueue | GenProcIDReplaceLimit** configuration parameter defines the limit for process substitutions. The maximum number of conflicting processes are mapped in the DialogProcessSubstitute table. If necessary, you can set the configuration parameter in the Designer and change the value.

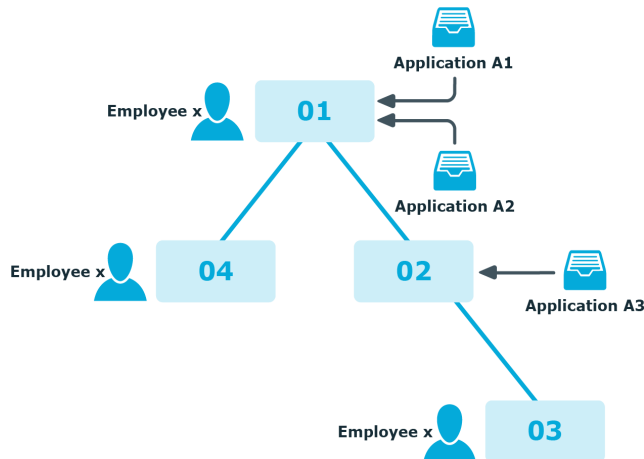
Related topics

- [Example of replacing the GenProcID](#) on page 326
- [Processing DBQueue tasks](#) on page 501

Example of replacing the GenProcID

A hierarchical role structure exists which consists of 4 roles O1, O2, O3, and O4. Employee X is assigned to roles O1, O4, and O3. The assignment of software to roles is depicted in the following.

Figure 30: Role structure as in the example above



Three processes run between two DBQueue Processor runs, each with its own GenProcID:

- P1: Software application A1 is assigned to the role O1
- P2: Software application A2 is assigned to the role O1
- P3: Software application A3 is assigned to the role O2

The following operations are in the DBQueue (DialogDBQueue table) and in the process information:

Operation	Object	GenProcID
OrgHasApp	O1	P1
OrgHasApp	O1	P2
OrgHasApp	O2	P3

The operation OrgHasApp cannot be subdivided with respect to O1 because the union of software applications is being calculated for O1. At this point, no more information is available as to which GenProcID has been entered by the assignment for which software application.

In order to achieve uniqueness for the combination of operation and object, a new GenProcID P4 is introduced and the two O1 operations are compacted into this GenProcID. P1 and P2 are noted in the DialogProcessSubstitute table as possible predecessors of P4 (but not clearly in the individual actions).

Operation	Object	GenProcID
OrgHasApp	O1	P4
OrgHasApp	O2	P3

The following constellations can occur depending on whether the operation OrgHasApp is processed as a single step or in bulk:

- Case 1) O1 is calculated and then O2.
- Case 2) O2 is calculated and then O1.
- Case 3) O1 and O2 are calculated together simultaneously in a bulk operation.

After these operations have been run and assuming that they all cause changes to the total sets affected, the following situation arises:

Case 1) O1 is calculated and then O2.

Operation	Object	GenProcID
OrgHasApp	O2	P3
OrgHasApp	O4	P4
OrgHasApp	O2	P4
OrgHasApp	O3	P4
PersonHasApp	X	P4

Before the next DBQueue Processor run, the GenProcID's must be compressed again, because the OrgHasApp operation did not produce a unique result for the object O2. P5 is introduced with possible predecessors P4 and P3.

Operation	Object	GenProcID
OrgHasApp	O2	P5
OrgHasApp	O4	P4
OrgHasApp	O3	P4
PersonHasApp	X	P4

Now the calculation is done for O2:

Operation	Object	GenProcID
OrgHasApp	O3	P5
PersonHasApp	X	P5

Operation	Object	GenProcID
OrgHasApp	O4	P4
OrgHasApp	O3	P4
PersonHasApp	X	P4

Because O3 is not unique, P6 is introduced with possible predecessors P4 and P5.

Operation	Object	GenProcID
OrgHasApp	O3	P6
PersonHasApp	X	P5
OrgHasApp	O4	P4
PersonHasApp	X	P4

After O3 and O4 have been calculated, the following situation exists:

Operation	Object	GenProcID
PersonHasApp	X	P6
PersonHasApp	X	P5
PersonHasApp	X	P4

There is no uniqueness for object X such that P7 is introduced with possible predecessors P4, P5 and P6.

Case 2) O2 is calculated and then O1.

Operation	Object	GenProcID
OrgHasApp	O1	P4
OrgHasApp	O2	P3

After running, the following entries are in the DBQueue:

Operation	Object	GenProcID
OrgHasApp	O1	P4
OrgHasApp	O3	P3

The following situation is the result after the next step:

Operation	Object	GenProcID
OrgHasApp	O3	P3
OrgHasApp	O4	P4
OrgHasApp	O2	P4
OrgHasApp	O3	P4
PersonHasApp	X	P4

To achieve uniqueness for O3 a process P5 with possible predecessors P3 and P4 is created:

Operation	Object	GenProcID
OrgHasApp	O3	P5
OrgHasApp	O4	P4
OrgHasApp	O2	P4
PersonHasApp	X	P4

After the calculations, the following situation exists:

Operation	Object	GenProcID
PersonHasApp	X	P5
PersonHasApp	X	P4

There is no uniqueness for object X such that P6 is introduced with possible predecessors P4 and P5.

Case 3) O1 and O2 are calculated together simultaneously in a bulk operation.

Operation	Object	GenProcID
OrgHasApp	O1	P4
OrgHasApp	O2	P3

After the first step in the calculation the following entries are in the DBQueue:

Operation	Object	GenProcID
OrgHasApp	O4	P4

Operation	Object	GenProcID
OrgHasApp	O2	P4
OrgHasApp	O3	P4
OrgHasApp	O3	P3
PersonHasApp	X	P4

Uniqueness is achieved for O3 by introducing P5 with possible predecessors P3 and P4:

Operation	Object	GenProcID
OrgHasApp	O4	P4
OrgHasApp	O2	P4
OrgHasApp	O3	P5
PersonHasApp	X	P4

After the next step in the calculation, the following content is found

Operation	Object	GenProcID
OrgHasApp	O3	P4
PersonHasApp	X	P4
PersonHasApp	X	P5

After O3 has been calculated in the next run and has not created a new PersonHasApp entry, only X exists with P4 and P5 because X already exists with P4.

Operation	Object	GenProcID
PersonHasApp	X	P4
PersonHasApp	X	P5

There is no uniqueness for object X such that P6 is introduced with possible predecessors P4 and P5.

Archiving and deleting records

All entries logged in One Identity Manager are initially saved in the One Identity Manager database. The proportion of historical data to total volume of a One Identity Manager database should not exceed 25%. Otherwise, performance problems may arise. You must

ensure that log entries are regularly removed from the One Identity Manager database and archived.

The following methods are provided for regularly removing recorded data from the One Identity Manager database:

- The data can be transferred directly from the One Identity Manager database into a One Identity Manager History Database. This is the default procedure for data archiving. Select this method if the servers on which the One Identity Manager database and the One Identity Manager History Database are located have network connectivity.
- The data is deleted from the One Identity Manager database after a certain amount of time without being archived.

For detailed information about setting up archiving of data in a History Database, see *One Identity Manager Data Archiving Administration Guide*.

Detailed information about this topic

- [Deleting log entries in the One Identity Manager database without archiving](#) on page 331
- [Specifying log retention times](#) on page 332
- [Optimizing performance by deleting log entries](#) on page 333

Deleting log entries in the One Identity Manager database without archiving

If records from separate sections are kept in the One Identity Manager database for a certain amount of time but are not archived later, you have the following options:

- To exclude a certain section from archiving, do not configure it for export, just specify a retention period.
- To delete all sections without archiving, specify a retention period. In the Designer, set the **Common | ProcessState | ExportPolicy** configuration parameter and enter the value **NONE**.

The records are deleted from the One Identity Manager database by DBQueue Processor when the retention period has ended. In addition, all entries for triggered actions are deleted if they have no corresponding records in those sections.

NOTE: If you do not specify a retention period, the records from that section are deleted from the One Identity Manager database during daily DBQueue Processor maintenance tasks.

Related topics

- [Specifying log retention times](#) on page 332
- For more information, see [Optimizing performance by deleting log entries](#) on page 333.

Specifying log retention times

Once the retention period has ended, the recorded data is either exported or deleted from the One Identity Manager database depending on which archiving method has been chosen. A longer retention period should be selected for sections whose records will be exported than for those that will be deleted.

NOTE: If you do not specify a retention period, the records in this section will be deleted daily from the One Identity Manager database within the daily DBQueue Processor maintenance tasks.

The recordings are not exported until the retention period for all sections has expired and no other active processes for the process group (GenProcID) exist in the DBQueue, process history, or as scheduled operation.

You use configuration parameters to define the data retention periods for the individual sections.

Table 124: Configuration parameter for handling change data

Configuration parameter	Meaning
Common ProcessState PropertyLog IsToExport	Exports the data changes. If this configuration parameter is not set the information is deleted once the retention period has expired.
Common ProcessState PropertyLog LifeTime	Maximum retention period in days of logged data changes in the database. The default value is 30 .

Table 125: Configuration parameter for handling process information

Configuration parameter	Meaning
Common ProcessState ProgressView IsToExport	Exports the data in the process information. If this configuration parameter is not set the information is deleted once the retention period has expired.
Common ProcessState ProgressView LifeTime	Maximum retention period in days of process information in the database. The default value is 30 .

Table 126: Configuration parameter for handling process history

Configuration parameter	Meaning
Common ProcessState JobHistory IsToExport	Exports the information in the process history. If this configuration parameter is not set the information is deleted once the retention period has expired.
Common ProcessState JobHistory LifeTime	Maximum retention period in days of logged process history in the database. The default value is 30 .

Optimizing performance by deleting log entries

If there is a large amount of data, you can specify the number of objects to delete per DBQueue Processor operation and run in order to improve performance. You use configuration parameters to make the choice for each section.

Table 127: Configuration parameters for deleting logged data changes

Configuration parameter	Meaning
Common ProcessState PropertyLog Delete	Allows configuration of deletion behavior for logged data changes.
Common ProcessState PropertyLog Delete BulkCount	Number of entries to be deleted in any operation. The default value is 200 .
Common ProcessState PropertyLog Delete TotalCount	Total number of entries to be deleted in any processing run. The default value is 10000 .

Table 128: Configuration parameters for deleting process information

Configuration parameter	Meaning
Common ProcessState ProgressView Delete	Allows configuration of deletion behavior for process information.
Common ProcessState ProgressView Delete BulkCount	Number of entries to be deleted in any operation. The default value is 200 .
Common ProcessState ProgressView Delete TotalCount	Total number of entries to be deleted in any processing run. The default value is 10000 .

Table 129: Configuration parameters for deleting process history

Configuration parameter	Meaning
Common ProcessState JobHistory Delete	Allows configuration of deletion behavior for the process history.
Common ProcessState JobHistory Delete BulkCount	Number of entries to be deleted in any operation. The default value is 200 .
Common ProcessState JobHistory Delete TotalCount	Total number of entries to be deleted in any processing run. The default value is 10000 .

Table 130: Configuration parameters for deleting process status entries

Configuration parameter	Meaning
Common ProcessState Delete	Allows configuration of deletion behavior for process status entries.
Common ProcessState Delete BulkCount	Number of entries to be deleted in any operation. The default value is 500 .
Common ProcessState Delete TotalCount	Total number of entries to be deleted in any processing run. The default value is 10000 .

Conditional compilation using preprocessor conditions

Conditional compiling of program code is integrated into One Identity Manager. Conditional compilation allows parts of the program code to be parsed whereas other parts remain untouched.

Conditional compiling has the following advantages:

- Assemblies are reduced in size
- Structures the system configuration
- Improves clarity for the model and permissions
- Speeds up processing
- Hides unnecessary data in all VB.Net expressions
- Hides unnecessary model components

Conditional compiling in One Identity Manager is controlled using preprocessor conditions. Preprocessor conditions can be used in:

- Objects with the **Preprocessor condition** property.
- VB.Net expressions

NOTE: Preprocessor conditions help stop the module being disabled. For example, SQL procedures and triggers are still run even if the objects are disabled by preprocessor conditions. To disabled a module, remove the module from the One Identity Manager database. For more information about removing modules, see the *One Identity Manager Installation Guide*.

Configuration parameters and their options define the possible preprocessor conditions. In order to become effective on a system-wide basis, every modification to preprocessor relevant configuration parameters as well as modifications to preprocessor conditions on objects and VB.Net expressions requires the One Identity Manager database to be recompiled.

Detailed information about this topic

- [Preprocessor-relevant configuration parameters](#) on page 336
- [Preprocessor conditions in objects](#) on page 337
- [Preprocessor conditions in VB.Net expressions](#) on page 338
- [Evaluation of preprocessor conditions during compilation](#) on page 339

Preprocessor-relevant configuration parameters

IMPORTANT: The One Identity Manager database needs to be recompiled every time a preprocessor-relevant configuration parameter and its options are changed.

The **Preprocessor relevant parameter** option is used to label a configuration parameter as preprocessor relevant. A preprocessor expression is entered in the associated configuration parameter option.

When a preprocessor relevant configuration parameter is set it is valid globally across the system. The preprocessor condition does not come into effect until the database has been compiled.

NOTE: Predefined preprocessor configuration parameters are overwritten during schema installation. Define company-specific, preprocessor-relevant configuration parameters and options in the Designer under the **Custom** configuration parameter.

To display preprocessor relevant configuration parameters

1. In the Designer, select the **Base data > General > Configuration parameters** category.
2. In the Configuration Parameter Editor, select the **View > Preprocessor definitions** menu item.

The **Preprocessor definitions** view shows all preprocessor conditions. Double-click an entry to display the configuration parameter.

NOTE: In the Designer, you can find an overview of existing preprocessor dependencies in the **One Identity Manager Schema > Preprocessor dependencies** category.

Related topics

- [Creating custom configuration parameters](#) on page 41
- [Preprocessor conditions in objects](#) on page 337
- [Preprocessor conditions in VB.Net expressions](#) on page 338
- [Evaluation of preprocessor conditions during compilation](#) on page 339

Preprocessor conditions in objects

IMPORTANT: Each modification to preprocessor objects requires recompiling the One Identity Manager database.

You can enter a preprocessor condition directly for certain objects.

To enter a preprocessor condition

- In the **Preprocessor condition** property, enter the preprocessor expressions of the configuration parameters. You can link preprocessor expressions together with AND, OR, NOT, ().

Example:

The column `Person.RiskIndexCalculated` should only be shown in the interface if the risk function is set.

The following preprocessor conditions are entered in the column definition (`DialogColumn` table).

Table 131: Example for preprocessor conditions

Table	Column	Preprocessor condition
Employee	RiskIndexCalculated	COMPLIANCE

If a preprocessor-relevant configuration parameter is enabled or disabled, tasks are created for the DBQueue Processor to calculate all preprocessor and calculation tasks for the affected objects. The **Disabled by preprocessor** option is updated for each object. If the re-interpretation of the preprocessor conditions leads to a change in the option, the preprocessor interpretation tasks that follow are generated for the dependent objects. User permissions can also be affected. After DBQueue Processor has processed the tasks, the database needs to be recompiled.

The interpretation of preprocessor conditions has the following effects:

- If a table is disabled by a preprocessor condition then all the columns and object definitions that relate to the table and the user interface forms and the associated navigation are disabled.
- If a primary key column is disabled, all foreign key columns that relate to it are disabled.
- If a primary key member is disabled according to the preceding rule (for example, in the case of many-to-many tables), then this primary key's table and all further columns belonging to this table are also disabled.

This method has the advantage that, for example, when a table such as ADSSGroup is disabled then all assignments are automatically disabled, such as the table, DepartmentHasADSSGroup.

NOTE: In the Designer, you can find an overview of existing preprocessor dependencies in the **One Identity Manager Schema > Preprocessor dependencies** category.

Related topics

- [Preprocessor-relevant configuration parameters](#) on page 336
- [Preprocessor conditions in VB.Net expressions](#) on page 338
- [Evaluation of preprocessor conditions during compilation](#) on page 339

Preprocessor conditions in VB.Net expressions

IMPORTANT: Every modification to preprocessor conditions in VB.Net expressions requires recompiling the One Identity Manager database.

Preprocessor conditions can be used in VB.Net expressions. Script code that is dependent on a preprocessor condition has to be passed in an `#if...then...#else` statement. To formulate the preprocessor condition, use the preprocessor expressions of the preprocessor-related configuration parameters. You can link preprocessor expressions together with AND, OR, NOT, ().

The interpretation of the preprocessor conditions is not carried out until the script is generated.

Syntax

```
#If <preprocessor_condition_1>
    ' code, for this preprocessor condition
#ElseIf <preprocessor_condition_2> then
    ' code, for this preprocessor condition
#Else
    ' other code
#Endif
```

Example:

The ITSHOP preprocessor condition is entered in the column definition (DialogColumn table) for the ADSGroup.IsForITShop column. The template in the ADSGroup.DisplayName column should reference the IsForITShop column. In order to remain compatible, the following construction has to be used for the template:

```
#If ITSHOP Then
    If $IsForITShop:Bool$ And $UID_AccProduct$ <> "" Then
        Value = $FK(UID_AccProduct).Ident_AccProduct$
    Else
        value = $cn$
    End If
#Else
    value = $cn$
#End If
```

Related topics

- [Preprocessor-relevant configuration parameters](#) on page 336
- [Preprocessor conditions in objects](#) on page 337
- [Evaluation of preprocessor conditions during compilation](#) on page 339

Evaluation of preprocessor conditions during compilation

In order to become effective on a systemwide basis, every modification to preprocessor relevant configuration parameters as well as modifications to preprocessor conditions on objects and VB.Net expressions requires the One Identity Manager database to be recompiled.

The following is true for compiling:

- Internal program code in the form of an #if...then...#else statement is created for objects that have a preprocessor condition. Program code in sections whose preprocessor condition does not apply, do not exist for the compiler and are therefore not parsed. These objects are assumed not to exist.
- VB.Net expressions that contain preprocessor conditions are compiled. The program code exists. The interpretation of the preprocessor conditions is not carried out until the script is generated.

These templates are valid for compiling:

- Templates for columns that are disabled by preprocessor conditions are not compiled and the resulting relations are not saved in the `DialogNotification` table. These columns are therefore considered to be non-existent.
- Templates that relate to disabled columns cause a compiler error message if the corresponding part of code is not linked in a preprocessor statement.

Scripts in One Identity Manager

Scripts are used in One Identity Manager to monitor and maintain data consistency and customer business logic in the database. Scripts can be used to:

- Test column values
- Trigger events
- Create, change, and delete objects and therefore manipulate the database.

Detailed information about this topic

- [Visual Basic .NET scripts usage](#) on page 341
- [Notes on message output](#) on page 342
- [Notes on using date values](#) on page 342
- [Tips for using Windows PowerShell scripts](#) on page 343
- [Using dollar \(\\$\) notation](#) on page 344
- [Using base objects](#) on page 350
- [Calling functions](#) on page 350
- [Pre-scripts for use in processes and process steps](#) on page 351
- [Using session services](#) on page 351
- [Using #LD-notation](#) on page 354
- [Script library](#) on page 358

Visual Basic .NET scripts usage

One Identity Manager scripts are written in VB.Net syntax, which allows all VB.Net functions to be used. The values to be edited are given as preprocessor instructions.

NOTE: You can find detailed examples for syntax and usage of scripts on the installation medium in the QBM\dvd\AddOn\SDK\ScriptSamples directory.

You can use scripts in:

- Templates and formatting scripts (DialogColumn table)
- Table scripts (DialogTable table)
- Script library (DialogScript table)
- Tasks (DialogMethod table)
- Object definition selection scripts (DialogObject table)
- Views selection scripts (DialogTable table)
- Scripts to find the servers to carry out the process steps (Job table)
- Process step parameters (Jobrunparameter table)
- Process control notification (Job table)
- Generating conditions for process steps and processes (Job and JobChain tables)
- Process step and process pre-scripts (Job and JobChain tables)
- Process information (Job, JobChain and JobEventgen table)
- Mail templates (DialogRichMailBody table)

Notes on message output

You should never use the VB.Net MsgBox and Inputbox functions on servers. Use the VID_Write2Log, RaiseMessage, or AppData.Instance.RaiseMessage functions.

For examples of One Identity Manager Service log file output, see the script examples on the installation medium in the QBM\dvd\AddOn\SDK\ScriptSamples directory.

Notes on using date values

- If no date is given, the date 12/30/1899 is used internally. Take this into account when values are compared, for example, when used in reports.

Example: Expression for displaying data columns in reports

```
{IIF(Person.ExitDate.ToString() = "12/30/1899 12:00:00 AM", "-",
    Person.ExitDate)}
```

- Time stamps, such as insert dates or modification dates, are stored in the database with the respective UTC. The object layer transforms this time data into the currently valid time zone data when an object is loaded. The user, therefore, sees all the values

in local time. When an object is saved the current time zone data is transformed into UTC data.

NOTE: The use of `DateTime.Now` in scripts must be critically tested. It is better to use `DateTime.UtcNow` than `DateTime.Now` to display the value to users.

- It is not recommended to convert date values in non-U.S. notation from the `String` data type to the `DateTime` data type in scripts:

```
Value = CDate("2014-12-31")
```

This always causes a problem if the script is running on a U.S. system. In the best case, you are sent an error message like "Cast from string...to type Date is not valid". In the worst case, the wrong date is returned as month and day are swapped (3.12.2014 becomes 12.3.2014).

If possible, you should avoid a string conversion altogether in this case. The `DateTime` type provides several constructors for this purpose. For the example above, that would be:

```
Value = new DateTime(2014, 12, 31)
```

However, if the data type `String` is to be used, the ISO date notation should be applied as this is converted correctly in all settings:

```
Value = CDate("2014-12-31")
```

```
Value = CDate("2014-12-31 15:22:12")
```

The complicated version is to input the language code format for the date:

```
Value = DateTime.Parse("12.31.2014", new CultureInfo("en-US"))
```

```
Value = DateTime.ParseExact("12.31.2014", "mm.dd.yyyy",  
CultureInfo.InvariantCulture)
```

Tips for using Windows PowerShell scripts

For examples of syntax and usage for Windows PowerShell scripts in One Identity Manager, see <https://github.com/OneIdentity/IdentityManager.PoSh>. Note the installation prerequisites and guidance given there.

By default, the Windows PowerShell module tries to load all referenced DLLs from a valid One Identity Manager installation. In the default installation, One Identity Manager is installed under:

- `%ProgramFiles(x86)%\One Identity` (on 32-bit operating systems)
- `%ProgramFiles%\One Identity` (on 64-bit operating systems)

Using dollar (\$) notation

Dollar (\$) notation is used to access object properties in One Identity Manager.

Syntax

`$<definition>:<data type>{<format>}$`

If you are using dollar notation you need to ensure that the value is allocated the correct data type. Dollar notation returns a `String` type by default.

Permitted data types are:

- Binary
- Bool
- Byte
- Date
- Decimal
- Double
- Int
- Long
- Short
- String (default)
- Text

The format specification is optional. If the format is specified, the target type of the expression is a `string`. If the format is not specified, it is the specified data type.

The format specifications correspond to the format strings of the `string.format` function for the individual types. For more information about the format string, see <https://docs.microsoft.com/en-us/dotnet/standard/base-types/composite-formatting#format-string-component>.

Examples:

`$MaxValidDays:Int{000}$`

with the value **42** give the result **"042"**

`$XDateUpdated:Date{t}$`

gives **"11:16"**

NOTE: If you want to use a dollar \$ sign in scripts, without it representing access to a column name, you must mask it by doubling.

Example:

In Windows PowerShell scripts, instead of:

```
theScript.AppendLine("foreach ($Domain in $Domains)")
```

use:

```
theScript.AppendLine("$foreach ($$Domain in $$Domains)")
```

Detailed information about this topic

- [Accessing local object columns](#) on page 345
- [Accessing columns of an object connected by a relation](#) on page 346
- [Accessing the old column value](#) on page 346
- [Accessing the display value of a column](#) on page 347
- [Accessing references in comments](#) on page 348
- [Accessing metavalues of the local object](#) on page 349
- [Accessing objects' display values](#) on page 349

Accessing local object columns

Syntax

```
$<column name>:<data type>{<format>}$
```

Examples for use in templates

The Active Directory user display name should comprise of the first and last name of the Active Directory user. The template for ADSAccount.Displayname is:

```
If $Givenname$<>" " And $Surname$<>" " Then
```

```
    Value = $Surname$ & " " & $Givenname$
```

```
ElseIf $Givenname$<>" " Then
```

```
    Value = $Givenname$
```

```
ElseIf $Surname$<>" " then
```

```
    Value = $Surname$
```

```
End If
```

If an employee is disabled, the leaving date should be set. The template for Person.Exitdate is:

```
If $IsInactive:bool$ Then
```

```
    Value = Date.Today
```

```
End If
```

Related topics

- [Accessing columns of an object connected by a relation](#) on page 346
- [Accessing the old column value](#) on page 346
- [Accessing the display value of a column](#) on page 347
- [Accessing references in comments](#) on page 348
- [Accessing metavalues of the local object](#) on page 349
- [Accessing objects' display values](#) on page 349

Accessing columns of an object connected by a relation

The only relation currently permitted is the foreign key relation.

Syntax

```
$FK(<foreign key column>).<column name>:<data type>{<format>}$
```

Example for use in templates:

An Active Directory user's first name should be based on the assigned employee. The template for ADSSAccount.Givenname is:

```
Value = $FK(UID_Person).Firstname$
```

Related topics

- [Accessing local object columns](#) on page 345
- [Accessing the old column value](#) on page 346
- [Accessing the display value of a column](#) on page 347
- [Accessing references in comments](#) on page 348
- [Accessing metavalues of the local object](#) on page 349
- [Accessing objects' display values](#) on page 349

Accessing the old column value

Syntax

```
$columnname[o]$
```

Example for use in process step parameters:

Optional process step parameters are not generated if the value is set to **Nothing** or not assigned in the value template. This makes it possible to limit the number of parameters for target system components. If such a value should be cleared, an empty string should be transferred instead of **Nothing**.

A value template may look like this:

```
If $Lastname[o]$ <> $Lastname$ Then
    Value = $Lastname$
End If
```

NOTE: For some standard columns such as XDateInserted, XDateUpdated, XUserInserted, XUserUpdated, XOrigin, XIsInEffect, and XMarkedForDeletion, the new values are only mapped after saving the object. This means that when processing the templates, the new column value is always the same as the old value (for example, \$XDateUpdated[o]\$ = \$XDateUpdated\$).

Related topics

- [Accessing local object columns on page 345](#)
- [Accessing columns of an object connected by a relation on page 346](#)
- [Accessing the display value of a column on page 347](#)
- [Accessing references in comments on page 348](#)
- [Accessing metavalues of the local object on page 349](#)
- [Accessing objects' display values on page 349](#)

Accessing the display value of a column

When a display value for a column is created, the **Multilingual** (IsMultiLanguage) and **List of permitted values** (LimitedValues) properties are resolved.

Syntax

\$columnname[D]\$

To access the display value of a column's old value, combine the [O] and [D] properties.

\$column name[OD]\$

\$column name[DO]\$

Example of use:

A list of permitted values is defined for the restriction type of the HCL Domino server restrictions.

```
PrivateList=Run Personal Agent RestrictedList=Run Restricted Agent  
UnrestrictedList=Run Unrestricted Agent
```

If a server restriction has the **PrivateList** value, the **Run Personal Agent** value is displayed on the information form.

Example for use in templates:

The display value for the server restriction should be formatted from the name of the HCL Domino user and the display value of the restriction type.

```
Value = vid_Left($FK(UID_NotesUser).FullName1st$,39) & " [" & vid_Left  
($NotesAgentMgrType[D]$, 22) & "]"
```

Related topics

- [Accessing local object columns on page 345](#)
- [Accessing columns of an object connected by a relation on page 346](#)
- [Accessing the old column value on page 346](#)
- [Accessing references in comments on page 348](#)
- [Accessing metavalues of the local object on page 349](#)
- [Accessing objects' display values on page 349](#)

Accessing references in comments

The preprocessor also interprets references that are embedded in comments, for example, `$Lastname$`. Referencing a column in a script comment results in the script being run when the column value is changed.

Example for use in templates:

An employee's starting date is filled with a template. This template should run when the employee's surname changes. The template for `Person.Entrydate` is therefore:

```
'$Lastname$  
Value = Date
```

Related topics

- [Accessing local object columns on page 345](#)
- [Accessing columns of an object connected by a relation on page 346](#)
- [Accessing the old column value on page 346](#)
- [Accessing the display value of a column on page 347](#)

- [Accessing metavalues of the local object](#) on page 349
- [Accessing objects' display values](#) on page 349

Accessing metavalues of the local object

Syntax

`$(IsLoaded):Bool$`

Table 132: Metavalues and their meaning

Metavalue	Meaning
IsLoaded	This value specifies whether the object is loaded from the database.
IsChanged	This value specifies whether the object is altered when it is loaded from the database.
IsDifferent	This value specifies whether the new value is different from the old value. You can access to the column through: <code>Columnname[C]</code> .
IsDeleted	This value specifies whether the object is marked for deletion.

Related topics

- [Accessing local object columns](#) on page 345
- [Accessing columns of an object connected by a relation](#) on page 346
- [Accessing the old column value](#) on page 346
- [Accessing the display value of a column](#) on page 347
- [Accessing references in comments](#) on page 348
- [Accessing objects' display values](#) on page 349

Accessing objects' display values

Syntax

Display pattern corresponding to an object's `DisplayPattern` column:

`$(Display)$`

Display pattern corresponding to an object's `DisplayPatternLong` column:

`$(LongDisplay)$`

Example of usage on the base object

`$(Display)$`

Example of usage over foreign key relation

For assignments of Active Directory groups to departments, use the ADGroup's DisplayPatternLong.

```
$FK(UID_ADGroup).[LongDisplay]$
```

Related topics

- [Accessing local object columns](#) on page 345
- [Accessing columns of an object connected by a relation](#) on page 346
- [Accessing the old column value](#) on page 346
- [Accessing the display value of a column](#) on page 347
- [Accessing references in comments](#) on page 348
- [Accessing metavalues of the local object](#) on page 349

Using base objects

The Base. syntax always accesses the object that is currently loaded. The base. object can be used in tasks, selection scripts for object definitions and insert values. However, the base. object cannot be used in templates, formatting scripts, or processes.

Syntax

- Simple value assignment
`Base.PutValue("<column>", <value>)`
- Value assignment with variable replacement (value must be a character string)
`Base.PutValue("<column>", context.Replace(<value>))`

Example

```
Base.PutValue("IsForITShop", 1)  
Base.PutValue("UID_ADSContainer", context.Replace("%cont%"))
```

Calling functions

Functions are stored in the script library (DialogScript table).

Example of a function in the script library

```
Public Function BuildInternalName(ByVal Firstname As String, ByVal Lastname As String) As String
    BuildInternalName = Lastname & Firstname
End Function
```

Using the function in a template on `person.internalname`

```
Value = BuildInternalName($Firstname$, $Lastname$)
```

Pre-scripts for use in processes and process steps

Pre-script code is code that is run before other scripts are run. You can define process specific variables. Process specific variables are local data spaces when a process is generated. They are used for determining values on a one-off basis within a pre-script, which can then be made further use of within the processes and their processes steps, for example, in generating conditions or server selection scripts, or in the parameters.

NOTE: It is recommended only to set process specific variables in the pre-script and to have read access to them during further usage.

Syntax in the pre-script of a process

```
values("Name") = "value"
```

Usage in the process and process step code sections

```
Value = values("Name")
```

Related topics

- [Using process-specific and global variables for the process definition](#) on page 245
- [Querying session object global variables](#) on page 353

Using session services

The session object is the instance that makes data available to a user session. This includes the current user, their permissions groups and program functions. Furthermore, the session object makes various services available for accessing data. The services provided

by the session object are made available through a generic interface (Resolve (Of Service)()). In the following sections, examples are provided of frequently used service.

NOTE: You can find a complete description of all parameters in the VI.DB.DLL documentation.

Detailed information about this topic

- [Querying configuration parameters](#) on page 352
- [Testing for the existence of certain database entries](#) on page 353
- [Querying session object global variables](#) on page 353

Querying configuration parameters

The full path for the configuration parameter always has to be entered when configuration parameter are queried.

Syntax

```
Session.Config().GetConfigParm("<full path>")
```

When a configuration parameter is tested in a generating condition in VB.Net syntax, the function returns a string. In order to compare this value to a numerical value, the configuration parameter has to be set and contain a numerical value. This depends on the implicit value type conversion from VB.Net. If the configuration parameter is not enabled, the function returns an empty string ("") that cannot be compared to a numerical value. This results in a VB.Net runtime error. Configuration parameter values are therefore always compared to strings.

Do not use:

```
Session.Config().GetConfigParm("QER\Person\User\DeleteOptions\Homedir")=1
```

Use instead:

```
Session.Config().GetConfigParm("QER\Person\User\DeleteOptions\Homedir")="1"
```

In order to ensure that a logical value is always returned, the VID_IsTrue function should be used.

Example

```
If VID_IsTrue(Session.Config().GetConfigParm  
("QER\Person\User\DeleteOptions\Homedir")) Then ...
```

Related topics

- [Testing for the existence of certain database entries](#) on page 353
- [Querying session object global variables](#) on page 353

Testing for the existence of certain database entries

NOTE: The test should take place without taking access permissions into account.

Syntax

```
Session.Source().Exists("<Tablename>", "<WhereClause>")
```

Example

```
Session.Source().Exists("Person", "CentralAccount = '' & acct & '' and uid_person  
<> '' & uid_person & ''")
```

Related topics

- [Querying configuration parameters](#) on page 352
- [Querying session object global variables](#) on page 353

Querying session object global variables

Global variables are allocated by the set up program. In addition to the predefined variables, all environment variable and custom variables defined on the session object can be used. Custom session variables can be defined, for example, using scripts, methods, or customizers.

NOTE: If you define a custom session variable, you must remove it again afterward. Otherwise it remains for the rest of the session and, in certain circumstances, the wrong processes can be generated.

Syntax

```
Variables("<Variable name>")
```

Example of use in process step parameters

```
Value = Variables("GENPROCID")  
Value = CBool(Session.Variables("FULLSYNC"))
```

Table 133: Permitted predefined global variables

Variable	Meaning
EnvUserName	Name of user to be authenticated in the environment,

Variable	Meaning
	for example, Domain\User in Active Directory
FullSync	The variable is set by all synchronizers. The values are True and False .
GenProcID	Unique Process ID number
LogonUser	DialogUser.Username of the currently logged in user.
DialogUserUID	DialogUser.UID_DialogUser of the logged in user.
UserName	Name displayed in XUserInserted or XUserUpdated.
UserUID	Logged in user's UID_Person, if user related authentication is being used.
ShowCommonData	Specifies whether system data is shown (value = 1) or not shown (value = 0). The variable is evaluated in the Designer by the Show system data program setting.
SessionType	Specifies whether a direct database connection or a connection over an application server is supported. Direct database connection only: '%SessionType%' = 'Direct' Connect with the application server only: '%SessionType%' = 'AppServer'
Feature_<Featurename>	Queries additional program functions (DialogFeature) that are available for the user. The value is 1 when the program function is available, otherwise the variable is not set.
ManageOutstandingOperation	This variable is used to differentiate between running operations during post-processing of outstanding objects in target system synchronization. Permitted values are Delete , DeleteState , and Publish .

Related topics

- [Querying configuration parameters](#) on page 352
- [Testing for the existence of certain database entries](#) on page 353

Using #LD-notation

#LD notation is used for displaying language-dependent information. #LD notation is mainly used in process tracking and processing notification, but it can also be used in scripts that are stored in the script library.

Syntax

Value=#LD[<language>|<language code>](<key>,{<Parameter>}*)#

where:

<language> <language code>	(Optional) Language or language variant for the output
<Key>	Basis string with place holder. The place holder syntax corresponds to a format place holder in VB.Net ({0} to {9})
<Parameter>	Parameter for replacing the place holder (comma delimited)

Table 134: Using #LD-notation

Context	Table.column	Remarks
Process tracking	Job.ProcessDisplay	Mapped to DialogProcessStep.DisplayName
	JobChain.ProcessDisplay	Mapped to DialogProcessChain.DisplayName
	JobEventgen.ProcessDisplay	Mapped to DialogProcess.DisplayName
Process handling notification	Job.NotifyAddress	
	Job.NotifyAddressSuccess	
	Job.NotifyBody	
	Job.NotifyBodySuccess	
	Job.NotifySender	
	Job.NotifySenderSuccess	
	Job.NotifySubject	
	Job.NotifySubjectSuccess	
	JobRunParameter.ValueTemplate	On in the MailComponent process component
Templates	DialogColumn.Template	
	DialogColumn.CustomTemplate	
Formats	DialogColumn.FormatScript	
	DialogColumn.CustomFormatScript	
Task definitions	DialogMethod.MethodScript	

Context	Table.column	Remarks
Insert values	DialogObject.InsertValues	
	DialogTable.InsertValues	
	DialogTree.ListInsertValues	
	DialogSheet.InsertValues	
Selection scripts	DialogTable.SelectScript	
	DialogObject.SelectScript	
Process generating scripts	Job.GenCondition	
	Job.PreCode	
	Job.ServerDetectScript	
	JobChain.GenCondition	
	JobChain.PreCode	

Related topics

- [Using #LD notation in process tracking](#) on page 356
- [Example of specifying the language or language variant](#) on page 357

Using #LD notation in process tracking

For language-dependent representation of process information, a relevant template must be defined to display the captions in the active languages.

The captions for language-dependent text are entered in `DialogMultiLanguage` when the script is compiled. A key (column `Entrykey`), the language and the translation (column `EntryValue`) are entered into the table. The key should be in the corresponding default language. If a language caption has not been entered, the key is used as the display text. Use the Language Editor to add translations for the captions in other languages.

Example:

A change is made to an employee. The language-dependent process information could be formulated as follows:

- Value template for the process information on the Update event
Value = #LD("Change of properties of person {0}.", \$InternalName\$)#
- Templates for the display texts in the DialogMultiLanguage table

Key	Language	Value
Changed properties of employee {0}.	English - United States [en-US]	Changed properties of employee {0}.
Changed properties of employee {0}.	German - Germany [de-DE]	Änderung der Daten der Person {0}.

With InternalName = JBasset, the following display texts are produced in the process view.

Current user's language	Display text in the process view
English - United States [en-US]	Change of properties of person JBasset.
German - Germany [de-DE]	Änderung der Daten der Person JBasset.

Related topics

- [Displaying translations in the Language Editor](#) on page 214

Example of specifying the language or language variant

#LD notation supports the specification of a language or language variant. This is particularly useful in cases where users need to receive system messages in their preferred language.

Examples

- Output in the default language:
Value = #LD("Test: {0}", <parameter>)#
Value = #LD[""]("Test: {0}", <parameter>)#
- Output always in English
Value = #LD["en-US"]("Test: {0}", <parameter>)#
Value = #LD["english"]("Test: {0}", <parameter>)#
- Using a variable:

```
Dim lang As String = "en-US"

Value = #LD[lang]("Test: {0}", <parameter>)#
```

You do not need to enter the language in square brackets, it is optional. However, it is important that the language statement is a String expression. If the language is not specified or the resulting String expression is empty or **Nothing**, the language currently set for the application is used for translation.

Script library

The script library contains source code for the scripts used in One Identity Manager. The default scripts that we supply cannot be edited. These scripts are overwritten during schema installation even if they are used in custom scripts.

NOTE: You can find detailed examples for syntax and usage of scripts on the installation medium in the QBM\dvd\AddOn\SDK\ScriptSamples directory. You can find examples of unit tests under QBM\dvd\AddOn\SDK\UnitTestSamples.

Scripts are displayed under **Script Library** in the Designer. You can gather all the information about usage, for example, in column definitions, processes, or other scripts, in the script overview.

Use the Script Editor to create, edit, and test scripts. To use Visual Studio's more extensive debug and edit options, edit, and test the scripts in the System Debugger.

NOTE: Scripts for synchronization projects from the Synchronization Editor's script library are not available in the Designer. For more information about the Synchronization Editor script library, see *One Identity Manager Target System Synchronization Reference Guide*.

Detailed information about this topic

- [Support for processing scripts in the Script Editor](#) on page 359
- [Creating and editing scripts in the Script Editor](#) on page 365
- [Copying scripts in the Script Editor](#) on page 366
- [Testing script compilation in the Script Editor](#) on page 368
- [Testing scripts in the Script Editor](#) on page 367
- [Overriding scripts](#) on page 368
- [Permissions for running scripts](#) on page 369
- [Editing and testing script code with the System Debugger](#) on page 370
- [Extended debugging in the Object Browser](#) on page 376

Support for processing scripts in the Script Editor

A special input field is used for editing scripts. It has an advanced edit mode that provides additional actions.

To switch to advanced mode

- Press **Ctrl + Alt + Enter** or click the button at the bottom right.

Figure 31: Directly entering a database query

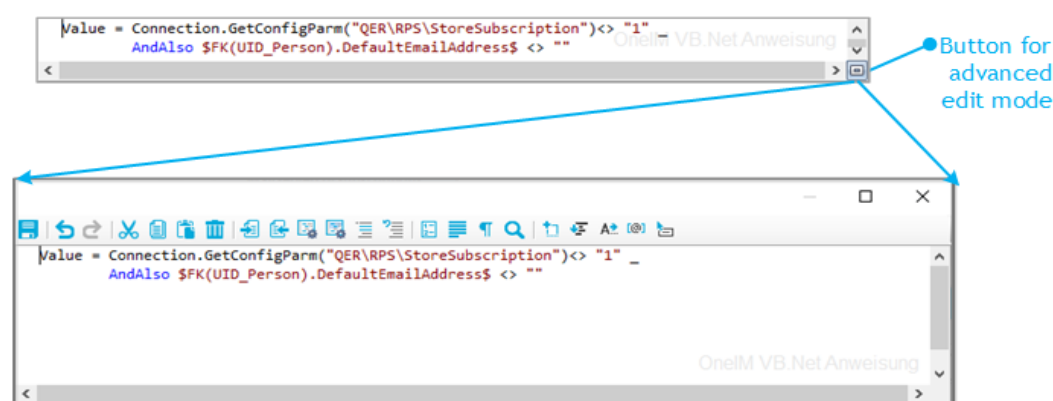














Table 135: Meaning of icon in advanced edit mode

Icon	Meaning
	Quits advanced edit mode.
	Undoes last change.
	Redoes last change.
	Cuts selected code.
	Copies selected code into clipboard.
	Inserts code from clipboard.
	Deletes selected code.
	Decreases insert.
	Increases insert.
	Automatic text formatting.
	Formats text selection automatically.
	Shows/hides line numbers.







Icon	Meaning
	Inserts comments.
	Removes comments.
	Inserts or removes line numbers.
	Inserts or removes automatic line breaks.
	Shows or hides spaces and tabs.
	Searches within code.
	Inserts code snippet.
	Shows list of objects properties.
	Shows auto completion list.
	Shows list with parameter information.
	Shows additional information.

Table 136: Shortcut for editing scripts

Shortcut	Description
Ctrl + C	Copy to clipboard.
Ctrl + Ins	Copy to clipboard.
Ctrl + X	Cut and copy to clipboard.
Shift + Del	Cut and copy to clipboard.
Ctrl + L	Cut row and copy to clipboard.
Ctrl + V	Paste from clipboard.
Shift + Ins	Paste from clipboard.
Ctrl + Y	Redo action.
Ctrl + Shift + Z	Redo action.
Ctrl + Z	Undo action.
Backspace	Remove character behind cursor.
Shift + Backspace	Remove character behind cursor.
Ctrl + Backspace	Remove word behind cursor.
Del	Delete character in front of cursor.
Ctrl + Shift + L	Delete row.
Ctrl + Del	Delete to end of next word.

Shortcut	Description
Data	Insert line break.
Shift + Return.	Insert soft line break.
Ctrl + Return	Insert row above.
Ctrl + Shift + Return	Insert row below.
Ctrl + Space	Auto-complete.
Ctrl + Space + Space	Show list with parameter information.
Tab	Insert indent/tab.
Shift + Tab	Remove indent/tab.
Ctrl + U	Change marked characters to lowercase.
Ctrl + Shift + U	Change marked characters to uppercase.
Ins	Toggle insert mode.
Ctrl + T	Swap characters in front and behind cursor.
Ctrl + Shift + T	Swap words in front and behind cursor.
Shift + Alt + T	Swap row with previous row.
Alt + PgUp	Move row up.
Alt + PgDn	Move row down.
PgUp	Move up
PgDn	Move down.
Left arrow	Move left.
Right arrow	Move right.
Ctrl + Left arrow	Move to previous word.
Ctrl + Right arrow	Move to next word.
Home	Move to start of line.
End	Move to end of line.
Ctrl + Home	Move to start of script.
Ctrl + End	Move to end of script.
PgDn	Move up a page.
PgUp	Move down a page.
Ctrl + PgUp	Move to visible start.

Shortcut	Description
Ctrl + PgDn	Move to visible end.
Ctrl +]	Move to next bracket. (not possible on German keyboard)
Ctrl + Down arrow	Scroll down.
Ctrl + Up arrow	Scroll up
Ctrl + F	Open search dialog.
F3	Search next.
Ctrl + F3	Search forward.
Shift + F3	Search backward.
Ctrl + Shift + F3	Find previous.
Ctrl + H	Replace.
Ctrl + I	Search forward incrementally.
Ctrl + Shift + I	Search backward incrementally.
Ctrl + Shift + Num-	Reduce code block selection.
Ctrl + Shift + Num+	Extend code block selection.
Esc	Remove selection.
Shift + Down arrow	Extend selection down.
Shift + Up arrow	Extend selection up.
Shift + Left arrow	Extend selection left.
Shift + Right arrow	Extend selection right.
Ctrl + Shift + Left arrow	Extend selection to previous word.
Ctrl + Shift + Right arrow	Extend selection to next word.
Shift + Home	Extend selection to start of line.
Shift + End	Extend selection to end of line.
Shift + Alt + Home	Extend selection to start of script.
Ctrl + Shift + End	Extend selection to end of script.
Shift + PgUp	Extend selection by one page up.
Shift + PgDn	Extend selection by one page down.
Ctrl + Shift + PgUp	Extend selection to visible start.
Ctrl + Shift + PgDn	Extend selection to visible end.

Shortcut	Description
Ctrl + A	Select all.
Ctrl + Shift + W	Select word.
Ctrl + Shift +]	Select up to the next bracket. (not possible on German keyboard)
Shift + Alt + Down arrow	Extend selected block down.
Shift + Alt + Up arrow	Extend selected block up.
Shift + Alt + Left arrow	Extend selected block left.
Shift + Alt + Right arrow	Extend selected block right.
Ctrl + Shift + Alt + Left arrow	Extend selected block by one word to the left.
Ctrl + Shift + Alt + Right arrow	Extend selected block by one word to the right.
Ctrl + mouse wheel	Adjust text size.

There is additional help provided for creating script code.

Syntax highlighting

The input fields support syntax highlighting depending on the syntax type.



Auto-completion

You can use auto-completion when you write script code. The amount of scripted code to enter is reduced by displaying the names of properties or functions that can be used. To use auto-completion, use the shortcut **Ctrl + SPACE** in the relevant positions within the input fields. The contents of the list is determined by the key words in the code.

Entering code snippets


Input fields that require data in VB.Net syntax support code snippets. In the **Visual Basic** category, general code snippets are provided. The **Object Layer** category contains special code snippets for the One Identity Manager object layer. In the **SQL Formatter** category, code snippets database queries are formatted from VB.Net.

You can insert code snippets using the following options:

1. Using the  icon
 - a. Select the  icon.
 - b. Select the **Object Layer**, **SQL Formatter**, or **Visual Basic** category.
 - c. Select the code snippet.

2. Using a shortcut
 - a. Press **F2**.
 - b. Select the **Object Layer**, **SQL Formatter**, or **Visual Basic** category.
 - c. Select the code snippet.
3. Using an aliases
 - a. Enter an alias.
 - b. Use **Tab** to insert the code snippet.

NOTE: Be aware of case sensitivity when you enter the alias.

NOTE: If you select a code snippet directly using a shortcut or the  icon, a short description and the shortcut name are displayed in a tooltip.

Custom code snippets

You can use custom code snippets. To do this, create a CustomSnippets directory in the One Identity Manager installation directory to store the code snippets. Use Visual Studio documentation to develop your own code snippets.

To sort custom code snippets, use the following syntax to enter a sort order in the code snippet header in the code snippet file. When the code snippet is entered, the numbering is hidden.

```
<Title>(1) Your title for the code snippet</Title>
<Title>(2) Your title for the code snippet</Title>
```

Inputting values using dollar (\$) notation

In input fields where a VB.NET term is expected, a help list opens when you enter **\$**. All properties of the current object are displayed. You can also see a tooltip with a detailed description of the property. When you select a foreign key (FK) column, you can navigate to the columns in the relevant table using the arrow keys. To end the selection in the target column, press **Enter** or double-click. The complete \$ notation for your selection should now be shown in the input column. To close the help list without copying any data, press **Esc** or leave the input field.

Figure 32: Help list for dollar notation

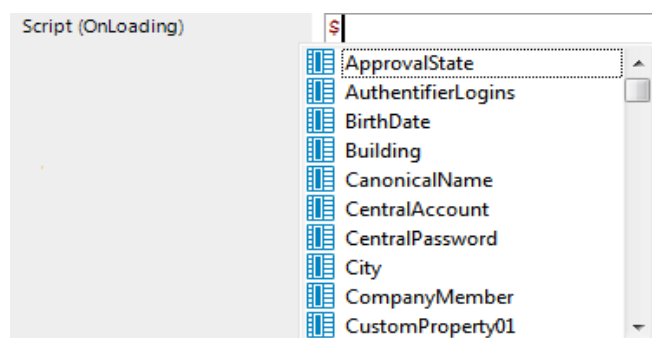


Table 137: Meaning of icons used in the help list for dollar notation.








Icon	Meaning
	Property of current object.
	Primary key (PK).
	Foreign key (FK).
	Dynamic foreign key
	Table
	Special properties
	Script

Table 138: Help list functions for dollar notation

Shortcut	Action
Arrow down key	Opens the help list.
Arrow up key, arrow down key	Navigate to previous or next entry.
Arrow left key, arrow right key	Navigate to the object above or below over the foreign key.
Data	Accepts the value in dollar notation.

Creating and editing scripts in the Script Editor

IMPORTANT: After creating and editing the script, you should test compiling the script. Compile the scripts in the script library for this script to take effect.

NOTE: You can find detailed examples for syntax and usage of scripts on the installation medium in the QBM\dvd\AddOn\SDK\ScriptSamples directory.

To create a new script

1. In the Designer, select the **Script Library** category.
2. Start the Script Editor using the **Create a new script** task.
3. Edit the main data.

Table 139: Script main data

Property	Description
Script	Name of the script. Prefix custom scripts with CCC_ .
Description	Detailed description about the script's function
Script code	One Identity Manager scripts are written in VB.Net syntax, which allows all VB.Net functions to be used. The values to be edited are given as preprocessor instructions.
Locked	Specifies whether the script is locked and therefore may not be used. Locking the script is useful, for example, if it is currently being edited.

To edit a script

1. In the Designer, select the script in the **Script Library**.
2. Select the **Edit script** task.
3. Edit the script main data.

Detailed information about this topic

- [Copying scripts in the Script Editor](#) on page 366
- [Overriding scripts](#) on page 368
- [Testing scripts in the Script Editor](#) on page 367
- [Visual Basic .NET scripts usage](#) on page 341

Copying scripts in the Script Editor

IMPORTANT: After creating and editing the script, you should test compiling the script. Compile the scripts in the script library for this script to take effect.

To copy an existing script

1. In the Designer, select the script that you want to copy from the **Script Library** category.
2. Select the **Copy script** task.
3. In the **Copy script** dialog, check the following information and correct if necessary.

Table 140: Copying a script

Property	Description
Old script name	Name of the copied script.
script	The name of the new script is made up of the CCC_ prefix and the name of the old script. You can change the name. Prefix custom scripts with CCC_ .
Script code	The script code from the original is copied over. If necessary, you can modify the script code of the script to copy beforehand.

4. To create the copy, click **OK**.
5. In the Script Editor, edit the main data of the script.

Related topics

- [Creating and editing scripts in the Script Editor](#) on page 365

Testing scripts in the Script Editor

You can use the Script Editor to test a script.

To test a script

1. In the Designer, select the script in the **Script Library**.
2. Select the **Edit script** task.
3. Select the **View > Test script** menu item.
4. In the **Test script** view, select the script from the menu and modify the parameters as required.

All the parameters to be passed to the script are displayed with their data types. You can edit the values. You can also predefine values for the Base and Value script base class variables as input parameters and use these in the script.

5. Use **Options** to select one or more of the following options for running the test.
 - **Use main database connection:** Specifies whether the script test is tested against the main database or an internal SQLite database. Scripts that relate to the application part of the One Identity Manager data model should always be tested with the main database. Scripts for system parts can be test with against the main database or the internal SQLite database.
 - **Use transaction:** Specifies whether the script is run within a transaction with subsequent rollback or whether the script is run immediately against the database.

- **Record SQL log:** Specifies whether the database actions should be recorded in an SQL log while the script is running. The output is displayed in a separate dialog window. The time to run the script is output in addition to the statement run.
6. Select the **Start** button to run the script test.
- The test results are displayed in the **Result** field after the script has been run.


Related topics

- [Editing and testing script code with the System Debugger](#) on page 370

Testing script compilation in the Script Editor

If you have created a new script, you need to compile it. The script cannot be generated until it has been compiled. You can test script compilation in the Script Editor.

To test compiling scripts

- In the Designer, select the script in the **Script Library** category.
- Select the **Edit script** task.
- Start compilation with the icon , the **Script > Compile script** menu item or **F9**.

All scripts are converted during compilation. The assemblies are created and placed on the workstation where generating will take place. During the conversion, the script code is tested for validity. This process may required some time.

Error messages are sent to the **Compiler errors** view. A double-click on the error message takes you straight to the corresponding line in the script code view where you can edit it. It can be modified at this point.

IMPORTANT: Once you have tested the script it needs to be added to the One Identity Manager database and compiled with the Database Compiler. For more information, see the *One Identity Manager Operational Guide*.

Overriding scripts

You might want to label scripts for overriding if there are limits to how much you can modify default scripts. Scripts that can be overwritten are labeled with the `overridable` property.

NOTE: Only the default scripts that are supplied can be overridden. Custom scripts cannot be overridden because these are saved in a `Custom scripts` script class.

To override a script

1. In the Designer, select the script to override in **Script Library > Overridable scripts**.
2. Select the **Copy script** task.
3. In the **Copy script** dialog, edit the following information.
 - **Script:** The name of the new script is made up of the prefix **CCC_** and the name of the old script. You can change the name. Prefix custom scripts with **CCC_**.
4. To create the copy, click **OK**.
5. In the Script Editor, replace the overridable property with overrides in the script header.
6. Modify the other script code accordingly to suit your requirements.

IMPORTANT: After creating and editing the script, you should test compiling the script. Compile the scripts in the script library for this script to take effect.

Syntax example

```
Public overridable Function My_Function() as Boolean
    'script code of the original version
End Function

Public overrides Function My_Function() as Boolean
    'Custom script code
End Function
```

Related topics

- [Creating and editing scripts in the Script Editor](#) on page 365
- [Copying scripts in the Script Editor](#) on page 366
- [Testing script compilation in the Script Editor](#) on page 368
- [Testing scripts in the Script Editor](#) on page 367
- [Visual Basic .NET scripts usage](#) on page 341

Permissions for running scripts

The basic permissions for running scripts are granted to the logged in user through the program feature **Allow the starting of arbitrary scripts from the frontend** (Common_StartScripts).

If a script is assigned a program function (QBMScriptHasFeature table), users can only run this script if they have the necessary permissions groups. An error occurs if the user does not own this program function and tries to run it.

Detailed information about managing permissions and running scripts through program functions can be found in the *One Identity Manager Authorization and Authentication Guide*.

Editing and testing script code with the System Debugger

The System Debugger gives you the opportunity to test scripts, templates, formatting rules, methods, and table scripts. Visual Studio debug and edit options are available to you.

The following software must be installed to use the System Debugger:

- Visual Studio 2012 with the current service pack
- Microsoft .NET Framework Version 4.7.2 Developer Pack or later

NOTE: To use the System Debugger with privileges without starting Visual Studio, you must install the One Identity Manager components in a local directory which is not controlled through user accounts.

Detailed information about this topic

- [Loading the script library](#) on page 370
- [Tips for editing script code in the System Debugger](#) on page 372
- [Logging database queries and object actions](#) on page 372
- [Testing script code in the System Debugger](#) on page 373
- [Saving changes to the database](#) on page 376

Loading the script library

When you call the System Debugger, a SystemLibrary.sln solution template with the **SystemLibrary** solution is loaded in the Visual Studio for editing and testing the scripts.

The following projects are defined in the solution.

Table 141: Solution project files

Project	Script File	Description
Methods	Methods.vb	This script file contains all methods.

Project	Script File	Description
Scripts	VIScripts.vb	This script file contains all predefined scripts from the model components.
	VIDScripts.vb	This script file contains all predefined scripts from the system components.
	CustomerScripts.vb	This script file contains custom scripts. Add new scripts here.
SystemDebugger	Main.vb	Start up project.
Tables	Tables.vb	This script file contains all the table scripts.
Templates	Templates.vb	This script file contains all templates and formatting scripts.

To load the system library

1. Run the SystemLibrary.sln file in the One Identity Manager installation directory.
2. Check whether **SystemDebugger** is entered in Visual Studio as the start project.
3. Start the solution with **F5** in Visual Studio.
4. Connect to the database.
5. Check the solution file directory and the options for creating the script library.

Options	Meaning
Export system scripts	Specifies whether predefined scripts are loaded into the system library.
Export custom scripts	Specifies whether custom scripts are loaded into the system library.
Export locked scripts	Specifies whether only active scripts or also locked scripts are loaded from the script library.
Update project references	Specifies whether references used in scripts are also loaded.
Create backups of existing files	Specifies whether backups of existing files are made.

6. Click **OK**.
The script library files are filled with data from the database.
7. Confirm reloading of each project in Visual Studio.
8. Start the solution with **F5** in Visual Studio.
The source code generated for the solution is compiled.

9. Reconnect to the database.
Starts the System Debugger.

Related topics

- [Testing script code in the System Debugger](#) on page 373

Tips for editing script code in the System Debugger

After loading the system library, you can edit scripts, templates, formatting scripts, methods, and table scripts in the System Debugger and test them.

Note the following:

- You are not permitted to edit VI-Key comments in the source code or to delete them because they label each code block and are needed for backing up scripts in the database.
- When templates and formatting scripts are loaded, the \$ notation is converted into a `GetTriggerValue` method call. All `GetTriggerValue` methods calls are converted into \$ notation when the changes are saved in the database.

Example:

`$FK(UID_Person).IsExternal:Bool$` is converted into `GetTriggerValue("FK(UID_Person).IsExternal").Bool` when it is loaded

- In the Designer, you can use the Script Editor to create scripts. Enter the name of the script in the Script Editor and a skeleton script body. This you can export to the script library where you can edit the script with the System Debugger.
- In the Designer, you can create templates, formatting scripts, methods, and table scripts. You can edit these elements with the System Debugger after you have exported them to the system library.

Related topics






- [Testing script code in the System Debugger](#) on page 373
- [Saving changes to the database](#) on page 376

Logging database queries and object actions

Use database query and object action logging in the System Debugger to look for errors and optimize scripts during development. The processing time and the command that was run are logged.

- SQL log
Open the log dialog box by selecting the **View > SQL log** menu item.
- Object log
Open the log dialog box by selecting the **View > Object log** menu item.

Table 142: Functions for logging database queries and object actions

Icon	Meaning
	Starts recording.
	Stops recording.
	Copies logged data to the clipboard.
	Save logged data in a file.
	Deletes the logged data.

Testing script code in the System Debugger

The System Debugger gives you the opportunity to test scripts, templates, formatting rules, methods, and table scripts. Visual Studio debug and edit options are available to you.

Detailed information about this topic

- [Testing scripts in the System Debugger](#) on page 373
- [Testing templates and formatting scripts in the System Debugger](#) on page 374
- [Testing methods in the System Debugger](#) on page 375
- [Testing table scripts in the System Debugger](#) on page 375

Testing scripts in the System Debugger

To test a script

1. In the System Debugger, select the desired script in **Scripts** view.
2. Enter value for the script parameters as required.
3. Check the options for running the script.
 - **Run in debug mode:** Jumps into the source code. This allows you to use all Visual Studio debugging options.
 - **Define base data:** The Base and Value variables of the script base class can be pre-allocated as input parameters to be used in the script.

Example:


Base is initialized with a DB object key in order to use `base.GetValue("column name").String`.

- **Transaction with rollback:** Use this option to specify whether the script is run within a transaction with subsequent rollback or whether the script is run immediately against the database.

4. Select **Start**.

The script starts running. After the script has run, the result and the processing time of the script is displayed.

TIP: To find scripts more easily, you can use the following functions in the **Scripts** view.

- In the **Find script** field, enter the string to filter on.
- Modified scripts are marked with a * in the System Debugger.
- To find all modified scripts, click  and apply the **Changed scripts** menu item.

Related topics

- [Tips for editing script code in the System Debugger](#) on page 372
- [Saving changes to the database](#) on page 376
- [Testing scripts in the Script Editor](#) on page 367

Testing templates and formatting scripts in the System Debugger

To test a template

1. Select the column with template in System Debugger from **Templates**.
2. Select the column with the template you want to test under **Notifier column**.
3. Select the object in **Database object** to which to apply the template.
4. Check the **Transaction with roll back** option for running the templates.

Use this option to specify whether the template is run within a transaction with subsequent rollback or whether the template is run immediately against the database.

5. Select one of the following actions to test the template.

Action	Meaning
Save	The object is saved.
Discard	The changes made to the object are discarded.

Action	Meaning
Load	The object is reloaded.
New	A new object is created.
Run	The template is run again.

To test a formatting script

1. In the System Debugger, select the column with the formatting script in **Formats**.
2. Select the object in **Database object** to which to apply the formatting script.

Related topics

- [Tips for editing script code in the System Debugger](#) on page 372
- [Saving changes to the database](#) on page 376

Testing methods in the System Debugger

To test a method

1. In the System Debugger, select the method in the **Dialog methods** area.
2. Select the object to apply the method to under **Base object**.
3. Check the **Transaction with roll back** option for running the methods.
Use this option to specify whether the method is run within a transaction with subsequent rollback or whether the method is run immediately against the database.
4. Select **Start**.
The method starts running.

Related topics

- [Tips for editing script code in the System Debugger](#) on page 372
- [Saving changes to the database](#) on page 376

Testing table scripts in the System Debugger

To test table scripts

1. In the System Debugger, select the table and table script from the **Tables** view.
2. Select the object to test the table script on under **Database object**.
3. Check the **Transaction with roll back** option for running the table scripts.

Use this option to specify whether the table script is run within a transaction with subsequent rollback or whether the table script is run immediately against the database.

4. Select the following actions to test the table script.

Action	Meaning
Save	The object is saved. (OnSaved, OnSaving)
Discard	The changes made to the object are discarded. (OnDiscarded, OnDiscarding)
Load	The object is reloaded. (OnLoaded)
New	A new object is created.

Related topics

- [Tips for editing script code in the System Debugger](#) on page 372
- [Saving changes to the database](#) on page 376

Saving changes to the database

To save changes to the database

1. In the System Debugger, select the script, template, formatting script, method, or the table script.
2. Select the **Scripts > Save script** menu item.
This opens a dialog displaying the script name, database object, database connection, and script code to be added.
3. Select a change label under **Change labels** to group your changes.
4. Click **Save**.

TIP: To save several scripts in the **Scripts** view, hold the **CTRL** key down, click on the scripts and select the **Scripts > Save script** menu item.

NOTE: Ensure you recompile the database after making changes.

Extended debugging in the Object Browser

The Object Browser supports debugging of scripts, templates, format scripts, table scripts, processes, and methods. You can make use of the Visual Studio debug options for this

purpose. You cannot alter the scripts, templates, formatting rules, table scripts, processes, or methods. Correct the errors in the Designer.

Prerequisites

- To use the debug function in the Object Browser, you must install the following software:
 - Visual Studio 2012 with the current service pack
 - Microsoft .NET Framework 4.7.2 Developer Pack or later
- The user requires the **Allows local debug assemblies to be created** program function (Common_CompileForDebug). This provides the user with an additional compiler option in the Configuration Wizard for creating local debug assemblies.
In the Designer, assign the program function to a custom permissions group and add the system user to this permissions group. For more information about controlling conditions with program functions, see the *One Identity Manager Authorization and Authentication Guide*.
- Local debug assemblies are available on the user's local workstation.

Detailed information about this topic

- [Creating local debug assemblies](#) on page 377
- [Debugging in the Object Browser](#) on page 378
- [Troubleshooting debugging in the Object Browser](#) on page 379

Creating local debug assemblies

To generate local debug assemblies

1. In the Database Compiler, on the **Compiler settings** page, set the **Create debug information** option.
2. Select the **Scripts including all dependencies** compiler setting.
3. Start the compiler.

During compilation you will see more messages that refer to creating the debug assemblies locally. Some compiler steps, for example, compiling web projects, are skipped because they cannot be debugged locally.

The Database Compiler saves the assemblies and associated PDB files in the %USERPROFILE%\AppData\Local\One Identity\One Identity Manager\AssemblyCache directory on the local computer.

The source code is saved in the %USERPROFILE%\AppData\Local\One Identity\One Identity Manager\AssemblyCache\Sources directory.

Therefore, the assemblies are not transferred to the database.




Debugging in the Object Browser

NOTE: You can debug locally until assemblies without debug data are generated on the workstation or new assemblies are loaded over a database connection.

To debug in the Object Browser

1. Start the Object Browser and select the **Debug > Debugger start/stop** menu item.

Visual Studio opens and the Object Browser is connected to the debugger. This process may take a few moments. If Visual Studio connection can be established, the **Manage breakpoints** dialog opens.

2. In the **Manage breakpoints** dialog, you can define different breakpoints for scripts, templates, format scripts, table scripts, processes, and task definitions.
 - To define a new breakpoint, click  and select one of the following options:
 - **Script processing:** Adds a breakpoint of **Script** type. Under **Breakpoint operation**, select a script.
 - **Column processing:** Adds a breakpoint of **Column** type. Under **Breakpoint operation**, select a format script, template, or script for conditionally removing permissions.
 - **Table processing:** Adds a breakpoint of **Table** type. Under **Breakpoint operation**, select a table script.
 - **Process generation:** Adds a breakpoint of **Process** type. Under **Breakpoint operation**, select a process.
 - **Object method:** Adds a breakpoint of **Object method** type. Under **Breakpoint operation**, select a task definition.
 - To use an existing breakpoint, select it in the list.
 - To delete a breakpoint, select it in the list and click .
 - To delete all breakpoints, click .

3. Click OK.

This closes the **Manage breakpoints** dialog. The breakpoint definitions are transferred to Visual Studio.

NOTE: You can open the **Manage breakpoints** dialog again from the **Debug > Configure breakpoints** menu.

4. In the Object Browser, run the actions that you want to debug, for example, call a script, run a column template, or generate a process.

The moment the action is about to be run, Visual Studio is brought into the foreground and opens the place in the source code with the selected breakpoint. From this point on, all of Visual Studio's comprehensive debugging options are available to you in full. For example, use **F10** to step through the code line for line or use **F5** to continue with the program.

5. After you have completed debugging, select the **Debug > Debugger start/stop** menu item to
disconnect Visual Studio from the Object Browser and close it.

Related topics

- [Creating local debug assemblies](#) on page 377
- [Troubleshooting debugging in the Object Browser](#) on page 379

Troubleshooting debugging in the Object Browser

Problem

In the Database Compiler, the **Create debug information** is not shown.

Cause

The system user has not been assigned the **Allows local debug assemblies to be created** program function (Common_CompileForDebug) through their permissions groups.

Solution

In the Designer, assign the program function to a custom permissions group and add the system user to this permissions group. For more information about controlling conditions with program functions, see the *One Identity Manager Authorization and Authentication Guide*.

Problem

The **Debug** menu is not shown in the Object Browser.

Possible cause

- Visual Studio is not installed with the required options.
- The assemblies do not contain debug information.

Possible solutions

- Check your Visual Studio installation on the local workstation. For more information, see [Extended debugging in the Object Browser](#) on page 376.
- Check the database connection. The debug assemblies always belong to a fixed database. If another database connection is selected, the debug information is not

longer available.

- Check whether new assemblies have been loaded from the database or not. The date of DLL and PDB files must be the same.
- Recompile the assemblies with debug information, if necessary. For more information, see [Creating local debug assemblies](#) on page 377.

Problem

Breakpoints are shown as disabled in Visual Studio.

Cause

Breakpoints are shown as disabled if the assembly with the function to be debugged is yet not loaded into memory. For example, the assemblies for generating processes are not loaded into the application until the point of generation. From then on the breakpoint is enabled and you can jump to it.

Problem

Breakpoints in Visual Studio are always shown as disabled and you cannot jump to them.

Possible cause

- The Object Browser still has the wrong assemblies loaded.
- The Object Browser could not find debug information for the assemblies.

Possible solution

If Visual Studio is connected the Object Browser, switch to Visual Studio and open the **Debug > Windows > Modules**. Here you will find a list of all the modules that are loaded and additional information.

One Identity Manager query language

The One Identity Manager query language can be used to create queries or Where clause expressions against the One Identity Manager object layer.

For example, the One Identity Manager query language is used to communicate between application servers and clients. Currently, you can use the One Identity Manager query language in the Object Browser's query window .

The query language is not case sensitive. Keywords can be written in upper or lower case. To be able to identify them more easily, keywords are always written in capital letters in the following examples.

In a lot of parts, the query syntax is based on SQL syntax. This makes it easier to convert simple Where clauses from SQL syntax to the query syntax.

Detailed information about this topic

- [Language elements of the One Identity Manager query language](#) on page 381
- [Formulating queries in the One Identity Manager query language](#) on page 386

Language elements of the One Identity Manager query language

The One Identity Manager query language supports different language elements.

Detailed information about this topic

- [Comments](#) on page 382
- [Identifier](#) on page 382
- [Literal values](#) on page 383

- [Parameter references](#) on page 385
- [Preformatted Where clauses](#) on page 385

Comments

There are two types of comments in the One Identity Manager query language. These are analog to comments in SQL syntax:

Line comments

Line comments start with `--` and end at the end of the line.

```
-- This is a line comment
```

Line comments do not have to be at the beginning of the line.

```
FROM Person -- This is a line comment too
```

Block comments

Block comments start with the `/*` and end with `*/`. Block comments can span more than one line.

```
/* This is a block comment  
that spreads over  
more than one line */
```

Identifier

In the One Identity Manager query language, identifiers always start with an ASCII character in the range A to Z or with an underscore (`_`). Digits are valid only after the first position. Identifiers can contain several parts delimited by dots (`.`).

Examples of valid identifiers

Name

Name1

_Name

Name_1

Alias.Name

Identifiers can be compared using the following regular expression.

```
[A-Z_][A-Z0-9_.*]*
```

Literal values

Literals define values of certain data types in the query. The following literal values are supported in the One Identity Manager query language.

Detailed information about this topic

- [String values](#) on page 383
- [Integer values](#) on page 383
- [Decimal values](#) on page 384
- [Date and time values](#) on page 384

String values

In the One Identity Manager query language, strings are enclosed in single quotes. Single quotes within strings are masked by doubling the single quote. All string contents are Unicode. The N character at the beginning of the string is allowed to ensure some compatibility with SQL syntax. It has no special meaning.

Examples of valid strings

```
''  
N''  
'A string'  
'A ''string'''  
N'A string'
```

Strings can be compared using the following regular expression.

```
N?'([^\']|'')*'
```

A special case of strings are multi-valued properties. These are strings delimited by ASCII 7 characters. Multi-valued properties can be expressed in the query syntax in this form:

```
MVP('Value 1', 'Value 2', 'Value 3')
```

The resulting value can be used in most places where strings are valid.

Integer values

In the One Identity Manager query language, integer values contain digits and an optional plus sign (+) or minus sign (-).

Examples of permitted values

42

+42

-42

Integer values can be compared using the following regular expression.

```
[+-]?\\d+
```

Decimal values

In the One Identity Manager query language, decimal values contain digits and an optional plus sign (+) or minus sign (-). The decimal part is separated by a dot (.).

Examples of permitted values

3.14

+3.14

-3.14

Decimal values can be compared using the following regular expression.

```
[+-]?\\d*\\.\\d+
```

Date and time values

In the One Identity Manager query language, date and time values are specified in ISO notation. The time part is optional. If a time is given, seconds and the milliseconds are also optional.

By default, the time values are specified in the UTC time zone. Alternatively, a time zone can be specified with the `TIMEZONE` keyword and a string containing the time zone's name. In date ranges, the `TIMEZONE` name comes at the end of the format value and applies to both date and time.

Examples of permitted values

2020-04-01

2020-4-1

2020-04-01 12:30

2020-04-01 12:30:24

2020-04-01 12:30:24.523

2020-04-01 12:30:24 TIMEZONE 'W. Europe Standard Time'

Date values can be compared using the following regular expression.

`d{4}-\d{1,2}-\d{1,2} \d{1,2}:\d{1,2}(:\d{1,2}(\.\d{1,3}))?`

Parameter references

In the One Identity Manager query language, query parameters are specified in SQL syntax.

`@ParameterName`

The parameter name matches the identifiers' definition.

Examples of valid parameter references

`@Parameter`

`@Parameter1`

`@Parameter_1`

`@Parameter.SecondPart`

Parameters can be compared using the following regular expression.

`@[A-Z_][A-Z0-9_\.]*`

Related topics

- [Identifier](#) on page 382

Preformatted Where clauses

As it is not possible to write all possible Where clauses due to the limitations of the One Identity Manager query language, you can incorporate preformatted Where clauses written in the SQL syntax of the underlying database system.

Preformatted where clauses are enclosed in square brackets ([and]). Opening square brackets in the clause can be masked with [[].

Example of a preformatted Where clause

`[isnull(lastname, '') = N'Harris']`

Preformatted Where clauses can be compared using the following regular expression.

`[([^\]]|[[])]*\]`

Related topics

- [Using preformatted Where clauses](#) on page 398

Formulating queries in the One Identity Manager query language

The One Identity Manager query language can be used to create queries or Where clause expressions against the One Identity Manager object layer.

The query language is not case sensitive. Keywords can be written in upper or lower case. To be able to identify them more easily, keywords are always written in capital letters in the following examples. In a lot of parts, the query syntax is based on SQL syntax.

Detailed information about this topic

- [Query header](#) on page 386
- [Where clauses](#) on page 387
- [Search clauses](#) on page 387
- [Select clauses](#) on page 387
- [Order by clauses](#) on page 389
- [Paging clauses](#) on page 389
- [Display value clauses](#) on page 390
- [Query hints](#) on page 392
- [Conditions](#) on page 392
- [Comparing columns](#) on page 394
- [Comparison by means of IN and NOT IN clauses](#) on page 395
- [Compare date differences](#) on page 396
- [Compare date ranges](#) on page 396
- [Compare fixed values](#) on page 397
- [Comparing parameters](#) on page 398
- [Using preformatted Where clauses](#) on page 398

Query header

Data queries in the One Identity Manager query language always start with the FROM keyword and a table name. An optional alias for the table can be specified after the AS keyword.

```
FROM <table name>
```

```
FROM <table name> AS <table alias>
```

An alternative is to query directly using an object key.

Example of the query in long form

```
FROM Table WHERE PRIMARYKEY '<Key><T>Table</T><P>UID</P></Key>'
```

Example of the query in short form

```
FROM '<Key><T>Table</T><P>UID</P></Key>'
```

You can specify a list of clauses after the query header. The clause types do not have to be in any particular order.

Where clauses

Where-clauses in the One Identity Manager query language start with the **WHERE** keyword and contain a condition that the resulting data must match. Unlike SQL syntax, multiple Where clauses can be combined in a One Identity Manager query language query.

Examples of Where clauses

```
WHERE Lastname = 'Harris'  
WHERE Lastname = 'Harris' AND Firstname = 'Clara'  
WHERE [isnull(lastname, '') = N'Harris']  
WHERE Lastname IN ('Harris', 'Basset')  
WHERE XDateInserted IN RANGE LAST MONTH
```

Related topics

- [Conditions](#) on page 392

Search clauses

In the One Identity Manager query language the **SEARCH** keyword is used to search for all matching entries from the full text index. To use this type of clause, a full text index must be installed and accessible from the application.

```
SEARCH 'Search string'
```

Select clauses

The One Identity Manager query language Select clauses describe the data that is fetched from the underlying database. You define the minimum amount of returned data. The

object layer can select more data to meet arbitrary requirements. An example is the primary key, which is always selected, or special columns like the X columns.

Multiple select clauses can be combined into one query.

Different variations of the Select clause are supported.

Selecting specific columns of a table

```
SELECT COLUMNS <list of columns>
```

Example

```
FROM Person
```

```
SELECT COLUMNS Firstname, Lastname, CentralAccount
```

Select all columns of a table

```
SELECT ALL
```

Selecting all display values of a table

```
SELECT DISPLAYS
```

Display columns that are selected are:

- The table's display pattern (DialogTable.DisplayPattern)
- The table's display pattern (long) (DialogTable.DisplayPatternLong)
- Primary key of the table

In the case of returned entries, the table's display pattern and the display pattern (long) can be overwritten by a display value clause. For more information, see [Display value clauses](#) on page 390.

Selecting the table's display pattern

```
SELECT DISPLAYPATTERN
```

Selects only the columns from the table's display pattern or DISPLAY clause if it is used. This can provide improved performance over SELECT DISPLAYS because fewer columns are selected.

Select all columns of a table that are not marked as a BLOB field

```
SELECT NONLOBS
```

Selects all columns of the table that are not marked as a very long binary object or text object by the DialogColumn.IsBlobExternal column.

A matching entry exists

```
EXISTS
```

Determines whether an entry exists that fulfills the WHERE clause. The Exist clause overrides all other Select clauses except the Count clause.

Determine the number of matching entries

COUNT

Counts the entries that fulfill the where clauses. The Count clause overrides all other clauses.

Order by clauses

The Order by clause in the One Identity Manager query language, specifies the order in which entries are returned.

ORDER BY <list of columns>

The column order can be specified by

- ASC or ASCENDING (Standard)
- DESC or DESCENDING

When display values are selected, a default order by clause is created for the display columns.

Examples of Where clauses

ORDER BY Lastname, Firstname

ORDER BY EntryDate DESC

You can specify fallback columns, which will be used for sorting if the previous values are **NULL**. These fallback values are specified by the null coalescing operator ??.

Examples for a fallback column's data

ORDER BY DisplayName ?? CN

ORDER BY DisplayName ?? CN DESC, XDateInserted

Paging clauses

Paging operators of the One Identity Manager query language make it possible to return only a subset of the selected entries.

Take clause

The Take clause specifies the maximum number of entries to be returned. If more than one Take clause is specified, only the last take clause is effective.

TAKE <integer>

Example

```
-- Return only the first ten persons from the result set
FROM Person SELECT DISPLAYS TAKE 10
```

Skip clause

The Skip clause specifies how many entries should be skipped from the beginning of the results before the entries are returned.

SKIP <integer>

Example

```
-- Skip 50 persons and return the following 15
FROM Person SELECT DISPLAYS SKIP 50 TAKE 15
```

Display value clauses

Display value clauses allow the definition of a user-defined display pattern. The table's display pattern and the display pattern (long) can be overwritten by a display value clause in the returned entries.

```
DISPLAY 'Display pattern'
LONGDISPLAY 'Display pattern'
```

The parameter in both cases is a string containing the display pattern with placeholders in the form %ColumnName%.

Example

```
FROM Person
SELECT DISPLAYS
DISPLAY '%Lastname%, %Firstname%'
LONGDISPLAY '%Lastname%, %Firstname% - %CentralAccount%'
```

Related topics

- [Select clauses](#) on page 387

Query parameter clauses

In the One Identity Manager query language, query parameter clauses allow values to be passed in parameters that can be used in where clauses. You can pass a single parameter or a list of parameters.

Syntax for a single parameter

```
PARAM <Parameter name> [ OF <Type> ] = <Value>
```

Syntax for multiple parameters

```
PARAMS
```

```
<Param1> [ OF <Type> ] = <Value>,  
<Param1> [ OF <Type> ] = <Value>,
```

Permitted parameter names are:

- Syntax for parameter references with a qualifying @ character
- Identifier

Valid types are the .Net data types of the object layer defined in `ValType` enumeration. If the type can be derived from the value, you do not have to give a type.

Examples of query parameters

```
PARAM Parameter1 = 'Harris'  
PARAM @Parameter2 = 'Harris'  
PARAM Parameter3 OF String = 'Harris'  
PARAM @Parameter4 OF String = 'Harris'  
PARAM Parameter5 = 42  
PARAM Parameter6 OF Int = 42  
PARAMS  
    Parameter7 OF Double = 3.14,  
    Parameter8 OF Date = 2020-04-30
```

Example: Complete query with parameter reference and definition

```
FROM Person  
WHERE LastName = @Param1  
SELECT DISPLAYS
```

```
PARAM Param1 = 'Harris'
```

Related topics

- [Identifier](#) on page 382
- [Parameter references](#) on page 385

Query hints

Query hints (keyword HINT) can be used in the One Identity Manager query language to provide the query processor with additional data. Query hints are used internally to provide a context for additional permission queries to display columns that are only available when loaded as display values for foreign keys of another table.

```
HINT 'Name' = 'Value'
```

Example: loading a Person object referenced by a Department object as UID_PersonHead

```
FROM Person
WHERE PRIMARYKEY '<Key><T>Person</T><P>99918ef1-113f-480a-8e6e-704b1a3cf73a</P></Key>'
SELECT DISPLAYS
HINT 'SourceContext' = 'Org'
```

Conditions

Conditions combine all expressions that filter the resulting entities according to one or more criteria.

In the One Identity Manager query language, conditions can be linked with the AND and OR operators. You can overwrite operator precedence with curly brackets ({ }). Conditions can be inverted using the NOT keyword.

Example of a condition

```
WHERE Lastname = 'Harris'
AND (Firstname = 'Fred' OR Firstname = 'Sam')
```


Related topics

- [Special conditions](#) on page 393

Special conditions

Selecting an entity using the primary key

In the One Identity Manager query language, use the PRIMARYKEY keyword to select the entity with the matching primary key. The primary key must be given in object key notation.

```
WHERE PRIMARYKEY 'Object Key'
```

Example

```
FROM Person
```

```
WHERE PRIMARYKEY '<Key><T>Person</T><P>99918ef1-113f-480a-8e6e-704b1a3cf73a</P></Key>'
```

```
SELECT DISPLAYS
```

Selecting an entity using a key

Use the KEY keyword to select the entity with the matching key. Keys can be object keys or alternative object keys, each in their XML notation.

```
WHERE KEY 'Key'
```

Example

```
FROM ADSAccount
```

```
WHERE KEY '<Key><Table Name="ADSAccount" Key="c149784b-6386-45d7-a38d-3c6e8e1b69d4"><Prop Name="UID_ADSAccount"><Value>c149784b-6386-45d7-a38d-3c6e8e1b69d4</Value></Prop></Table></Key>'
```

```
SELECT COLUMNS cn
```

Select using predefined SQL

With the LIMITEDSQL keyword you use a preformatted Where clause from the QBMLimitedSQL table. Any parameters used in it can be specified with the PARAMETER or the PARAMETERS clause.

```
WHERE LIMITEDSQL 'Identifier'
```

Comparing columns

Columns can be compared against another set of targets. In the One Identity Manager query language, all these comparisons start with the column name.

WHERE <Column> <Operator> <Operand>

The following operators are supported when comparing columns.

Operator	Program	Type of operand
=	Equal	Value that matches the column type, the parameter, or another column
<>	Not equal	Value that matches the column type, the parameter, or another column
<	Less than	Value that matches the column type, the parameter, or another column
>	Greater than	Value that matches the column type, the parameter, or another column
<=	Less than or equal to	Value that matches the column type, the parameter, or another column
>=	Greater than or equal to	Value that matches the column type, the parameter, or another column
LIKE	Match with a specified pattern (as in SQL operator like)	String or multivalued property
NOT LIKE	No match with a given pattern	String or multivalued property
STARTSWITH	String starts with	String or multivalued property
ENDSWITH	String ends with	String or multivalued property
CONTAINS	String contains	String or multivalued property
BITSSET	The given bit positions are set	Integer value or parameter
BITSNOTSET	The given bit positions are not set	Integer value or parameter

Examples of column comparisons

WHERE Lastname = 'Einstein'

WHERE XDateInserted > 2020-02-01

WHERE Lastname STARTSWITH 'Ein'

WHERE XMarkedForDeletion BITSSET 2

Comparison by means of IN and NOT IN clauses

In the One Identity Manager query language, comparisons with the keywords IN and NOT IN enable the comparison of a column with a set of values.

The values can be given

- In list form
- As a multivalued string, delimited by an ASCII 7 character
- As a parameter containing a multivalued string delimited by an ASCII-7 character

Value lists

```
WHERE <Column> IN ( <Value>, <Value>, ...)
```

```
WHERE <Column> NOT IN ( <Value>, <Value>, ...)
```

All values must be of the same type and must be convertible to the column type.

Examples of value lists

```
WHERE StringColumn IN ('Value 1', 'Value 2', 'Value 3')
```

```
WHERE StringColumn NOT IN ('Value 1', 'Value 2', 'Value 3')
```

Multivalued string

```
WHERE <Column> IN 'Separated string value'
```

```
WHERE <Column> NOT IN 'Separated string value'
```

Example for multivalued strings

```
WHERE Lastname IN 'Harris de Harris'
```

Parameters

Multivalued strings can be passed to queries by parameters.

```
WHERE <Column> IN @Parameter
```

Parameter examples

```
WHERE Lastname IN @Lastnames
```

You can also use MVP syntax. The normal IN clause syntax should be preferred in this case. For more information, see [String values](#) on page 383.

Compare date differences

A special case of column comparisons in the One Identity Manager query language are date difference comparisons. These compare the value of a column with a time range based on the current time.

WHERE <Column> <Operator> DATE <Integer> <Unit> AGO

Time unit	Keyword
Years	YEARS YEAR Y
Months	MONTHS MONTH M
Weeks	WEEKS WEEK
Days	DAYS DAY D
Hours	HOURS HOUR H
Minutes	MINUTES MINUTE MIN
Seconds	SECONDS SECOND S
Milliseconds	MILLISECONDS MILLISECOND MS

Examples

WHERE XDateInserted < DATE 3 MONTHS AGO

WHERE XDateInserted > DATE 5 MIN AGO

Compare date ranges

Comparisons of date ranges in the One Identity Manager query language check whether a date falls within a given date range.

Syntax for named ranges

WHERE <Column> IN RANGE <Range name> [TIMEZONE 'Timezone ID'] [CULTURE 'Culture ID']

Syntax for a specific time range

WHERE <Column> IN RANGE <Start time> TO <End time> [TIMEZONE <Timezone ID>]

Area start time is included, end time is excluded.

Permitted identifiers are:

TODAY

YESTERDAY
THIS WEEK
THIS MONTH
THIS YEAR
LAST WEEK
LAST MONTH
LAST YEAR
LAST <integer> DAYS
LAST <integer> DAY

The names of the time zones must match the identifiers on the system running the query. If no time zone is given, UTC is used.

The culture indicates which day starts the week. It is only useful if one of the week ranges is specified.

Examples

```
WHERE XDateInserted IN RANGE YESTERDAY
WHERE XDateInserted IN RANGE YESTERDAY
      TIMEZONE 'W. Europe Standard Time'
WHERE XDateInserted IN RANGE LAST YEAR
WHERE XDateInserted IN RANGE THIS WEEK
      TIMEZONE 'W. Europe Standard Time'
      CULTURE 'de-DE'
WHERE XDateInserted IN RANGE 2020-01-01 TO 2020-02-01
      TIMEZONE 'W. Europe Standard Time'
```

Related topics

- [Date and time values](#) on page 384

Compare fixed values

In the One Identity Manager query language, comparisons between fixed values support only the operators equal to (=) and not equal to (<>).

Example

```
FROM Person
WHERE 1 = 0
```

SELECT DISPLAYS

Comparing parameters

In the One Identity Manager query language, parameters can only be compared against fixed values. All comparisons that are valid for columns can also be used for parameters.

| NOTE: IN clauses and NOT IN clauses cannot be used.

Examples

```
WHERE @StringParameter = ''
```

```
WHERE @IntParameter > 5
```

```
WHERE @IntParameter BITSET 4
```

Related topics

- [Comparing columns](#) on page 394

Using preformatted Where clauses

More complex Where clauses can be specified in the One Identity Manager query language in the format of the underlying database system. They are checked for SQL injection attempts when they are run. Preformatted where clauses are enclosed in square brackets ([and]).

```
WHERE [preformatted clause]
```

Example

```
FROM Person
```

```
WHERE [isnull(LastName, N'') = N'Einstein']
```

```
SELECT DISPLAYS
```

Related topics

- [Preformatted Where clauses](#) on page 385

Reports in One Identity Manager

One Identity Manager provides the means to create and run multi-object reports, including totals and other aggregate functions. It is also possible to create groups and graphically represent data. Predefined reports are supplied with the schema installation. You can create and edit custom reports with Report Editor.

You can also send reports to specified email addresses using scheduled subscriptions. You can create reports for the current state or over a specified period. For every report, you can create different subscribable reports that can be requested by Web Portal users. In addition, you can embed reports in the Manager or the Designer's user interface.

For more information about report subscription, see the *One Identity Manager Report Subscriptions Administration Guide* and the *One Identity Manager Web Designer Web Portal User Guide*.

Detailed information about this topic

- [Creating and editing reports in the Report Editor](#) on page 404
- [Example of a simple report with data grouping](#) on page 428
- [Translating reports](#) on page 432
- [Embedding reports in the user interface](#) on page 433
- [Generating and exporting reports on a cyclical basis](#) on page 434

Working with the Report Editor

The Report Editor is a program for creating and editing reports. The program uses StimulReport.Net components for designing the reports. You can find accurate descriptions and the functionality of individual components in the Stimulsoft online help (www.stimulsoft.com).

NOTE: When you start the Report Editor for the first time, you can select the configuration type (**basic**, **default** or **professional**) for the report. The configuration type determines the range of properties displayed when editing a report. You can change the configuration type later in the edit view using the context menu in the property view.

NOTE: Reports with changes in historical data analyze changes made to data in a One Identity Manager History Database. If the One Identity Manager History Database is linked by an ID to the One Identity Manager database's TimeTrace, you must log in to the Report Editor through an application server that has this ID in its configuration file (`web.config`). For more information about connecting to the One Identity Manager History Database through an application server and the required configuration, see the *One Identity Manager Operational Guide*.

Menu items in Report Editor

Table 143: Meaning of items in the menu bar

Menu	Menu item	Meaning
Database	New connection	Establishes a database connection.
	Settings	For configuring general program settings.
	Exit	Exits the program.
Report	New	Creates a new report.
	Save	Saves the current report in the database.
	Delete	Deletes the current report.
	Edit	Opens the property dialog for the current report.
	Reload data	Reloads the report data from the database.
	New virtual data source	Opens a dialog box for creating a virtual data source.
Help	Community	Opens the One Identity Manager community website.
	Support portal	Opens the One Identity Manager product support website.
	Training	Opens the One Identity Manager training portal website.
	Online documentation	Opens the One Identity Manager documentation website.
	Search	Opens the search dialog box.
	Report Editor help	Opens program help.
	Info	Shows the version information for program.

Table 144: Meaning of icons in the general toolbar









Icon	Meaning
	Creates a new report.
	Deletes the current report.
	Saves the current report in the database.
	Opens a dialog box for editing change labels.
	Defines the current change label as default and applies it automatically.
	Opens the property dialog for the current report.
	Reloads with the newest report data.
	Opens a dialog for creating a new virtual data source.

Table 145: Functions in the report list toolbox





Icon	Meaning
	Displays all reports.
	Uses a filter condition to limit the number of reports displayed.
	Runs the filter and shows all reports that satisfy the filter condition. The filter condition is interpreted internally as a LIKE comparison.
	Updates the report list.

Table 146: Functions in the report list context menu

Context Menu Item	Meaning
New	Creates a new report.
Edit	Opens the property dialog box for the current report.
edit properties	Loads the properties dialog box for the selected report.
Copy	Copies the selected report.
Delete	Deletes the current report.

Views in the Report Editor

The Report Editor has several views for editing reports.

Table 147: Report Editor views

View	Description
Report list	All reports are displayed by category. Uses a filter condition to limit the number of reports displayed.
Edit view for reports	Reports are designed with the Report Designer in the edit pane. Using the Report Designer's toolbar, you can place the controls you want on the report form. NOTE: Use the online help from Stimulsoft StimulReport.Net (www.stimulsoft.com) as a basis for the report design.
Property dialog box	Use the view edit the properties of the selected report. A default context menu is available for input fields.
SQL log	Database queries are listed in this view. Use query logging to look for errors and to optimize the report during the design phase. For more information, see Logging database queries on page 403.

Report Editor program settings

General configuration settings are specified in a ReportEdit2.exe.config configuration file. Valid global configuration settings can also be defined through the Global.cfg global configuration file in One Identity Manager's own format. The configuration files are stored in the program directory. For more information, see the *One Identity Manager Process Monitoring and Troubleshooting Guide*.

To change the program settings in the Report Editor

1. In the Report Editor, select the **Database > Settings** menu item.
2. In the **Language settings** pane, modify the following:
 - **Language:** Language used for formatting data, such as date formats, time formats, and number formats.
 - **Other user interface language:** Language for the user interface. The initial program login uses the system language for the user interface. Changes to the language settings take effect after the program has been restarted. The language is set globally for all One Identity Manager programs, which means the language setting does not have to be configured for each program individually.
3. In the **Behavior** pane, modify the following:
 - **Show code tab:** Specifies whether the tab for editing the script code is shown in the Report Designer.
 - **Ask on save without change label:** You should book changes to reports to a change label. Set this option so that an alert box is called when changes are

saved without a change label.

- **Max. number of previews:** Maximum number of data records that are used to preview the report.
4. Save the settings with **OK**.

Logging database queries

Use database query logging in the Report Editor to look for errors and to optimize the report during the design phase. The processing time and the command that was run are logged.

To start and stop writing to the log

1. In the Report Editor, open the log window using **SQL log** at the bottom of the program.
2. (Optional) Use the **Select column** context menu, specify which columns are shown in the log.
3. To start logging, select **Start/Stop** from the context menu.
4. To stop logging, select **Start/Stop** from the context menu again.




















TIP: You can configure how the messages are displayed in the error log. To do this, switch the log to advanced mode by clicking  on the right of the column headers.

Table 148: Meaning of icons in the log

Icon	Meaning
	Logs all critical error messages. (Info level Fatal)
	Logs all information. (Info level Info)
	Logs all warnings. (Info level Warning)
	Logs all error messages. (Info level Error)
	Logs debugger output. This setting should only be used for testing. (Info level Debug)
	Logs highly detailed information. This setting should only be used for analysis purposes. The log file quickly becomes large and cumbersome. (Info level Trace)
	Adds a custom filter condition.
	Deletes filter condition.
	Searches for term.
	Searches next term.

Icon	Meaning
	Marks all messages with a specific term.
Buffer size	Sets the message buffer size. The buffer's level is displayed next to the field.
	Deletes the buffer contents.
	Stops logging.
	Starts logging.
	Saves log to file.
	Specifies which column are displayed in the error log.
	Copies selected messages to the clipboard.
	Opens the error log with a text editor.

Creating and editing reports in the Report Editor

NOTE: Reports with historical data changes analyze data changes in a One Identity Manager History Database. If the One Identity Manager History Database is linked by an ID to the One Identity Manager database's TimeTrace, you must log in to the Report Editor through an application server that has this ID in its configuration file (`web.config`). For more information about connecting to the One Identity Manager History Database through an application server and the required configuration, see the *One Identity Manager Operational Guide*.

Create and edit reports with the Report Editor program. Reports are stored in the database DialogReport table. The following steps are required to create a report:

1. Defining report properties, data sources, and report parameters.
2. Designing the report form with the Report Designer.

Predefined reports supplied with One Identity Manager by default, automatically customized during schema installation. If customizations are required to the default reports:

1. Create a copy of the report.
2. Edit the required report properties.
3. Use the customized report from now on.

When you add or copy a report, the property dialog box opens first, which you use to enter the general data for the report, the data source required and an parameters for the report definition. Then a new report form is created in the edit view with the Report Designer. This

forms the basis of the report design. Using the Report Designer's toolbar, you can place the controls you want on the report form.

NOTE: Use the online help from Stimulsoft StimulReport.Net (www.stimulsoft.com) as a basis for the report design.

To create a new report

- In the Report Editor, select the **Report > New** menu item.

To copy a report

- In the Report Editor, select the report in the report list and then, in the context menu, click **Copy**.

This creates a new report and the property dialog opens. The properties in the new report are taken from the original.

To edit a report

1. In the Report Editor, select the report in the report list and open it by double-clicking or clicking **Edit** in the context menu.

This opens the report form in the Report Designer.

2. To open the property dialog, select the **Report > Edit** menu item.

To edit the report properties without loading the report in the Report Designer

- In the Report Editor, select the report in the report list and then **Edit properties** from the context menu.

This opens the property dialog.

NOTE: After you have customized a report, you can mark it by setting change labels. These change labels are offered in the Database Transporter as export criteria when a customer transport package is created.

Detailed information about this topic



- [Editing general report properties](#) on page 406
- [Creating and editing data sources](#) on page 407
- [Report parameters](#) on page 418
- [Using virtual data sources](#) on page 425
- [Editing the report form](#) on page 425

Editing general report properties

To edit general report properties

1. In the Report Editor, open the report.
2. Select the **Properties** tab in the properties dialog.
3. Edit the general properties.
4. Save the changes.

Table 149: General report properties

Property	Meaning
Name	Report name Label custom reports with the CCC_ prefix.
Display name	<p>Display name of the report. The display name is available when the report is created as ReportAlias. It can, for example, be used to compose the title of the report or the file name when you export a report in the Web Portal. Translate the given text using the  button.</p> <p>The report display name can contain variables, permitted are system variables such as report parameters. The variables are passed using a percent character.</p> <p>Example:</p> <p>Name of report %variable%</p>
Max. runtime [sec]	Maximum number of seconds available to generate the report If this period is exceeded, the report stops generating.
Description	Report description. Translate the given text using the  button.
Filter criteria	Filter criteria for displaying the report in the web front-end.
Base table	Basis table for the report.
Category	Category for classifying reports. Permitted values are the Common , Mail , Attestation , and Dashboard categories.
Preprocessor condition	Preprocessor conditions can be added to reports. In this case, a report is only available if the preprocessor condition is fulfilled.
Custom properties > Spare field no. 01 ... Spare field no. 10	Additional company-specific information. Use the Designer to customize display names, formats, and templates for the input fields.
Extended properties > Report	UID for finding the report in the database.

Related topics

- [Creating and editing data sources](#) on page 407
- [Report parameters](#) on page 418


Creating and editing data sources

For each report you need to create a data source from which to read the report data to be displayed. Normally one data source is sufficient for one report. However, you can define several data sources for each report. You can test the results while processing a data source.


To edit a data source

1. In the Report Editor, open the report .
2. Select the **Data source** tab in the properties dialog box.
3. Select the data source from the **Defined queries** list.
- OR -
Click **Add**.
This creates a new data source.
4. Edit the data source properties.
5. Save the changes.

To test a data query

1. In the Report Editor, open the report .
2. Select the **Data source** tab in the properties dialog box.
3. Select the data source from the **Defined queries** list.
4. Click the  button next to **Query module**.

The result of a data source is shown in a separate dialog.

NOTE: When a data query is copied to the clipboard, a database query is generated in SQL syntax, which you can run on the database with an appropriate SQL query tool. To copy the data query, use the  button next to **Query module**.

To delete a data source

1. In the Report Editor, open the report .
2. Select the **Data source** tab in the properties dialog box.
3. Select the data source from the **Defined queries** list.
4. Click **Delete**.
5. Save the changes.

Detailed information about this topic

- [Data retrieval using SQL queries](#) on page 408
- [Data retrieval using database views](#) on page 409
- [Data retrieval using an object](#) on page 410
- [Data retrieval using single object history](#) on page 411
- [Data retrieval using multiple object history](#) on page 413
- [Data retrieval using historical assignments](#) on page 415
- [Data query for simulation data](#) on page 417

Related topics

- [Using virtual data sources](#) on page 425

Data retrieval using SQL queries

Data queries with the **SQL** query module are run directly on the database without checking user access permissions. This means that a column to be used in the report is displayed even though the user may not have access permission to it.

Table 150: Data source SQL properties

Property	Meaning
Name	Name of the data source.
Description	Description of data source.
Max. lines	Maximum number of result lines for this query. NOTE: The report only displays results up to this maximum even if the number of results exceeds it. In the default, no error messages or tips are displayed. Any possible messages must be customized in the report.
Parent query	Not used.
Query module	Select the SQL query module.
Query	Full database query SQL syntax. The query must contain all the columns used in the report. You can also use SQL parameters in the query. Add these parameters subsequently to the report by entering them on the Parameters tab. Syntax for parameters: <code>@<parameter name></code> Syntax for parameters of Date data type and a scope (time period

Property	Meaning
	from/until):
	@<parameter name>Start
	@<parameter name>End
	Example of usage in the SQL query:
	and StartDate between @<parameter name>Start and @<parameter name>End

Example:

The query should return the employees (Person table) assigned to an department. The department (UID_Department) is found with the object key (XObjectKey). This is passed as a parameter to the report. The query queries employee's first name (firstname), last name (lastname), and department name (departmentname).

```
Select Firstname, Lastname, Departmentname
    from person join Department
    on person.uid_Department = department.uid_Department
    where Department.XObjectKey = @XObjectKeyBase
```

Related topics

- [Creating and editing data sources](#) on page 407
- [Report parameters](#) on page 418

Data retrieval using database views

You can use the **View** query module to create data queries using predefined database views and thus control user access permissions.

Table 151: Data source view properties

Property	Meaning
Name	Name of the data source.
Description	Description of data source.
Max. lines	Maximum number of result lines for this query. If this number is exceeded, the report stops generating.

Property	Meaning
Parent query	Not used.
Query module	Select the View query module.
View name	Name of the database view.
Condition	<p>Condition for limiting the data set returned from the database table. You formulate the condition as a valid WHERE clause for database queries. You may use SQL parameters in the condition. Add these parameters subsequently to the report by entering them on the Parameters tab.</p> <p>Syntax for parameters:</p> <p>@<Parametername></p>
Sort order	The data queries are sorted by these database view columns.

Related topics

- [Creating and editing data sources](#) on page 407
- [Report parameters](#) on page 418

Data retrieval using an object

Data queries with the **Object** query module are created using the object layer and therefore take user access permissions fully into account.

Table 152: Data source object properties

Property	Meaning
Name	Name of the data source.
Description	Description of data source.
Max. lines	<p>Maximum number of result lines for this query.</p> <p>NOTE: The report only displays results up to this maximum even if the number of results exceeds it. In the default, no error messages or tips are displayed. Any possible messages must be customized in the report.</p>
Parent query	In a parent query, restrictions are applied to the data record that are passed on to subsequent queries, all members of a department, for example. Parameters that are defined in the parent query are also available in subsequent queries.
Query module	Select the Object query module.

Property	Meaning
Table	Select the table to find the object in.
Columns	<p>Columns to use in the report.</p> <p>Some columns are always added to the report definition and must not be explicitly entered here. These include:</p> <ul style="list-style-type: none"> • The table's primary key column. • All columns used in the table display template. • Pseudo columns (<code>_Display</code> and <code>_DisplayLong</code>) supplied by the table's display template. • An additional column (<code><column>_Display</code>) is also created for the display value for foreign key columns and columns with a list of defined values or multi-language entries.
Resolve foreign key	Set this option if the display value of the referenced object should be returned in <code><column>_Display</code> rather than the UID.
Condition	<p>Condition for limiting the data set returned from the table. You formulate the condition as a valid WHERE clause for database queries. You may use SQL parameters in the condition. Add these parameters subsequently to the report by entering them on the Parameters tab.</p> <p>Syntax for parameters:</p> <p><code>@<Parametername></code></p> <p>Syntax for columns of a parent query:</p> <p><code>@<name of parent query>.<column of the parent query></code></p>
Sort order	The data queries are sorted by these table columns.

Related topics

- [Creating and editing data sources](#) on page 407
- [Report parameters](#) on page 418

Data retrieval using single object history

Use data queries with the **Single object history** query module when you want to create reports about a single object, for example, one employee, with its history data.

Table 153: Properties of data source single object history

Property	Meaning
Name	Name of the data source.

Property	Meaning
Description	Description of data source.
Max. lines	Maximum number of result lines for this query. NOTE: The report only displays results up to this maximum even if the number of results exceeds it. In the default, no error messages or tips are displayed. Any possible messages must be customized in the report.
Parent query	In a parent query, restrictions are applied to the data record that are passed on to subsequent queries, all members of a department, for example. Parameters that are defined in the parent query are also available in subsequent queries.
Query module	Select the Single object history query module.
Object key	The object key can be queried directly or using a parameter. Add these parameters subsequently to the report by entering them on the Parameters tab. Columns in a parent query are formatted with the following syntax: <parent query name>.<parent query column>
Min date or range	Use the minimum date to specify the point in time that the history data should start from. You can define the date directly or using a parameter. In the case of a parameter, the minimum date of all affected entries in the connected One Identity Manager History Database databases is determined. Add these parameters subsequently to the report by entering them on the tab Parameters .
Columns	Columns for which the changes are determined.
Resolve foreign key	Set this option if the display value of the referenced object should be returned rather than the UID.

The data query returns the following columns.

Table 154: Columns from a data query using single object history

Column	Meaning
ChangeID	Unique identifier (UID) for the record.
ObjectKey	Object key or the record.
ObjectUID	Unique identifier (UID) for the modified objects.
User	Name of user that caused the change.
ChangeTime	Time of change
ChangeType	Type of change (Insert, Update, Delete)

Column	Meaning
Columnname	Name of column whose value has changed.
ColumnDisplay	Display name of column whose value has changed.
OldValue	Old column value.
OldValueDisplay	Old column display value.
NewValue	New column value.
NewValueDisplay	New value display value.

Related topics

- [Creating and editing data sources](#) on page 407
- [Report parameters](#) on page 418

Data retrieval using multiple object history

Use data queries with the **Multiple object history** query module to create reports about multiple objects with historical data that can be further restricted by a particular criterion. This could be all employees with a certain last name.

Table 155: Properties of data source multiple object history

Property	Meaning
Name	Name of the data source.
Description	Description of data source.
Max. lines	Maximum number of result lines for this query. NOTE: The report only displays results up to this maximum even if the number of results exceeds it. In the default, no error messages or tips are displayed. Any possible messages must be customized in the report.
Parent query	Not used.
Query module	Select the Multiple object history query module.
Table	Select the table to find the object in.
Minimum date or range	Use the minimum date to specify the point in time that the history data should start from. You can define the date directly or using a parameter. In the case of a parameter, the minimum date of all affected entries in the connected One Identity Manager History Database databases is determined. Add these parameters subsequently to the report by entering them

Property	Meaning
	on the Parameters tab.
Columns	Columns for which the changes are determined.
Criteria	Column, table, and value used for further narrowing down the objects found. The value can be queried directly or as a parameter. Add these parameters subsequently to the report by entering them on the tab Parameters .

The data query returns the following columns.

Table 156: Columns from a data query using single object history

Column	Meaning
ChangeID	Unique identifier (UID) for the record.
ObjectKey	Object key or the record.
ObjectUID	Unique identifier (UID) for the modified objects.
User	Name of user that caused the change.
ChangeTime	Time of change
ChangeType	Type of change (Insert, Update, Delete)
Columnname	Name of column whose value has changed.
ColumnDisplay	Display name of column whose value has changed.
OldValue	Old column value.
OldValueDisplay	Old column display value.
NewValue	New column value.
NewValueDisplay	New value display value.

Example:

A history of all employees with the last name "Harris" should be created. The report data can be defined in the following way:

Table:	Employee
Min Date	MinDate

Criteria: column	Lastname
Criteria: value	Harris

Related topics

- [Creating and editing data sources](#) on page 407
- [Report parameters](#) on page 418

Data retrieval using historical assignments

Use data queries with the **Historical assignments** query module to create reports with historical data from object assignments, for example, employee role memberships. This type is used for queries through foreign key relations as well as through assignment tables (many-to-many tables) and child relations.

Table 157: Properties of data source historical assignments

Property	Meaning
Name	Name of the data source.
Description	Description of data source.
Max. lines	Maximum number of result lines for this query. NOTE: The report only displays results up to this maximum even if the number of results exceeds it. In the default, no error messages or tips are displayed. Any possible messages must be customized in the report.
Parent query	In a parent query, restrictions are applied to the data record that are passed on to subsequent queries, all members of a department, for example. Parameters that are defined in the parent query are also available in subsequent queries.
Query module	Select the Historical assignments query module.
Assignment direction	Assignment to be used in the report. Permitted values are Assignments (CR & MN) and Referenced objects (FK) .
Table	Table for the assignment.
Minimum date or range	Use the minimum date to specify the point in time that the history data should start from. You can define the date directly or using a parameter. In the case of a parameter, the minimum date of all affected entries in the connected One Identity Manager History Database databases is determ-

Property	Meaning
	ined. Add these parameters subsequently to the report by entering them on the Parameters tab.
Criteria column	Column in the table for linking to the base object.
Criteria value	The value of the criteria column can be queried directly or using parameters. Add these parameters subsequently to the report by entering them on the Parameters tab. Columns in a parent query are formatted with the following syntax: <parent query name>.<parent query column>
Foreign key to query	Foreign key to retain historical assignments.
Disabling columns	Certain tables contain columns that can disable an object, for example, the AccountDisable column in the ADSAccount table. Enter these columns if an assignment should be labeled as "Deleted" when disabled and "Added" if enabled.
Additional object columns	Enter the columns from the table that should also be available in the report.
Additional criteria	Column of the table and value for further restriction of the base object.

The data query returns the following columns. In addition, columns are supplied that are edited like object columns.

Table 158: Columns from a data query using historical assignments

Column	Meaning
BaseKey	Object key for assignment base object.
BaseUID	Base object unique identifier.
ObjectKey	Assignment object key.
DestinationKey	Object key for assignment target object.
DestinationUID	Target object unique identifier.
Display	Target object display value.
CreationUser	User that created the assignment.
CreationTime	Time of assignment.
DeletionUser	User that deleted the assignment.

Column	Meaning
DeletionTime	Time of deletion.
Type	More detailed specification of the assignment, for example, assignment table name or target system type.
Origin	Bitmask for mapping the type of assignment.
OriginDisplay	Display name of the bitmask for mapping the type of assignment.

Related topics

- [Creating and editing data sources](#) on page 407
- [Report parameters](#) on page 418

Data query for simulation data

To select the simulation data generated during simulation in the Manager in a report, use the following query modules:

- Front-end simulation result **You can apply this query module to all parts of a simulation excluding rule violation analysis.**
- Front-end simulation result for compliance **You can apply this query module to publish the rule violation analysis in the report.**

Table 159: Data source front-end simulation result properties

Property	Meaning
Name	Name of the data source.
Description	Description of data source.
Query module	Select the Front-end simulation result query module.
Parent query	Not used.
Simulation analysis	<p>Defines which part of the simulation analysis is shown in the report.</p> <p>Permitted values are:</p> <ul style="list-style-type: none"> • Überblick: Shows which actions were triggered through changes made during the simulation in an overview. • Changed properties: Shows objects and their properties affected by the changes made during simulation. • DBQueue: Shows the calculation tasks for the DBQueue Processor resulting from changes made during simulation.

Property	Meaning
	<ul style="list-style-type: none"> • Trigger changes: Shows all changes made to objects during simulations due to triggering. • Generated processes: Shows processes and process steps generated during simulation due to the changes.

Table 160: Data source front-end simulation result for compliance properties

Property	Meaning
Name	Name of the data source.
Description	Description of data source.
Query module	Select the query module Frontend Simulation Result for Compliance.
Parent query	Not used.

Related topics

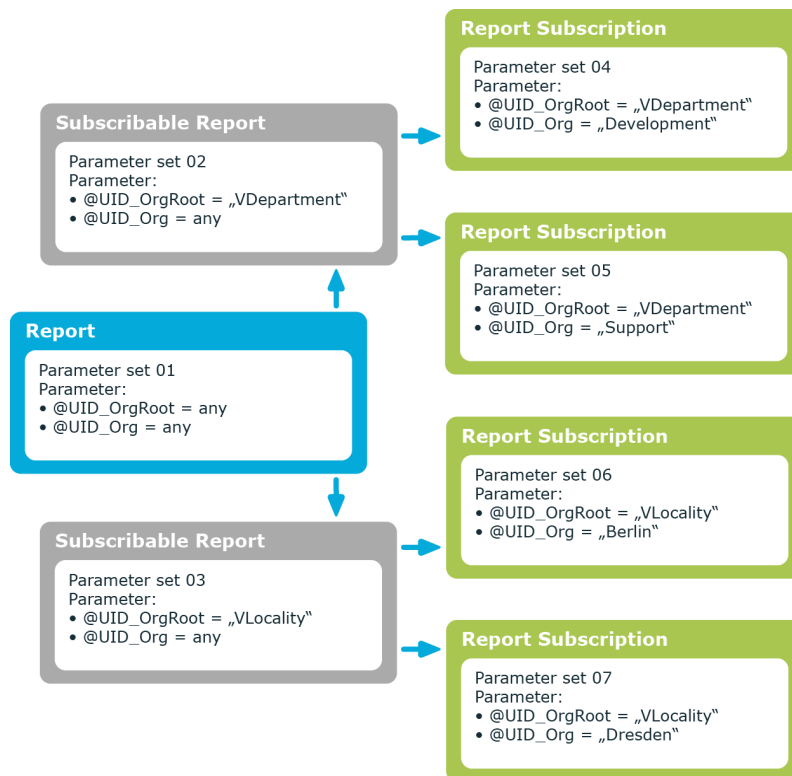
- [Creating and editing data sources](#) on page 407

Report parameters

A report can contain several parameters that are determined when the report is created or when an email notification is generated and passed to the report. The generated report is then displayed or send by email to the subscriber corresponding to the report subscription set up. The user can query the report parameters before the report is displayed. This means, you can, for example, limit the time period or pass specific departments for displaying the report.

Report parameters are grouped internally into parameter sets. A separate parameter set is automatically created for very report, every subscribable report, and every report subscription. The parameters and their settings are passed down in the sequence *report->subscribable report->report subscriptions*.

Figure 33: Report parameter inheritance



You can configure report parameters at several places.

Parameters for reports

Define the report parameters to use when you create the report in the Report Editor. This is where you specify which report parameters are viewable or writable and which are already predefined in a subscribable report.

Parameters for subscribable reports

When you add a subscribable report viewable parameters are displayed in the Manager. You can make further changes to these report parameters assuming they can be overwritten. That means, you specify which report parameters can be viewed or overwritten by Web Portal users and define parameter values.

Parameters for report subscriptions

Report parameters labeled as viewable and editable in subscribable reports, are shown to Web Portal users when they are setting up their personal report subscriptions. If the report parameters are editable, Web Portal users can modify the values in them.

NOTE: In the report, you must define all the report parameters that users can apply. For example, when the report is displayed, when subscribable reports are generated in the Manager, or in Web Portal report subscriptions.

Detailed information about this topic

- [Editing report parameters](#) on page 420
- [Editing general parameter settings](#) on page 421
- [Editing parameter value definitions](#) on page 422
- [Settings for calculating values](#) on page 424

Editing report parameters

To edit report parameters

1. Open the report in the Report Editor.
2. Select the **Parameters** tab in the properties dialog box.
3. Select the report parameter from **Defined queries**.
- OR -
Click **Add**.
Creates a new report parameter.
4. Edit the report parameter properties.
5. Save the changes.

To delete a report parameter

1. Open the report in the Report Editor.
2. Select the **Parameters** tab in the properties dialog box.
3. Select the report parameter from **Defined queries**.
4. Click **Delete**.
5. Save the changes.

Related topics



- [Report parameters](#) on page 418
- [Editing general parameter settings](#) on page 421
- [Editing parameter value definitions](#) on page 422
- [Settings for calculating values](#) on page 424

Editing general parameter settings

To edit general parameter settings

1. Open the report in the Report Editor.
2. Select the **Parameters** tab in the properties dialog box.
3. In the **Defined parameters** list, select the report parameter and then the **General** tab.

Table 161: General parameter settings

Property	Description
Parameter name	Parameter name. NOTE: The name must agree with the name of the parameter in the data query.
Parameter type	Type of parameter. Permitted values are: <ul style="list-style-type: none">• Fixed: Fixed parameter values are used. On the Value definition tab, enter the parameter value.• User prompt: The user must select a parameter value in the user prompt at runtime.• Calculation: The parameter value is calculated at runtime when the report is created. On the Value calculation tab, enter the table column and the condition for calculating the value. Other settings are shown or hidden depending on the type.
Display name	User friendly name for the parameter. To display language dependent display names, translate the given text with the  button.
Description	Text field for additional explanation. Translate the given text with the  button.
Sort order	Position of the parameter in the subscribable report view and in the Web Portal.
Mandatory parameter	Specifies whether this is a mandatory parameter. You must give a value for a mandatory parameter.
Viewable	Specifies whether the parameter is shown when a report subscription is set up in the Web Portal.
Can be overwritten	Specifies whether the parameter can be overwritten by Web Portal users when a report subscription is created.

Related topics

- [Editing report parameters](#) on page 420
- [Editing parameter value definitions](#) on page 422
- [Settings for calculating values](#) on page 424

Editing parameter value definitions

Specify the parameter value and define the parameter value characteristics. Other input is shown or hidden depending on the parameter definition values.


To edit parameter definitions

1. Open the report in the Report Editor.
2. Select the **Parameters** tab in the properties dialog box.
3. In the **Defined parameters** list, select the report parameter and then the **Value definition** tab.

NOTE: The **Parameter value** and **Default value** are affected by the parameter value definition. On the one hand, you can see this through dynamic customization of the controls for selecting a parameter value, or on the other hand, through the default value and the dynamic customization of the selectable values themselves. It is therefore recommended that you edit these values last.

Table 162: Value definition

Property	Description
Data type	Parameter data type.
Date add-on	Additional information about calculating date and time data for displaying in the user interface. The value can be edited if the Date data type is selected.
Value range	Specifies whether the report parameter value has to be within a given range. If Yes , additional fields appear.
Multivalue	Specifies whether the parameter accepts multiple values. If Yes , users can select multiple value from a list.
Multiline	Specifies whether the parameter contents can have multiple lines. If Yes , line breaks are permitted.
Data source	Type of data source. Permitted values are: <ul style="list-style-type: none">• None: The user can give any value.• Table: The user selects a value from a specified table column.• List of permitted values: The user selects a value from a predefined list.

Property	Description
	You may require additional data depending on the data source.
Table column (query)	<p>Additional data for the data source Table.</p> <p>Table column for selecting the parameter value. The user can select a value from this table column. If the parameter is multi-value, you can select several values from this column as well.</p>
Display pattern	<p>Additional data for the data source Table.</p> <p>Display pattern for table elements in lists in %column% notation. The ?? operator is permitted. This means, when one column's value is empty, another column's value is displayed.</p> <p>Example: %column1??column2??column3%</p>
Condition (query)	<p>Additional data for the data source Table.</p> <p>Limiting condition (Where clause) for selecting the parameter value using a table column. The user can select a value from the result set. If the parameter is multi-value, you can select several values from this result set as well.</p> <p>You can reference other parameters in the condition using the following syntax:</p> <p><code>\$PC(<Parametername>)\$</code></p> <p>Example:</p> <p><code>UID_Database = \$PC(UID)\$</code></p> <p>where UID is the name of the referenced report parameter.</p>
List of permitted values	<p>Additional data for the data source List of permitted values.</p> <p>List of values permitted in this parameter in the value=display name notation. If an = is no given, the entry counts as both value and display name.</p> <p>Example: 1=internal 2=external</p> <p>To display language dependent display names, translate each display name using the  button.</p>
Overwrite empty value	<p>Specifies whether an empty parameter value overwrites the default value.</p> <p>If this option is disabled, the default value is overwritten if a parameter value is not given.</p>
Example value	<p>Example of the parameter. The example value is used to create a report preview.</p> <p>If a value range is given, the Example value (from) and the Example value (to) are displayed.</p>
Default	Default value of the parameter. This is used if the user does not specify a

Property	Description
value	parameter value and the Overwrite empty value option is not set. If a value range is given, the Default value (from) and the Default value (to) are displayed.

Related topics

- [Editing report parameters](#) on page 420
- [Editing general parameter settings](#) on page 421
- [Settings for calculating values](#) on page 424
- [Display template for displaying a list](#) on page 132

Settings for calculating values

To edit settings for value calculation

1. In the Report Editor, open the report .
2. Select the **Parameters** tab in the properties dialog.
3. In the **Defined parameters** list, select the report parameter and then the **Value calculation** tab.

Table 163: Scripts for calculating values

Property	Description
Table column (calc.)	Additional input for Calculated parameter type. Table column for selecting the parameter value. The parameter value is determined at runtime when the report is created.
Condition (calc.)	Additional input for Calculated parameter type. Limiting condition (where clause) for selecting the value through a table column. The parameter value is determined at runtime when the report is created. If the parameter is multivalue as well, several values may be found. If a condition is not given and the parameter is not multivalue, the first value is used that is determined by the table column. If the parameter is multivalue and a condition is not given, all determined values are used.
Valuation script	Script in VB.Net syntax for modifying the parameter value. The script can be used as a formatting script and the existing parameter value modified or reset the parameter value.
Validation script	Script in VB.Net syntax for checking permitted values of parameters. Create a script that checks the user input.

Related topics

- [Editing report parameters](#) on page 420
- [Editing general parameter settings](#) on page 421
- [Editing parameter value definitions](#) on page 422

Using virtual data sources

You can use virtual data sources when you want to use a data source more than once within a report, but with other limitations or sorted differently.

To create a virtual data source

1. In the Report Editor, open the report.
2. Select the **Report > New virtual data source** menu item.
Opens a dialog window showing all existing data sources for the report.
3. Configure the properties for the virtual data source.

Related topics

- [Creating and editing data sources](#) on page 407


Editing the report form

You can create and edit reports in the edit view of the Report Editor. The Stimulsoft Reports.Ultimate Report Designer is integrated into the edit view. You can find accurate descriptions and the functionality of individual components in the Stimulsoft online help (www.stimulsoft.com).

NOTE: When you start the Report Editor for the first time, you can select the configuration type (**basic**, **default** or **professional**) for the report. The configuration type determines the range of properties displayed when editing a report. You can change the configuration type later in the edit view using the context menu in the property view.

The following functions are appended to the Stimulsoft Reports.Ultimate Report Designer toolbar:

Table 164: Extensions to Stimulsoft Reports.Ultimate Report Designer toolbar

Icon	Meaning
	Imports a reports (XML format).




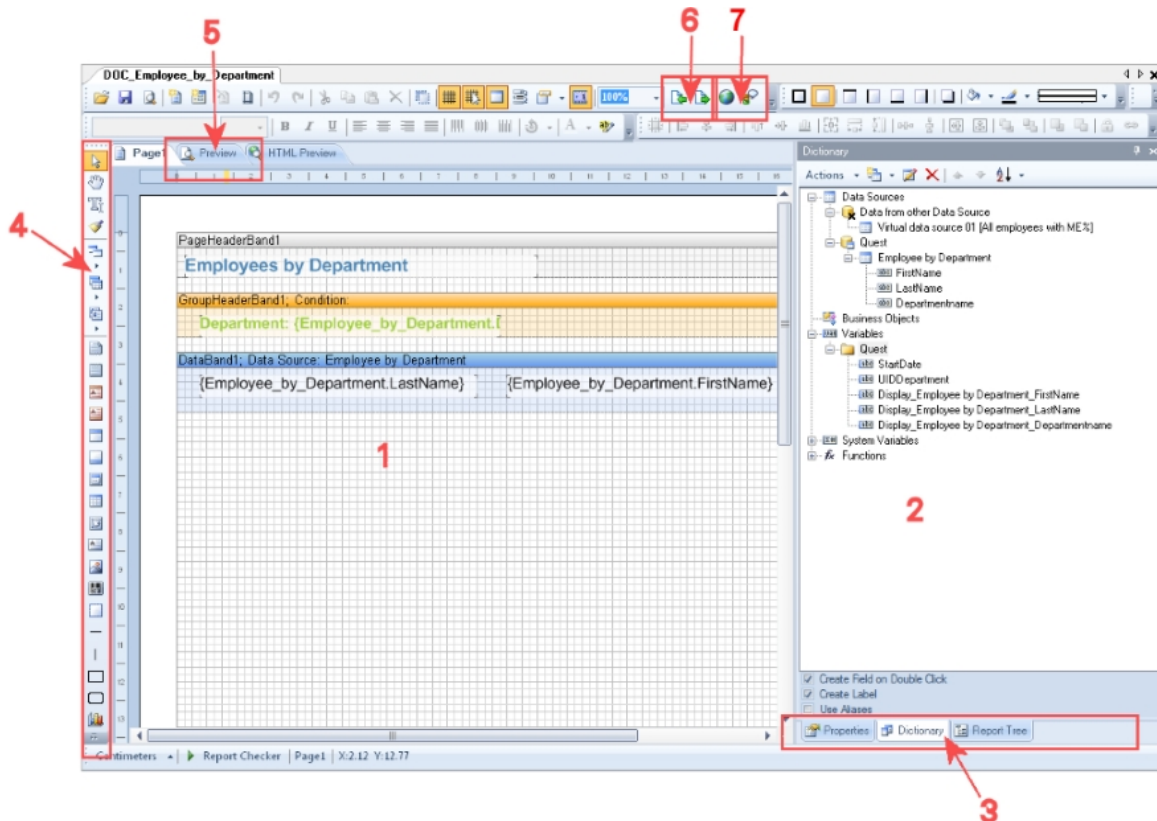
Icon	Meaning
	Export a report (XML format).
	Globalization editor. Opens the Report Designer globalization editor.
	Opens the Translate texts dialog.

Figure 34: Report Designer with report form (1), dictionary/properties view (2), tabs for swapping between dictionary/properties (3), toolbox (4), preview (5), import/export of report pages (6), translate report (7)



Detailed information about this topic

- [Adding data fields to report forms](#) on page 426
- [Translating reports](#) on page 432
- [Example of a simple report with data grouping](#) on page 428

Adding data fields to report forms

Add the control elements for the data you want to appear in the report on the report form and link them to the data source columns. After you have created the data sources,

they are listed with all the columns used in the Report Designer's dictionary under **Quest**. The report parameters are available as variables under **Quest**.

You can find accurate descriptions and the functionality of individual components in the Stimulsoft online help (www.stimulsoft.com).

To insert data boxes into the report form

1. Select the column you want to add to the report in the **Dictionary** tab.
2. Position the column on the report form using "drag and drop".
This creates a new control element on the report form which includes some predefined variables.
TIP: You can add other control elements as necessary with the Report Designer tool palette.
3. The Report Designer properties window (**Properties**) allows you to customize individual control elements.
4. Use **Preview** to view the report during editing. The preview takes some sample parameter values to determine the data for the preview display.

Related topics

- [Example of a simple report with data grouping](#) on page 428

Tips for entering dates in reports

If no date is given, the date 12/30/1899 is used internally. Take this into account when values are compared, for example, when used in reports.

To display a string other than the internal date **12/30/1899**, you have the following options:

- For data sources in the **SQL** query module, change the date conversion in the data source.


Example:

```
select ISNULL(convert(varchar, Person.ExitDate, 121), '-') as date_
substituted
```

- Change the expression for displaying the date columns when you create the report form.

Example:

```
{IIF(Person.ExitDate.ToString() = "12/30/1899 12:00:00 AM", "-
", Person.ExitDate)}
```

- Define a condition for displaying the date columns when you create the report form.
In the Report Designer, add conditions to toolbar using the  icon.

Example of a simple report with data grouping

We want to create a report that lists all employees as grouped in their respective departments.

1. A new report is created.

- The report is given the name **CCC_Employee_by_Department**. The display name defined is **Employees by Department %UID%**.

- **A data source (Employee by Department) is created for the report with the SQL**

query module. The data query should return the employees assigned to a department. The department is found with the object key (XObjectKey). This is passed as a parameter to the report. The employee's first name (firstname), last name (lastname) and department name (departmentname) are queried.

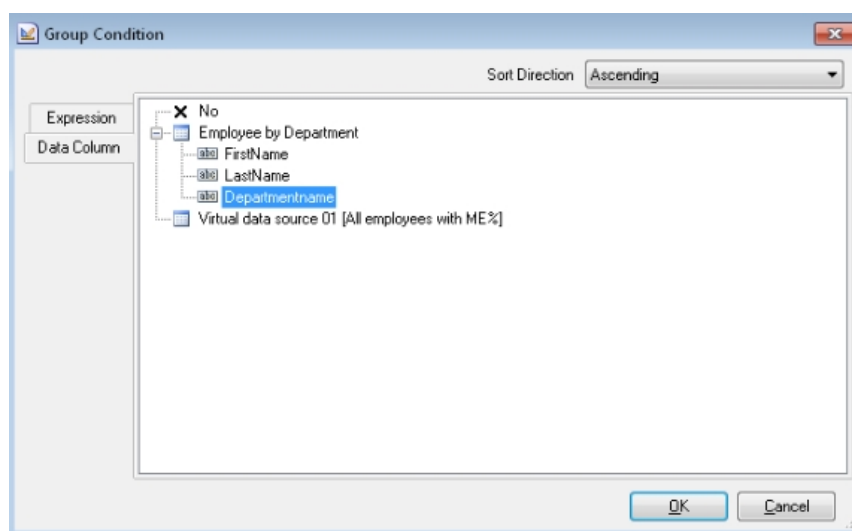
```
Select Firstname, Lastname, Departmentname
      from person join Department
      on person.uid_Department = department.uid_Department
      where Department.XObjectKey = @UIDDepartment
```

- This adds the UIDDepartment parameter to the report. It is populated with a sample value for the preview.

2. The control elements for the database columns are arranged on the report form.

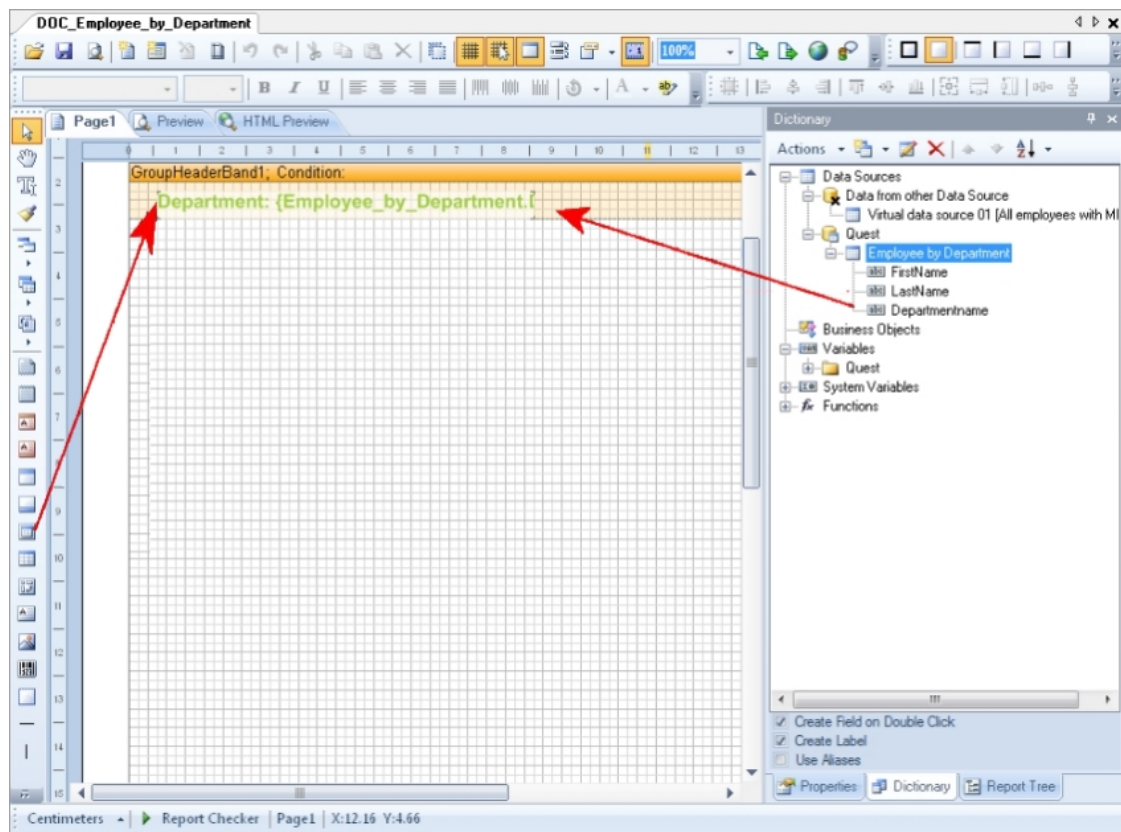
For data grouping, add a band of the **Group header** type from the Report Designer's toolbox to the report form. The column name used for grouping must be entered as a grouping condition. In the example, this is Departmentname.

Figure 35: Specifying the grouping condition



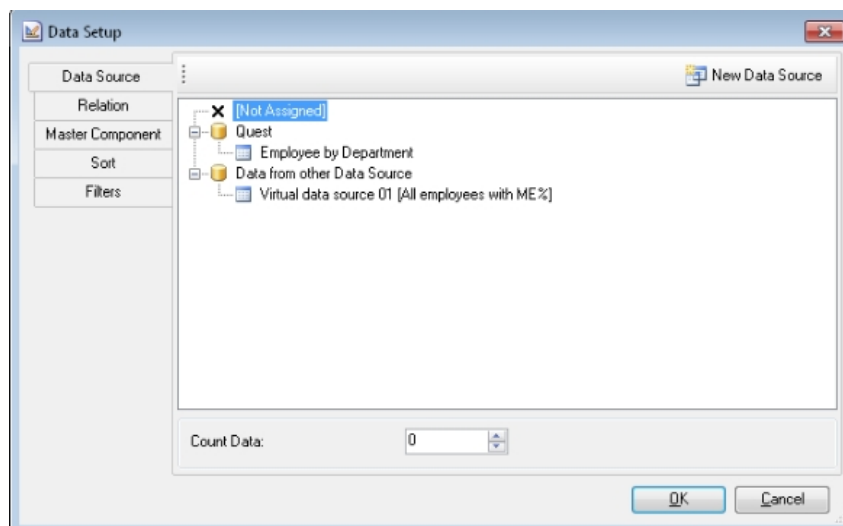
3. Drag and drop the Departmentname column from the Report Designer's dictionary (**Dictionary** tab) into the group header. This creates a new control element on the report form.

Figure 36: Creating a group



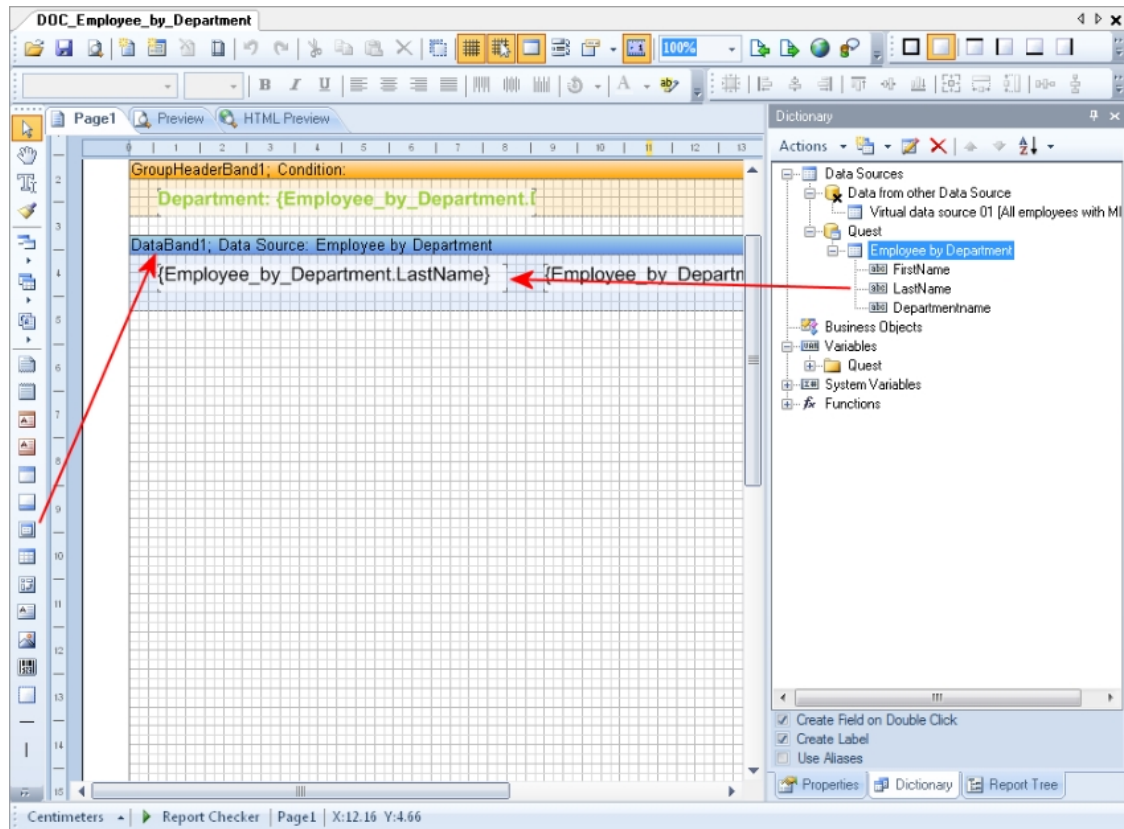
4. To display employees, add a **Data band** to the report form from the Report Designer's toolbox. Specify the data source as **Employee by Department**.

Figure 37: Specify the data source



5. Drag and drop the Lastname and Firstname columns from the Report Designer's dictionary (**Dictionary** tab) to the data band. This creates the respective control elements on the report form.

Figure 38: Organizing control elements on a report form




6. Other control elements such as a title (**PageHeader**) can be added as necessary with the Report Designer. The Report Designer **Properties** window allows you to customize individual control elements.
7. The preview can be used to view the report during setup. The preview uses the sample parameter values in the parameter view of the report edit dialog to determine the data for this.

Detailed information about this topic


- [Editing general report properties](#) on page 406
- [Creating and editing data sources](#) on page 407
- [Tips for entering dates in reports](#) on page 427
- [Report parameters](#) on page 418
- [Editing the report form](#) on page 425

Translating reports

A report can contain several elements that require translating in order to display the report in more than one language.


- Database columns used in the report definition.
Translate database columns with the Language Editor in the Designer.
- Display name/ReportAlias.
The report's display name used when a report is created as **ReportAlias**. The display name is entered in the report properties dialog. Translate the given text using the  button.
- Text elements on the report form.
Translate the text elements directly in the Report Editor with the Globalization Editor.

To translate all text elements in a report

1. In the report, select the report list and open it with double-click or with **Edit** from the context menu.
This opens the report form in the Report Designer.
2. Start the Globalization Editor.
 - Click on the  button in the Report Designer toolbar.
 - OR -
 - In the Report Designer's properties view, select the report from the menu on the **Properties** tab and use **Globalization Strings** to open the Globalization Editor.

NOTE: You can only start the Globalization Editor from the Report Designer's properties view if you have selected **Professional**. You can change the configuration type later in the edit view using the context menu in the property view.
3. Ensure that the **Auto Localize Report on Run** button is set.
This means the report is generated in the current language.
4. Enter a culture for the language using **Add Culture** and translate each entry.

To translate single captions

1. Select the report in the report list and open it with double-click or with **Edit** from the context menu.
This opens the report form in the Report Designer.
2. Select the caption on the report form.
3. Open the dialog box using the  in the Report Designer toolbar.
4. Translate the text and confirm the changes with **OK**.

Related topics

- [Displaying translations in the Language Editor](#) on page 214
- [Editing general report properties](#) on page 406

Embedding reports in the user interface

In order to display a report in a One Identity Manager administration tool, such as Manager, you need to link in the report as a custom interface form.

In the Manager's info system, you can display reports that you create in the Report Editor as statistics. To do this, you must alter the **Manager's** user interface. The report opens when you double-click on the statistic's header.

To create a user interface form

1. In the Designer, select the **User interface > Forms > User interface forms** category.
2. Select the **Edit form** task.
3. Select the **Form > Insert** menu item.
4. Edit the interface form's main data.

Take the following cases into account:

- Use the **VI_Report** form definition.

This form definition is configured for displaying in the graphical user interface and in web applications. You only need to set up one interface form for this. Which form template will be used to display the interface form is decided dynamically, depending on usage.

- In the form's configuration data, enter the name of the report to be run and the report parameters in the Properties section.

Syntax:

```
<DialogSheetDefinition FormatVersion="1.0">
  <Properties>
    <Property Name="ReportName">ReportName from the
      DialogReport</Property> table
    <Property Name="ReportParameter.Parameter1">value1</Property>
    <Property Name="ReportParameter.Parameter1">value</Property>
  </Properties>
</DialogSheetDefinition>
```

Example:

```
<DialogSheetDefinition FormatVersion="1.0">
```

```

    <Properties>
      <Property Name="ReportName">CCC_Employee_by_
      Department</Property>
      <Property Name="ReportParameter.UIDDepartment">%UID_
      Department%</Property>
    </Properties>
  </DialogSheetDefinition>

```

5. Assign the user interface form to the applications and permissions groups.
6. (Optional) Assign the user interface form to the object definitions.
7. (Optional) Assign the user interface form to the menu items.

Related topics

- [Editing user interface forms on page 140](#)
- [Creating user interface forms](#)
- [Assigning user interface forms to applications](#)
- [Assigning user interface forms to permissions groups](#)
- [Assigning user interface forms to menu items on page 146](#)
- [Assigning user interface forms to object definitions on page 144](#)
- [Using reports in statistics on page 177](#)

Generating and exporting reports on a cyclical basis

You can create customer-specific processes to control the creation of reports and perform the export on a cyclical or event-controlled basis.

You can use the ReportComponent process component to create reports and export them to different file formats. The following formats are supported: HTML, PDF, RTF, TEXT, XLS, TIFF, XML, CSV, XPS, DOCX, and XLSX.

To export reports in CSV format, you can also use the ScriptComponent process component with the CSVExport or CSVExportSingle process tasks.

NOTE: Use the default report server as executing server in the processes.

Related topics

- [Defining processes on page 224](#)

Adding custom tables or columns to the One Identity Manager schema

The object technology implemented in One Identity Manager makes it possible to add customer-specific columns and tables to the existing application data model at the database level. These are, therefore, available at the object level with all corresponding tasks. A custom extension to the system data model is not recommended.

Basic knowledge of the database system's SQL Server in use is a prerequisite for making schema extensions. It is assumed that you understand the concept and the architecture of One Identity Manager.

To implement a custom extension of the One Identity Manager schema, use the Schema Extension program. You can make the following extensions using the Schema Extension:

- Create new tables
- Create new assignment tables
- Create new columns
- Create new views
- Create new indexes
- Removing custom schema extensions

NOTE: You can delete custom schema extensions only on databases with the **Test environment** or **Development system** staging level. It is not possible to transport the changes.

The Schema Extension program creates the schema extensions in the database and ensures that the necessary extensions are made in the One Identity Manager schema. The basic table definitions and column definitions of the custom tables are entered in the DialogTable, DialogColumn, QBMRelation and DialogValidDynamicRef tables. You must then adjust the properties in the Designer to the desired requirements.

The Designer contains a variety of consistency checks. Run these consistency checks and apply the repair methods after carrying out a schema extension and after making changes to table and column definitions. For more information about checking data consistency, see the *One Identity Manager Operational Guide*.

You cannot create custom functions, triggers, or database procedures with the Schema Extension program. If you need custom functions, triggers, or database procedures, add

these to the database in a suitable program for running SQL queries. Keep to the following conventions for name database components.

- Name begin with the **CCC_** string.
- All names are a maximum of 30 characters long.
- One Identity recommends using UpperCamelCase as notation for the names.

Detailed information about this topic

- [Creating new tables](#) on page 436
- [Creating new assignment tables](#) on page 448
- [Extending tables](#) on page 438
- [Creating database views with read-only type](#) on page 445
- [Creating database views with Union type](#) on page 447
- [Creating indexes](#) on page 450
- [Removing custom schema extensions](#) on page 450
- [Permissions for schema extensions](#) on page 452
- [Change labels for the schema extensions](#) on page 452
- [Adding schema extensions to the database](#) on page 453

Related topics

- [One Identity Manager schema basics](#) on page 44
- [Table types and default columns in the One Identity Manager data model](#) on page 49
- [Recommendations for advanced configuration of custom schema extensions](#) on page 454
- [Managing custom database objects within the database](#) on page 456

Creating new tables

Use this task to create a simple table in the One Identity Manager schema.

Technical details

- The technical identifier for the table is automatically formed according to the CCC<Table name> schema.
- The following columns are generated automatically:

- Primary key column
The primary key column is automatically transferred as the UID. The name of the primary key column is formed according to the UID_CCC<table name>.
- X columns (XUserInserted, XUserUpdated, XDateInserted, XDateUpdated, XTouched, XObjectKey, XMarkedForDeletion)

To create a simple table in the Schema Extension

1. Start the Launchpad and log in to the One Identity Manager database.
2. Open the Launchpad and select the **One Identity Manager Schema Extension** entry. This starts the Schema Extension program.
3. Click **Next** on the start page.
4. On the **Database connection** page, check the connection data for the One Identity Manager database.
5. On the **Select method** page, select **New table**.
6. On the **Create new table** page, enter the following information.

Table 165: Table properties

Property	Description
Table	A technical name for the table.
Display name	Displays table name The display name is used, for example, to identify the table in a database search or for error output.
Description	Comments on using the table.

7. On the **Configure columns of table** page, create the new columns. For more information, see [Defining columns](#) on page 438.

Related topics

- [Table types and default columns in the One Identity Manager data model](#) on page 49
- [Extending tables](#) on page 438
- [Creating new assignment tables](#) on page 448
- [Creating database views with read-only type](#) on page 445
- [Creating database views with Union type](#) on page 447

Extending tables

To extend an existing table in the Schema Extension

1. Start the Launchpad and log in to the One Identity Manager database.
2. Open the Launchpad and select the **One Identity Manager Schema Extension** entry. This starts the Schema Extension program.
3. Click **Next** on the start page.
4. On the **Database connection** page, check the connection data for the One Identity Manager database.
5. On the **Select method** page, select **Extend table**.
6. On the **Extend table** page, select the table that you want to extend from the **Table** menu.
7. On the **Configure columns of table** page, create the new columns. For more information, see [Defining columns](#) on page 438.

Related topics

- [Table types and default columns in the One Identity Manager data model](#) on page 49
- [Creating new tables](#) on page 436
- [Creating new assignment tables](#)
- [Creating database views with read-only type](#) on page 445
- [Creating database views with Union type](#) on page 447

Defining columns

On the **Configure columns of table** page in the Schema Extension, you can see which columns already exist for the selected table and how many resources are free for new columns.

| **NOTE:** Take the maximum size allowed for a table into account when extending.

Detailed information about this topic

- [Creating simple columns](#) on page 439
- [Creating foreign key columns](#) on page 440
- [Creating dynamic foreign keys](#) on page 441
- [Creating new columns for database views with type view](#) on page 442
- [Advanced configuration of columns](#) on page 443

Related topics

- [Table types and default columns in the One Identity Manager data model](#) on page 49

Creating simple columns

Technical details

- The technical identifier for the column is automatically formed according to the CCC_<column name> schema.

To create a simple column in the Schema Extension



1. On the **Configure columns of table** page, click .
2. Select the **Simple column** option and enter the name of the column under **Column name**.
3. Click **OK**.
4. On the **Configure columns of table** page, enter at least the following information.

Table 166: General column properties

Property	Description
Required field	Specifies whether a column must be filled.
Data type	Column data type Permitted .Net data types are listed in a pop-up menu. These are represented internally as SQL data types.
Length	Column length The column length is only specified for the .Net String data type.
Display name	Specifies how the column is labeled.

5. (Optional) Click  to configure more column properties. For more information, see [Advanced configuration of columns](#) on page 443.

Related topics

- [Table types and default columns in the One Identity Manager data model](#) on page 49
- [Creating new columns for database views with type view](#) on page 442
- [Creating foreign key columns](#) on page 440
- [Creating dynamic foreign keys](#) on page 441

Creating foreign key columns

Restrictions

- The referenced table has a one-column primary key.

Technical details

- The technical identifier for the column is automatically formed according to the CCC_<column name> schema.
- Foreign key columns are created with the String data type and a length of 38 characters.
- The table relations and column relations are generated automatically.
 - The relation IDs follow the naming convention:
CCC-<database ID>-<4 digit sequential number>
 - If a foreign key column is added to a database view, the relation IDs follow the naming convention:
CCC-<database ID>-<4 digit sequential number> <Name of referenced table>
 - If a column from a base table is referenced using the **Base table** table type, the table relations and column relations are also created for the base table.
 - Validation of referential integrity is done by DLL or triggered.

To create a foreign key column in the Schema Extension program



1. On the **Configure columns of table** page, click .
2. Select **Foreign key column** and enter the following data.
 - **Column name:** Enter the name of the column. If possible, the name of the foreign key column should correspond to the name of the referenced table's primary key.
 - **From table:** Select the referenced table.
3. Click **OK**.
4. On the **Configure columns of table** page, enter at least the following information.
 - **Required field:** Specifies whether a column must be filled.
 - **Display name:** Specifies how the column is labeled.
5. Click  and enter the following information in the **Key column values** tab.

Table 167: Properties for foreign key columns

Property	Remarks
Foreign key	Specifies whether the column is a foreign key column. Enable the option.
From table	Referenced table for foreign key relations
Delete restrictions	Restriction for testing referential integrity when deleting an object
Insert restrictions	Restriction for testing referential integrity when inserting an object

6. (Optional) You can also configure more column properties. For more information, see [Advanced configuration of columns](#) on page 443.

Related topics


- [Table types and default columns in the One Identity Manager data model](#) on page 49
- [Table relations](#) on page 99
- [Creating simple columns](#) on page 439
- [Creating new columns for database views with type view](#) on page 442
- [Creating dynamic foreign keys](#) on page 441

Creating dynamic foreign keys

Technical details

- The technical identifier for the column is automatically formed according to the CCC_<column name> schema.
- Dynamic foreign key columns are created with the String data type and a length of 138 characters.

To create a dynamic foreign key column in the Schema Extension

1. On the **Configure columns of table** page, click .
2. Select **Dynamic foreign key columns** and enter the name of the column under **Column name**.
3. Click **OK**.
4. On the **Configure columns** page, enter at least the following information.
 - **Required field:** Specifies whether a column must be filled.

- **Display name:** Specifies how the column is labeled.

5. Click  and enter the following information in the **Dynamic foreign key** tab.

Table 168: Properties for foreign key columns

Property	Remarks
Referenced table	Select the tables to be referenced. All tables are permitted, if there are no restrictions.
Delete restrictions	Restriction for testing referential integrity when deleting an object
Insert restrictions	Restriction for testing referential integrity when inserting an object

1. (Optional) You can also configure more column properties. For more information, see [Advanced configuration of columns](#) on page 443.

Related topics


- [Table types and default columns in the One Identity Manager data model](#) on page 49
- [Dynamic foreign key](#) on page 102
- [Creating simple columns](#) on page 439
- [Creating new columns for database views with type view](#) on page 442
- [Creating foreign key columns](#) on page 440


Creating new columns for database views with type view

If the table to be expanded is a database view with the **View** table type, the selection of new columns is restricted to custom columns of the underlying base table. You can only select custom columns from the base table that are not yet used in the view.

- First extend the base table by a new column (simple column or foreign key column).
- Then you extend the database view with the new columns.

To create a new column for database views with the View type in the Schema Extension

1. On the **Configure columns of table** page, click .
2. Enter the following information.

- **Column name:** Enter the name of the column.
 - **Base column:** Select the base table column that you want to add to the database view.
3. Click **OK**.
 4. (Optional) Click  to configure more column properties. For more information, see [Advanced configuration of columns](#) on page 443.

Related topics

- [Creating simple columns](#) on page 439
- [Database views of the View type](#) on page 55

Advanced configuration of columns

To edit column properties


1. Select the column on the **Configure columns of table** page and click the  button.
2. Configure the column properties.
3. Enter the following information on the **Advanced** tab.

Table 169: Advanced configuration of columns

Property	Remarks
Name	Technical identifier for the column The column name is formed from CCC_<column name.
Data type	Column data type Permitted .Net data types are listed in a pop-up menu. These are represented internally as SQL data types. The only permitted data types are those already used in the One Identity Manager data model.
Length	Column length The column length is only specified for the .Net String data type. For columns containing UIDs, enter the value 38.
Column contains UIDs	Specifies whether this is UID column. This option is only permissible for columns with the String .Net data type and a length of 38 characters.
Column contains unicode	Specifies whether the column contains Unicode. This option is only permissible for String .Net data types.

4. Enter the following information on the **Base values** tab.

Table 170: Column base values

Property	Remarks
Primary key	Specifies whether the column is used as a primary key.
Required field	Specifies whether a column must be filled.
Display name	Specifies how the column is labeled.
Show in wizards	Specifies whether the column is offered in the Rule Editor for compliance rules to create queries and in the Web Portal to display tabular overviews.
Do not auto extend permissions	For custom columns in a predefined table, permissions are not automatically assigned to predefined permissions groups, even though the Common AutoExtendPermissions configuration parameter is set.
Comment	Additional information about the column.
Initial value	Initial value for the column This value is transferred to the existing data records of the extended table. The initial value for numerical data types is 0 . The initial value for the Bool data type is False .
Sort order	The sort order specifies the position for displaying the column on the generic form and the custom tabs of the default form. Columns with a value less than 1 are not displayed on the forms.

5. Enter the following information for foreign key columns on the **Key column values** tab.

Table 171: Properties for foreign key columns

Property	Remarks
Foreign key	Specifies whether the column is a foreign key column.
From table	Referenced table for foreign key relations
Delete restrictions	Restriction for testing referential integrity when deleting an object
Insert restrictions	Restriction for testing referential integrity when inserting an object

6. Enter the following information for dynamic foreign key columns on the **Dynamic foreign key** tab.

Table 172: Properties for foreign key columns

Property	Remarks
Referenced table	Select the tables to be referenced. All tables are permitted, if there are no restrictions.
Delete restrictions	Restriction for testing referential integrity when deleting an object
Insert restrictions	Restriction for testing referential integrity when inserting an object

7. Click **OK**.

Related topics

- [Creating simple columns](#) on page 439
- [Creating foreign key columns](#) on page 440
- [Creating dynamic foreign keys](#) on page 441
- [Creating new columns for database views with type view](#) on page 442
- [Table relations](#) on page 99
- [Dynamic foreign key](#) on page 102

Creating database views with read-only type

Technical details



- The technical identifier for the database view is automatically formed according to the CCC<Table name> schema.
- The first column of the database query (view definition) is used as the primary key column of the database view.
NOTE: It is recommended that you reference the primary key column of the queried table in the view definition as the first column. If this is not possible, then at least select a unique characteristic.
- If a database view contains a foreign key column, you specify which destinations tables should be referenced. The table and column relations are generated automatically.
- If you want to index the database view for the full-text search, the XDateInserted, XDateUpdated, and XObjectKey columns must be available.

To create a database view of read-only type in the Schema Extension

1. Start the Launchpad and log in to the One Identity Manager database.
2. Open the Launchpad and select the **One Identity Manager Schema Extension** entry. This starts the Schema Extension program.
3. Click **Next** on the start page.
4. On the **Database connection** page, check the connection data for the One Identity Manager database.
5. On the **Select method** page, select **Create view**.
6. On the **Create view** page, enter the following information.

Table 173: Database view properties

Property	Description
Table	A technical name for the database view.
Display name	Name displayed for the database view. The display name is used, for example, to identify the database view in a database search or error output.
Description	Comments on using the database view.
View definition	Enter the database query as a Select statement.

7. (Optional) Create the foreign key relations on the **Create FK relations for views** page.
 - Double-click the  icon in front of the column name and select the target table in the **Target table** column.
8. On the **Configure columns of table** page, enter display names for the columns.
9. (Optional) Click  to configure more column properties. For more information, see [Advanced configuration of columns](#) on page 443.

Related topics

- [Database views of the Read-only type](#) on page 61
- [Using Common Table Expressions in read-only database views](#) on page 447
- [Creating database views with Union type](#) on page 447
- [Creating new columns for database views with type view](#) on page 442

Using Common Table Expressions in read-only database views

In One Identity Manager 7.0, the behavior was changed for Common Table Expressions (CTEs) with the `with` keyword as condition for view definitions in **Read-only** database views. Conditions for view definitions are embedded in a summarized query. This means, there is no guarantee that CTEs are placed right at the top of the query.

Possible error messages

```
(execute slot single)50000 0 re-throw in Procedure QBM_ZViewBuildR, Line 1050000  
0 rethrow in Procedure QBM_PViewBuildR_intern, Line 10250000 0 re-throw in  
Procedure QBM_PViewBuildR_intern, Line 8250000 0 re-throw in Procedure QBM_  
PViewBuild_FromAddOn, Line 6550000 0 re-throw in Procedure QBM_PSQLCreate, Line  
26156 0 detected in (...) Procedure ..., Line 6156 0 Incorrect syntax near the  
keyword 'with'
```

Recommended action

1. Create a database view using the CTE.

Example:

```
create view CCC_Vxy as  
with myWithClause (column1, column2) as (  
select 1 as column1, 2 as column2  
)  
select * from myWithClause  
go
```

2. Use the database view in the additional view definition (QBMViewAddOn) of **Read-only** database views.

```
select * from CCC_Vxy
```

Creating database views with Union type

Technical details

- The technical identifier for the database view is automatically formed according to the schema `CCC<Table name>`.

- As the first column of the database query (view definition), the object key (XObjectKey) must be referenced. The object key allows faster access to a single object with its valid permissions.
- If you want to index the database view for the full-text search, the XDateInserted, XDateUpdated, and XObjectKey columns must be available.

To create a database view of Union type in the Schema Extension

1. Start the Launchpad and log in to the One Identity Manager database.
2. Open the Launchpad and select the **One Identity Manager Schema Extension** entry. This starts the Schema Extension program.
3. Click **Next** on the start page.
4. On the **Database connection** page, check the connection data for the One Identity Manager database.
5. On the **Select method** page, select **Create union view**.
6. On the **Create union view** page, enter the following information.

Table 174: Database view properties

Property	Description
Table	A technical name for the database view.
Display name	Name displayed for the database view. The display name is used, for example, to identify the database view in a database search or error output.
Description	Comments on using the database view.
View definition	Enter the database query as a Select statement.

Related topics

- [Database views of the Union type](#) on page 59
- [Creating database views with read-only type](#) on page 445
- [Creating new columns for database views with type view](#) on page 442

Creating new assignment tables

Technical details

- The technical identifier for the table is automatically formed according to the schema CCC<Table name>.


- The XObjectKey and XMarkedForDeletion columns are automatically generated.
- The table relations and column relations are generated automatically.

To create a new assignment (many-to-many) table in the Schema Extension.

1. Start the Launchpad and log in to the One Identity Manager database.
2. Open the Launchpad and select the **One Identity Manager Schema Extension** entry. This starts the Schema Extension program.
3. Click **Next** on the start page.
4. On the **Database connection** page, check the connection data for the One Identity Manager database.
5. On the **Select method** page, select **New relation table**.
6. On the **Create a relation table** page, enter the following information.

Table 175: Assignment table properties

Property	Description
Table	A technical name for the table.
Display name	Displays table name The display name is used, for example, to identify the table in a database search or for error output.
Description	Comments on using the table.
Create XOrigin column (for assignment requests)	You can create the origin column (XOrigin) optionally. The origin of an assignment is stored in this column as a bit field. Each time an entry is made in the assignment table the bit position is changed according to the assignment type.
Related tables	Use the Left table and Right table menus to specify which tables are included in the relation table.
Column names	In Column name fields, enter the relevant columns for each side of the table. Select the table's primary key column.

7. On the **Configure columns of table** page, enter display names for the columns.
8. (Optional) Click  to configure more column properties. For more information, see [Advanced configuration of columns](#) on page 443.

Related topics


- [Creating new tables](#) on page 436
- [Creating database views with read-only type](#) on page 445
- [Creating database views with Union type](#) on page 447

Creating indexes


Define indexes to optimize access to database columns. An index can contain one or more database columns.

NOTE: For tables that you create using the Schema Extension program, indexes are automatically created for the primary key column and the object key column (XObjectKey).

To create a new index in the Schema Extension

1. Start the Launchpad and log in to the One Identity Manager database.
2. Open the Launchpad and select the **One Identity Manager Schema Extension** entry. This starts the Schema Extension program.
3. Click **Next** on the start page.
4. On the **Database connection** page, check the connection credentials for the One Identity Manager database.
5. On the **Select method** page, select **Create index**
6. On the **Extend Table** page, under **Table**, select the table for which you want to create an index.
7. On the **Create index** page, define the columns for the index definition.
 - a. Click the  button.

This opens a dialog box where you can define the columns for the index. You can see all the columns in the table on the right-hand side of the dialog window. The columns on the left-hand side of the window belong to the index.
 - b. Enter the name of the index in the **Index name** input field.

A name is already suggested. You can change this as required.
 - c. On the right-hand side of the dialog window, select the column you want to add to the index.
 - d. Use the  button to add the column to the index.

Change the order of the columns in the index definition as required or remove a column from the index using the relevant button.
 - e. Click **OK**.

Removing custom schema extensions

NOTE: You can delete custom schema extensions only on databases with the **Test environment** or **Development system** staging level. It is not possible to transport the changes.

To remove custom schema extensions in the Schema Extension

1. Start the Launchpad and log in to the One Identity Manager database.
2. Open the Launchpad and select the **One Identity Manager Schema Extension** entry. This starts the Schema Extension program.
3. Click **Next** on the start page.
4. On the **Database connection** page, check the connection data for the One Identity Manager database.
5. On the **Select method** page, select **Remove extensions**.
6. On the **Remove extensions** page, select the custom schema extension that you want to remove.
7. To remove a custom table, select the table in the **Table** list and enable the **Remove whole table** option.
8. To remove custom columns, select the table in the **Table** list and select the columns under **Columns to remove**.
9. Click **Next**.
10. Confirm the security prompt with **Yes**.
11. Changes to the schema are displayed on the **System modifications** page. For more information, see [Adding schema extensions to the database](#) on page 453.

Related topics

- [Possible error messages due to custom schema extensions](#) on page 451

Possible error messages due to custom schema extensions

Error messages that arise when you delete schema extensions are displayed in the Schema Extension and logged in the system journal.

Table 176: Possible error messages

Step	Message	Solution
Checking Table empty	Table is not empty	Remove all objects from the table before you run the schema modification.
Checking template references	Column is referenced in template <TableName>.<ColumnName>	In the Designer, change all templates and formatting scripts that refer to the columns you want to delete. For more information, see Editing value

Step	Message	Solution
		templates on page 76 and Creating formatting scripts on page 83.
Checking referential integrity	Column is referenced as parent in RI <RelationID>	Remove all table relations from the QBMRelation table before you run the schema modification.
Checking dynamic RI	Column is referenced as parent in <TableName>.<ColumnName>	The column you want to delete is defined as a target for dynamically referenced tables. Use the Designer to resolve the reference. For more information, see Dynamic foreign key on page 102.
Checking indexes	Column is contained in index <TableName>.<IndexName>	Before you run the schema modification, change or delete the index so that the column is not referenced anymore.

Permissions for schema extensions

NOTE: At database level, the **End user role** database role is permitted for custom schema extensions.

For initial access to the schema extensions using One Identity Manager tools, select the permissions groups that contain the permissions for the schema extensions. After you have committed all the changes in the database, you can grant additional permissions through the Permissions Editor in the Designer. For more information, see the *One Identity Manager Authorization and Authentication Guide*.

To specify permissions groups in the Schema Extension

- On the **Permissions** page, use the menus to select
 - At least one custom permissions group that has read and write access
 - At least one custom permissions group that has read access only

Change labels for the schema extensions

Assign a change label to the schema extensions. Change labels are offered as export criteria in the Database Transporter when you create a customer transport package.

To assign a change label in the Schema Extension

- On the **Define change label** page, choose one of the following options.
 - **No change label**
 - **Add new change label:** In the **Change label** box, enter the name of the change label.
 - **Use existing change label:** Select a change label from the **Change label** menu.

For detailed information about working with change labels, see the *One Identity Manager Operational Guide*.

Adding schema extensions to the database

In this step, you add the schema extensions to the One Identity Manager database.

To add schema extensions in the Schema Extension

1. Changes to the schema are displayed on the **System modifications** page.
 - a. Set **Attach statements to existing file** to add the statements to an existing file.
 - b. Select **Save to file** and enter a file name. The statements are saved as an XML file.
2. Click **Continue**.
3. Confirm the security prompt with **Yes**.

The schema extensions are added to the database and the necessary extensions are made to the One Identity Manager system data model. This may take some time.
4. The current DBQueue Processor calculation tasks are displayed on the **System queue** page. After the calculation tasks have finished processing, click **Next**.
5. On the **Compilation** page, click **Next**.

The compilation process can take some time.
6. Click **Next** after compilation is complete.
7. On the last page, you return to the beginning of the wizard to enter more extensions or click **Finished** to end the program.

After completing the schema extensions, you can access them with One Identity Manager tools and make further changes.

Related topics

- [Recommendations for advanced configuration of custom schema extensions](#) on page 454
- [Managing custom database objects within the database](#) on page 456

Recommendations for advanced configuration of custom schema extensions

Once you have added custom tables or columns to the One Identity Manager schema, some additional steps are necessary to display the extensions in the Manager user interface.

General recommendations

- Edit the object layer using the One Identity Manager tools. This ensures that the data generated have the expected format.
- Always edit the object layer in the default language of a One Identity Manager installation, for example, **English - United States [en-US]**. For this purpose, set the login language to **English (USA)** in the One Identity Manager tools.
- The Designer contains a variety of consistency checks. Run these consistency checks and apply the repair methods after carrying out a schema extension and after making changes to table and column definitions. For more information about checking data consistency, see the *One Identity Manager Operational Guide*.

Recommendations for table definitions, column definitions and table relations.

The properties include, for example, display names, descriptions, display templates for tables and columns, value templates, formatting, required field definitions. For more information, see [One Identity Manager schema basics](#) on page 44.

- Use the Designer's Schema Editor to edit the table definitions and column definitions.
- Set the table usage types in the Designer. The table's usage type provides the basis for reports and the selection of tasks for daily maintenance.
- In the Designer, edit the display name and icon for the tables. These properties are used when you create object definitions for the table.
- In the Designer, define a display pattern to present table entries for instance in the result list of the One Identity Manager tools or in reports.

NOTE: You do not need to enter a display template for many-to-many tables. For these tables, the viDB.DLL forms the display template from the foreign keys.

- If there is a column combination for a table that needs to be unique, you define multi-column uniqueness in the Designer.
- In the Designer, arrange the tables in the schema overview of the Schema Editor. Otherwise, the schema overview shows all new tables in the upper left corner of the module. The colored module background will be automatically adjusted upon re-loading objects.
- In the Designer, record the display name for each column as well as a comment regarding display in the One Identity Manager tools.
- In the Designer, you can label columns containing passwords with **Encrypted**.
- The syntax type of the column definition is used to give the One Identity Manager tools the appropriate syntax highlighting or input assistance.
- In the Designer, flag columns containing a user account name with the **Central user account** value in the **Table lookup support** property.
- In the Designer, flag columns containing an email address with the **Email address** value in the **Table lookup support** property.
- To include tables when determining employees for user accounts or email addresses, in the Designer, enter the path to the **Person** object in the **Person object path for table lookup support** property. The resulting data is mapped in `QBMSplittedLookup.SplittedElement`. If an employee should not be determined for a table, enter the value **no**.

Recommendations for permissions

When you extend a schema using the Schema Extension program, you already assign permissions to permissions groups. You can carry on editing permissions in the Permissions Editor's Designer and also create permissions groups with the User & Permissions Group Editor. Permissions groups can be linked to application roles. The users are assigned to application roles and therefore receive the permissions they require. For more information, see the *One Identity Manager Authorization and Authentication Guide*.

Recommendations for object definitions

The data in the user interfaces is represented by means of objects. A generally applicable object definition without any limiting selection criteria is already created with the Schema Extension program. You can create other object definition constraints in addition. You create object definitions in the Designer. For more information, see [Object definitions for the user interface](#) on page 108.

Recommendations for navigation structure

Expand the menu to display the data in the Manager. Use the Designer's User Interface Editor to create menu items for navigation and result lists. For detailed information, see [User interface navigation](#) on page 112 and [Recommendations for editing menu navigation](#) on page 115.

Recommendations for user interface forms

Create or extend the forms for editing and displaying in the Manager. For detailed information, see [Recommendations for editing forms](#) on page 139, [Editing user interface forms](#) on page 140, [Forms for custom extensions](#) on page 147, and [Working with overview forms](#) on page 162.

Recommendations for task definitions

If you want to offer particular tasks for the objects in the Manager, you must create task definitions in the Designer. For more information, see [Task definitions for the user interface](#) on page 191.

- Create new task definitions if required.
- Task definitions are created for object definitions so that different tasks can be shown in the user interface depending on the selected objects. If required, create more object definitions.
- Assign the task definitions to the permissions groups for non role-based and role-based login.
- If required, assign a program function to the task definition. For more information, see the *One Identity Manager Authorization and Authentication Guide*.

Recommendations for analyzes

For data analysis purposes, you need to create statistics definitions and reports and incorporate these in the user interface. For more information, see [Statistics in One Identity Manager](#) on page 171 and [Reports in One Identity Manager](#) on page 399.

Recommendations for localizing texts

For language-dependent display of texts in the Manager such as column names, comments, menu items, and form names, translate the texts using the Designer's Language Editor. For more information, see [Language-dependent data representation](#) on page 210.

Managing custom database objects within the database

To create transport packages with the Database Transporter program and to create reports about the system configuration, information about database objects such as customized database tables and database columns, database procedures, features, triggers, indexes, or view definitions is stored in the database. The DBQueue Processor checks and updates this data.

NOTE: It is not usually necessary to edit the data manually although you might edit the comment for use in reports.

To customize database objects

1. In the Designer, select the **Base Data > Advanced > Modified SQL** category.
2. Select the database object.
3. Modify **Remarks**.

Table 177: Database object properties

Property	Description
Processing status	The processing status is used for creating custom configuration packages.
Remarks	Additional comments, for example, for using in system configuration reports.
Name	Database object name.
Modified	Specifies whether the database object has been changed.
Sort order	Order in which the data is presented.
Type	Type of database object, for example, procedure, function, trigger, index, view, custom table, custom column.

For detailed information about creating transport packages, see the *One Identity Manager Operational Guide*.

Web service integration

One Identity Manager offers you the option to integrate web services. For example, you can use web services to write data to applications, which cannot be connection to One Identity Manager as a default target system.

Data for external applications can be originate from any of the One Identity Manager schema's tables. They can, for example, be mapped as custom target systems.

Example:

The general data for a telephone system should be found from personnel data in One Identity Manager. The telephone system is mapped in One Identity Manager as a custom target system. One extension in the telephone corresponds a user account in One Identity Manager.

Once a new employee has been added in One Identity Manager, a new extension should become available in the telephone system. A new user account is added for each account definition. A web service passes the user account's main data onto the telephone system, where a new participant and telephone number is added. The web service passes this telephone number to One Identity Manager as the return value. The telephone number should be transferred to the employee's main data.

Proceed as follows

1. Set up a custom target system in One Identity Manager.
 - Select **Scripted synchronization** for the **Synchronized by** property.
2. Set up the server for provisioning the data.
 - Enter the server as the synchronization server in the custom target system.
3. Set up an account definition for automatic administration of user accounts in this target system.
4. Enter the required IT operating data.
5. Bind the web service to One Identity Manager. Use the generic web service call for this.

The web service integration wizard helps you to create scripts for provisioning data for the **Insert**, **Update**, and **Delete** default events. The provisioning processes are supplied by default through One Identity Manager.

6. Create additional scripts and processes for handling the web service return value.

TIP: When you insert, change, or delete containers, user accounts, and groups in a custom target system, the return values are saved by default as GUID objects in the database.

Create a process to add the telephone number from the object GUID to the employee main data.

For detailed information about setting up a custom target system, about account definitions, IT operating data and setting up a server, see the *One Identity Manager Administration Guide for Connecting to Custom Target Systems*.

Detailed information about this topic

- [Generic web service call](#) on page 459
- [Creating web service solutions with the Web Service Integration Wizard](#) on page 462
- [Modifying a web service solution](#) on page 466
- [Mapping processes in One Identity Manager](#) on page 220
- [Scripts in One Identity Manager](#) on page 341

Binding a web service

Create a custom script for integrating a web service into One Identity Manager. There is a wizard available to assist you. The Web Service Integration Wizard finds all the methods used by the web service and creates scripts to call the required methods. The data from One Identity Manager is passed as parameters to the method. Which operations in the external application can be run, is determined by the methods defined in the web service. The wizard created new entries in the `DialogWebService` and `DialogScript` tables.

The Web Service Integration Wizard supports different types of method calls. Each type supports the method call definition and, therefore, script creation to different degrees.

Generic web service call

You use the generic web service call to publish data from a custom target system to an external application through a web service. The Web Service Integration Wizard queries all the required parameters and generates scripts from them.

Prerequisites

The external application data is mapped in One Identity Manager as a custom target system.

- A custom target system is set up (UNSRotB table). The **Synchronized by** property has the value **Scripted synchronization**.
- A server for provisioning data is set up and stored as synchronization server in the custom target system.

For detailed information about setting up scripted provisioning, see the *One Identity Manager Target System Base Module Administration Guide*.

Default processes

One Identity Manager supplies default processes for provisioning data from custom target system to a web service.

To use these processes, the scripts you generated with the Web Service Integration Wizard must follow the naming convention:

<Customer prefix>_<table>_<Ident_UNSRot>_<event>.

IMPORTANT: If your target system contains a hyphen (-) in its name, you must remove it from the script function in the <Ident_UNSRot> part. Otherwise, error may occur during script processing.

Some of these processes handle the web service return values.

Table 178: Default processes for synchronizing by script

Object in custom target system (table)	Process	Saving the return value
Container (UNSContainerB)	VI_UnsContainer_Generic	UNSContainerB.ObjectGUID
User accounts (UNSAccountB)	VI_UnsAccountB_Generic	UNSAccountB.ObjectGUID
	VI_UnsAccountInGroup_Generic_Del	-
	VI_UnsAccountInGroup_Generic_Add	-
Groups (UNSGroupB)	VI_UnsGroup_Generic	UNSGroupB.ObjectGUID
	VI_UNSGroupBInUNSGroupB_Generic_Del	-
	VI_UnsGroupBInUNSGroupB_Generic_Add	-
Permissions controls	VI_UnsItem_Generic	-

Object in custom target system (table)	Process	Saving the return value
(UNSItemB)		
	VI_UnsGroupHasItem_Generic_Del	-
	VI_UnsGroupHasItem_Generic_Add	-
	VI_UnsAccountHasItem_Generic_Del	-
	VI_UnsAccountHasItem_Generic_Add	-

Direct web service call

The Web Service Integration Wizard finds all parameters that are defined in the method and from it, generates the script code. The parameters are passed in the function call. You can modify the parameters.

To run a script

- Create custom processes and pass the scripts and parameters to the process step.

Related topics

- [Mapping processes in One Identity Manager on page 220](#)

Self-defined web service call

The Web Service Integration Wizard finds all the parameters, which are defined in the method. You define how the parameter is passed.

To run a script

- Create custom processes and pass the scripts and parameters to the process step.

Related topics

- [Mapping processes in One Identity Manager on page 220](#)

Creating web service solutions with the Web Service Integration Wizard

Prerequisite

- Before you can bind a web service with the **WCF** service type, the `SvcUtil.exe` file must exist in the One Identity Manager installation directory.
Refer to Microsoft for information about where you can purchase this file.
- Before you can bind a web service with the **SOAP** service type, the `WSDL.exe` file must be on the server that carried out the provisioning.
Refer to Microsoft for information about where you can purchase this file.

To integrate a new web service

1. In the Designer, select the **Base Data > General > Web services** category.
2. Select the **Integrate new web service** task.
This starts the Web Service Integration Wizard.
3. Click **Next** on the start page.
4. Enter the access data and general web service properties on the **Integrate new web service** page.

Table 179: General properties of a web service

Property	Description
Web service name	Display name of the web service in One Identity Manager.
Description	Text field for additional explanation.
.NET namespace for the proxy code	Unique identifier for the .NET namespace.
Web service URL	URL at which the web service is run.
WSDL file URL	URL at which the <code>WSDL.exe</code> for the web service can be reached. If the <code>WSDL.exe</code> is not publicly available, it can also be saved locally. NOTE: If the web service operator changes the WSDL file, run the Web Service Integration Wizard again in order to implement the changes.
Service type	Type of web service.

Property	Description
Locked	Specifies whether the web service can be used.
User name	User name for logging in to the web service.
User domain	User domain.
User password and password confirmation	Password for logging in to the web service.
Proxy code generator	Path and file name for the proxy code generator. <ul style="list-style-type: none"> • If the WCF service type is selected, path to SvcUtil.exe file. • If the SOAP service type is selected, path to WSDL.exe file.

Table 180: Advanced properties of a web service

Property	Description
Proxy server URL	URL of the proxy server, if communication is routed through a proxy server.
Proxy server user name	User name for logging onto the proxy server.
Proxy server domain	Proxy server domain.
Proxy server password and password confirmation	Password and password confirmation for logging onto the proxy server.
Timeout for WSDL.exe	Timeout for accessing the WSDL file.
User-defined command line	<p>Command line for calling the proxy code generator. The command line can be extended by another parameter if required.</p> <p>Default command:</p> <pre>/nologo /language:VB "/namespace:%Namespace%" "/out:{0}" %Wsd1Url%</pre> <p>Example:</p> <pre>/nologo /language:VB "/namespace:EnricoHolidayWebservice" "/out:{0}" http://kayaposoft.com/enrico/ws/v1.0/index.php?wsdl</pre>

- a. Click **Check**.
This tests access to the web service.
 - b. If the test is successful, click **Next**.
5. The generated proxy code is shown on the page, **Create proxy code**.
The proxy code contains all web service methods, which are defined in the WSDL file and makes them available to the One Identity Manager script components.
6. On the **Select the service class** page, select the service class that you want to use.
If more than one service classes are available, a separate web service connection must be established for each service class.
7. On **Select method calls**, select the web service methods you want to use in One Identity Manager. A script is generated for each of the selected methods in the next step.
8. A script is generated to call the selected method on the **Generate web service call** page. Enter all the required parameter and properties for this.


- Click  to specify the type of method call.

Table 181: Type of method call

Type	Description
Self-defined web service call	For more information, see Self-defined web service call on page 461.
Direct web service call	For more information, see Direct web service call on page 461.
Generic web service call	For more information, see Generic web service call on page 459.

Table 182: Script properties

Property	Description
Script name	<p>Name of script. Prefix custom scripts with CCC_.</p> <p>Script names for the generic web service call must comply with the following pattern:</p> <p><customer prefix>_<table>_<target system>_<event></p> <p>Select the table, target system and event to create the script for. Parameter, value type, and data table are automatically determined from the selected table.</p>
Parameter	Name of the parameter.
Value type	Parameter data type.
Data table	Data table that contains the data to be transferred to the web service.
Return value	Data type of the parameter containing the return value.

Table 183: Data transfer

Property	Description
Parameter	Parameter transferred to the web service.
Value type	Parameter data type.
Mapped from	Parameter from the defined script properties. Open the menu and assign the associated parameters. If necessary, select the column from the data table which contains the value to be passed.

In the **Script code** view, you see the generated script. You can use extended edit mode to edit the script.

TIP: The script calls the `VID_GetWcfWebService` function, which in turn, uses the `GetWcfBinding` and `GetWcfEndpointAddress` functions. These three functions can be overwritten.

9. To end the Web Service Integration Wizard, click **Finish**.
10. Save the changes.
11. Compile the database.

Related topics

- [Scripts in One Identity Manager](#) on page 341
- [Overriding scripts](#) on page 368

Modifying a web service solution

You can change or extend an existing web service solution at any time. This overwrites the existing script or adds new scripts.

To extend a web service solution

1. In the Designer, select the **Base Data > General > Web services** category.
2. In the List Editor, select the web service.
3. Select the **Create web service call** task.
This start the Web Service Integration Wizard.
4. Follow the wizard's instructions.
5. Save the changes.
6. Compile the database.

To edit a web service solution

1. In the Designer, select the **Base Data > General > Web services** category.
2. Select the web service in the List Editor.
3. Select the **Edit web service** task.
This start the Web Service Integration Wizard.
4. Follow the wizard's instructions.
On the **Create proxy code** page, you can edit the generated proxy code.
 - To test the changes, set **Compare with previous version**.
5. Save the changes.
6. Compile the database.

NOTE: If the web service integration wizard is rerun, the proxy code is generated again. All manual changes to the proxy code are overwritten.

Detailed information about this topic

- [Creating web service solutions with the Web Service Integration Wizard](#) on page 462

Deleting a web service solution

To delete a web service solution from the database

1. Delete the web service.
2. Delete all associated custom scripts.
3. Determine all other custom element of your web service solution and delete them.
4. Compile the database.

One Identity Manager as SCIM 2.0 service provider

One Identity Manager provides an interface corresponding to the System for Cross-domain Identity Management (SCIM) 2.0. The interface enables the import and export of One Identity Manager objects by a SCIM client. For example, the interface can be used to:

- Import identity, department, cost center, and location data from an HR system
- Import user account and group data from a cloud system
- Match custom target systems that have a SCIM V2.0 client
- Provision information about identities through a standardized interface for further processing in third-party systems

The SCIM 2.0 service provider for One Identity Manager is provided as a plugin for the API Server and you can select it when you install the API Server. Further configuration of the SCIM plugin itself is not necessary. You can check in the API Server administration portal if the SCIM plugin was installed and activated successfully. For more information on installing an API Server, see the *One Identity Manager Installation Guide*.

Detailed information about this topic

- [Endpoints and base URL](#) on page 468
- [SCIM plugin features](#) on page 469
- [Authenticating SCIM clients](#) on page 469
- [Authenticating SCIM plugins in One Identity Manager](#) on page 470
- [Special features of generating the SCIM schema](#) on page 470
- [SCIM plugin requests](#) on page 472

Endpoints and base URL

NOTE: The providers and endpoints supplied with the interface are fixed and cannot be configured.

The SCIM 2.0 interface can be found in the API Server installation directory under the fixed subdirectory /ApiServer/scim/v2. The base URL is formed as follows:

<http | https>://[<subdomain | server name>.<domain name>]/ApiServer/scim/v2

There are other fixed endpoints. The endpoints are called without any further URL parameters.

- /ServiceProviderConfig

The endpoint provides information about the options implemented in the service provider, such as the authentication types on offer. The endpoint can be accessed without authentication.

- /Schemas

The return structure of the endpoint defines all data objects and their properties supported by the provider.

- /ResourceTypes

When the endpoint is called, the metadata for the data objects published through the /Schemas endpoint are returned. These are linked to an endpoint based on the base URL under which the respective provider of a specific data object type can be reached.

Related topics

- [SCIM plugin features](#) on page 469
- [Authenticating SCIM clients](#) on page 469
- [Special features of generating the SCIM schema](#) on page 470
- [SCIM plugin requests](#) on page 472

SCIM plugin features

The return structure of the /ServiceproviderConfig endpoint defines, among other things, the features that are supported by the SCIM plugin.

- **patch:** When transferring changes, only single operations on object properties are transferred, not the complete object to be changed.
- **filter:** This supports filtering of objects immediately on request or in patch operations.

Authenticating SCIM clients

An authenticationSchemes list is included in the structure returned by the /ServiceproviderConfig endpoint. The list publishes the possible authentication and

authorization methods with which the SCIM plugin can be used.

Supported are:

- HTTP Basic access authentication
- NTLM Authentication and Kerberos
- OAuth 2.0 client authorization

Related topics

- [Authenticating SCIM clients](#) on page 469

Authenticating SCIM plugins in One Identity Manager

To access the One Identity Manager database, the SCIM plugin be authenticated. Authentication is carried out by the One Identity Manager authentication modules. For more information, see the *One Identity Manager Authorization and Authentication Guide*.

The authentication modules are checked in the following order and the first successful authentication module is used for logging in. Ensure sure that at least one authentication module is enabled and configured.

1. Employee (Person)
2. Active Directory user account (ADSAccount)
3. Employee (role-based) (RoleBasedPerson)
4. Active Directory user account (role-based) (RoleBasedADSAccount)
5. HTTP Header (role-based) (RoleBasedHTTPHeader)
6. HTTP Header (HTTPHeader)
7. OAuth 2.0/OpenID Connect (role-based) (OAuthRoleBased)

Related topics

- [Authenticating SCIM clients](#) on page 469

Special features of generating the SCIM schema

The SCIM 2.0 schema exported to the /Schemas endpoint is generated from the One Identity Manager schema. The table definitions to take into account are supplied as are the

M:N figures to publish. A data object description with simple and complex properties is created for each table.

Columns in a table

The columns of a table are mapped to simple properties of integral types.

Foreign key relations

The foreign key relations of a table are only included in the schema if the reference's target table is also part of the schema. In this case, a complex property is published with the foreign key's column name. This complex property has the value, \$ref, and displayName properties.

The complex property is marked in the schema with the "returned" : "request" attribute. To be able to read this data, the property must be explicitly requested by the SCIM client using the attributes URL parameter.

Example:

```
https://<servername>.<domainname>/ApiServer/scim/v2/Locality/0294ce3c-8286-4641-bc13-9bcd4a2fd714?attributes=cn,City,UID_PersonHead
```

M:N tables

M:N tables are published under the members complex property in the schema. This also applies if there are several M:N tables to take into account. This complex property defines an array of subelements that have the value, type, \$ref, and display properties.

The members complex property is marked in the schema with the attribute "returned" : "request". To be able to read this data, the property must be explicitly requested by the SCIM client via the URL parameter attributes.

Example:

```
http://<servername>.<domainname>/ApiServer/scim/v2/UNSGroupB/94bbe614-0a6e-4659-8fe9-20da94d967c8?attributes=cn,members
```

If several M:N tables are grouped together, the distinction, from which table the respective element originates, is made on the basis of the value in the type property. Ensure that the value in the type property is also passed when writing to the property. The values accepted as correct are defined in the schema on the respective type subattribute as a list in canonicalValues.

If the value for type cannot be determined for the SCIM client, it can be left blank and is not transmitted with the PUT or PATCH request. The SCIM plugin tries to determine the correct

type. The element's ID passed in the value property is used to search in all One Identity Manager tables that are part of the members definition. If the object is found in the process, the operation can be performed.

SCIM plugin requests

Base URL requests

The SCIM 2.0 specification provides optional requests for the SCIM service provider base URL. These requests can contain a filter expression if required. This is mainly used to search for objects when their endpoint is not known exactly and so the search must be across endpoints.

The SCIM plugin supports these requests. In the filter, only logical OR operations and the comparison operators eq, sw, ew as well as co are allowed, which must reference the Resourcetype metadata.

Example:

```
https://<servername>.<domainname>/ApiServer/scim/v2?filter=
(meta.Resourcetype eq "Locality") or (meta.Resourcetype sw "D")
```

The result can contain a list of objects of different types, but the number of returned elements must not exceed 10,000 for load and performance reasons. Otherwise an error message of type tooMany is returned. The search condition should be refined and the result should be more restricted.

Endpoint URL requests

The SCIM 2.0 specification provides for optional filter, attributes, count, and startIndex parameters for requests to the endpoints defined by /ResourceTypes. Requests with the ID of a concrete object (the URL contains the id of the object) can have the excludedAttributes and attributes parameters. The SCIM plugin supports these parameters.

Endpoint requests return a list of all elements (or all elements matching the filter). This allows the SCIM client to initiate index-based paging by specifying the desired number of records per page (count and startIndex parameters).

Example: Endpoint request

```
http://<servername>.<domainname>/ApiServer/scim/v2/Person
```


Example: Request the first 100 elements of an endpoint with paging

`http://<servername>.<domainname>/ApiServer/scim/v2/Person?startindex=1&count=100`

Example: Endpoint request with filter

`http://<servername>.<domainname>/ApiServer/scim/v2/Person?filter=InternalName co "Y"`

Example: Endpoint request with filter and output of two additional properties

`http://<servername>.<domainname>/ApiServer/scim/v2/Person?filter=InternalName co "Y"&attributes=ExitDate,TechnicalEntryDate`

Example: Endpoint request for an object

`http://<servername>.<domainname>/ApiServer/scim/v2/UNSGroupB/94bbe614-0a6e-4659-8fe9-20da94d967c8`

Example: Endpoint request for specific properties of an object

`http://<servername>.<domainname>/ApiServer/scim/v2/UNSGroupB/94bbe614-0a6e-4659-8fe9-20da94d967c8?attributes=cn,members`

SOAP Web Service

One Identity Manager's SOAP Web Service provides a SOAP interface for accessing the One Identity Manager object model. The SOAP Web Service manages a connection pool. Not every call opens a new connection. Not all object layer functions are supported by the SOAP Web Service. The SOAP Web Service supplies methods for single objects, object lists, and function call.

Table 184: Methods for single objects

Method	Description
CreateSingleObject	Adds a new single object.
GetCompleteSingleObject	Loads a single complete object from the database with all parameters.
GetCompleteSingleObjectEx	Functionality analog to GetCompleteSingleObject with support for a multi-column primary key.
GetSingleObject	Loads a single object from the database.
GetSingleObjectEx	Functionality analog to GetSingleObject with support for a multi-column primary key.
ChangeSingleObject	Saves changes to a single object.
ChangeSingleObjectEx	Functionality analog to ChangeSingleObject with support for a multi-column primary key.
DeleteSingleObject	Deletes a single object.
DeleteSingleObjectEx	Functionality analog to DeleteSingleObject with support for a multi-column primary key.
Exists	Does a specific single object exist?
ExistsEx	Functionality analog to Exists with support for a multi-column primary key.
GetSingleProperty	Gets a single value from an object.
GetSinglePropertyEx	Functionality analog to GetSingleProperty with support for a multi-column primary key.

Table 185: Methods for object lists

Method	Description
GetListObject	Loads a list of objects.
GetListObjectWithDisplays	Loads a list of objects with data additional to the primary key about the columns to load.

Table 186: Methods for function calls

Function	Description
InvokeCustomizer	Calls a customizer method for an object.
InvokeCustomizerEx	Functionality analog to InvokeCustomizer with support for a multi-column primary key.
InvokeDialogMethod	Calls a dialog method for a dialog object.
InvokeDialogMethodEx	Functionality analog to InvokeDialogMethod with support for a multi-column primary key.
FireGenEvent	Generates processes of a specific event.
FireGenEventEx	Functionality analog to FireGenEvent with support for a multi-column primary key.

Detailed information about this topic

- [Installing and configuring the SOAP Web Service](#) on page 475
- [Examples of calls](#) on page 482

Installing and configuring the SOAP Web Service

To install the SOAP Web Service, you must provide a server on which the following software is already installed:

- Windows operating systems
The following versions are supported:
 - Windows Server 2022
 - Windows Server 2019
 - Windows Server 2016

- Windows Server 2012 R2
- Windows Server 2012
- Microsoft .NET Framework Version 4.7.2 or later
- Microsoft Internet Information Services 10, 8.5, 8, 7.5, or 7 with ASP.NET 4.7.2, and the Role Services:
 - Web Server > Common HTTP Features > Static Content
 - Web Server > Common HTTP Features > Default Document
 - Web Server > Application Development > ASP.NET
 - Web Server > Application Development > .NET Extensibility
 - Web Server > Application Development > ISAPI Extensions
 - Web Server > Application Development > ISAPI Filters
 - Web Server > Security > Basic Authentication
 - Web Server > Security > Windows Authentication
 - Web Server > Performance > Static Content Compression
 - Web Server > Performance > Dynamic Content Compression

Required permissions

- The user account that the Internet Information Service runs under, needs write access (**MODIFY**) to the installation directory.
- The following permissions are required for automatic updating:
 - The user account for updating requires write permissions for the application directory.
 - The user account for updating requires the local security policy **Log on as a batch job**.
 - The user account running the application pool requires the local security policies **Replace a process level token** and **Adjust memory quotas for a process**.

Detailed information about this topic

- [Installing the SOAP Web Service](#) on page 476
- [SOAP Web Service configuration](#) on page 479
- [Displaying the SOAP Web Service's status](#) on page 481
- [Uninstalling SOAP Web Service](#) on page 482

Installing the SOAP Web Service

IMPORTANT: Start the SOAP Web Service installation locally on the server.

To install the SOAP Web Service

1. Launch `autorun.exe` from the root directory of the One Identity Manager installation medium.
2. On the start page of the installation wizard:
 - a. Change to the **Installation** tab.
 - b. In the **Web-based components** pane, click **Install**.
Starts the Web Installer.
3. On the start page of the Web Installer, select **Install SOAP Web Service** and click **Next**.
4. On the **Database connection** page, do the following:

TIP: It is recommended to establish a connection through the application server.

 - To use an existing connection to the One Identity Manager database, select it in the **Select a database connection** menu.
- OR -
 - To create a new connection to the One Identity Manager database, click **Add new connection** and enter a new connection .
5. Select the authentication method and, under **Authentication method**, enter the login data for the database.
6. On the **Select setup target** page, configure the following settings and click **Next**.

Table 187: Settings for the installation target

Setting	Description
Application name	Name used as application name, as in the title bar of the browser, for example.
Target in IIS	Internet Information Services web page on which to install the application.
Enforce SSL	Specifies whether secure or insecure websites are available to install. If the option is set, only sites secured by SSL can be used for installing. This setting is the default value. If this option is not set, insecure websites can be used for installing.
URL	The application's Uniform Resource Locator (URL).
Install dedicated application pool	Specifies whether an application pool is installed for each application. This allows applications to be set up independently of one another. If this option is set, each application is installed in its own application pool.
Application pool	The application pool to use. This can only be entered if the Install dedicated application pool option is not set.

Setting	Description
	<p>If you use the DefaultAppPool default value, the application pool has the following syntax:</p> <pre><application name>_POOL</pre>
Identity	<p>Permissions for running an application pool. You can use a default identity or a custom user account.</p> <p>If you use the ApplicationPoolIdentity default value, the user account has the following syntax:</p> <pre>IIS APPPOOL\<application name>_POOL</pre> <p>You can authorize another user by clicking ... next to the box, enabling the Custom account option and entering the user and password.</p>
Web authentication	<p>Type of authentication against the web application. You have the following options:</p> <ul style="list-style-type: none"> • Windows authentication (single sign-on) <p>The user is authenticated against the Internet Information Services using their Windows user account and the web application logs in the employee assigned to the user account as role-based. If single sign-on is not possible, the user is diverted to a login page. You can only select this authentication method if Windows authentication is installed.</p> • Anonymous <p>Login is possible without Windows authentication. The user is authenticated against the Internet Information Services and the web application anonymously, and the web application is directed to a login page.</p>
Database authentication	<p>NOTE: You can only see this section if you have selected a SQL database connection on the Database connection page.</p> <p>Type of authentication against the One Identity Manager database. You have the following options:</p> <ul style="list-style-type: none"> • Windows authentication <p>The web application is authenticated against the One Identity Manager database with the same Windows user account that your application pool uses. Login is possible with a user-defined user account or a default identity for the application pool.</p> • SQL authentication

Setting	Description
	Authentication is completed with a SQL Server login and password. The SQL Server login from the database connection is used. Use the [...] button to enter a different SQL login, for example, if the application is run with a access level for end users. This access data is saved in the web application configuration as computer specific encrypted.

7. Specify the user account for automatic updating of the application server on the **Set update credentials** page.

1. The user account is used to add or replace files in the application directory.
 - Set **Use IIS credentials for update**, if you want to use the user account that is running the application for updates.
 - Set **Use other credentials for updates**, if you want to use another user account and enter the domain, user name, and password for the user.

8. Installation progress is displayed on the **Setup is running** page. Once installation is complete, click **Next**.

The Web Installer generates the web application and the corresponding configuration files (web.config) for each directory.

9. Click **Finish** on the last page to end the program.

SOAP Web Service configuration

The SOAP Web Service configuration is found in web.config in the installation directory. You can use any text editor to edit this file.

Table 188: Configurable options in the “web.config” configuration file

Section	Option	Permitted values	Description
connectionString			Database connection parameter.
runtime dirs	key="Cache"	value = "<path>"	Directory for storing the cache directory. Default: value="C:\inetpub\wwwroot\<web service name>\App_Data\Cache\DB"
	key="AssemblyCache"	value = "<path>"	Directory for storing the cache directory.

Section	Option	Permitted values	Description
			Default: value="C:\inetpub\wwwroot\ \<web service name>\App_ Data\Cache\Assemblies"
settings	key="timeout"	value="<time>"	Timeout for connections in the application pool. Default: value="00:05:00"
	key="maxconnectionlifetime"	value="<time>"	Maximum length of time to maintain the connections. After this time limit has expired, all the connections are closed even if the timeout has not expired yet. Default: value="00:05:00"
	key="usepropertybag"	value = "True" value = "False"	Specifies whether a property bag is used. A property bag is used when object properties are populated in order to maintain the particular fill order that is required because of side effects or templates. Permitted values are: <ul style="list-style-type: none"> • False: Value are set in the object in the order in which they were given. • True: The fill order is determined by the metadata. Default: value="True"
	key="ignoreinvisiblevalues"	value = "True" value = "False"	Specifies whether values that the user is not permitted to see are not returned. Permitted values are: <ul style="list-style-type: none"> • False: Values that the

Section	Option	Permitted values	Description
			<p>user is not allowed to see, generate an error message.</p> <ul style="list-style-type: none"> • True: Values that the user is not allowed to see, are not returned. If this value is set, the user is issued an error message. <p>Default: value="True"</p>
	key = "logdirectory"	value = "<path>"	<p>Log directory.</p> <p>Default: value = "C:\inetpub\wwwroot\<web service name>\App_Data\Logs"</p>
	key = "allowwebservicemethods"	value = "List of methods"	Semicolon-delimited list of permitted web service methods.
	key = "allowfunctions"	value = "List of functions"	List of the permitted functions for each CallFunction method. If no other function is given, all functions are permitted.

Related topics

- [Column dependencies for setting values](#) on page 84

Displaying the SOAP Web Service's status

The SOAP Web Service can be reached over a browser under:

http://<server>/<application name>

https://<server>/<application name>

TIP: You can open the web server's status display in the Job Queue Info. In the Job Queue Info, select **View > Server state** in the menu and, on the **Web servers** tab, open the web server status display from the **Open in browser** context menu.

In addition, API documentation is available here.


Uninstalling SOAP Web Service

To uninstall a web application

1. Launch `autorun.exe` from the root directory of the One Identity Manager installation medium.
2. On the start page of the installation wizard:
 - a. Change to the **Installation** tab.
 - b. In the **Web-based components** pane, click **Install**.

This starts the Web Installer.

3. On the Web Installer start page, click **Uninstall a web application** and click **Next**.
4. On the **Uninstall a web application** page, double-click the application that you want to remove.

The  icon is displayed in front of the application.
5. Click **Next**.
6. On the **Database connection** page, select the database connection and authentication method and enter the corresponding login data.
7. Click **Next**.
8. Confirm the security prompt with **Yes**.
9. The uninstall progress is displayed on the **Setup is running** page.
10. Once installation is complete, click **Next**.
11. On the **Wizard complete** page, click **Finish**.
12. Close the `autorun` program.

Examples of calls

You will find an overview of the methods supplied under [SOAP Web Service](#) on page 474. In the following there are some examples of a web service client calls in the programming language C#.

Preparation

Authentication is carried out by means of an authentication string containing an authentication module and the login data to use. You must create an instance of the web service and the object for the login data to log in to the system. The login data is passed to following calls.

Example:

```
var svc = new Q1IMServiceSoapClient();  
var login = new LoginInformation
```

```
{ AuthString = "Module=DialogUser;User=viadmin;Password=" };
```

Table 189: Examples of authentication

Authentication module	Example
System users	Module=DialogUser;User=<user name>;Password=<password>
Employee	Module=Person;User=<central user account>;Password=<password>
Active Directory user account (role-based)	Module=RoleBasedADSAccount
Active Directory user account (manual input/role-based)	Module=RoleBasedManualADS;User=<AD user name>;Password=<AD password>

For detailed information about the One Identity Manager authentication modules, see the *One Identity Manager Authorization and Authentication Guide*.

GetListObject

This method returns an array of objects, which correspond to the given WHERE clause. The returned array contains the object's primary key and a [DISPLAY] special key, which contains the object's display value.

Example:

```
Q1IMService.KeyValuePair[][] objects = svc.GetListObject(login, "Person",
    "FirstName like 'Hal%'");
```

GetListObjectWithDisplays

This method works in the same way as GetListObject and allows you to enter details of additional columns to be loaded.

Example:

In the example, the FirstName and LastName columns are available.

```
Q1IMService.KeyValuePair[][] objects = svc.GetListObjectWithDisplays(login,
    "Person",
    "FirstName like 'Hal%'",
    new [] {"FirstName", "LastName"});
```

GetCompleteSingleObject

All the properties of the object that is defined by the primary key are loaded by the method.

Example:

```
Q1IMService.KeyValuePair[] singleValues = svc.GetCompleteSingleObject(login,
    "Person", "UID_Person", "746a5662-054b-4531-a889-1c135dad4c05");
```

GetSingleObject

Properties of a single object are loaded with this method.

Example:

In the example, the `FirstName` and `LastName` columns and the display value are loaded. The display value is given in the `[DISPLAY]` key.

```
Q1IMService.KeyValuePair[] values = svc.GetSingleObject(login, "Person",
    "UID_Person", "746a5662-054b-4531-a889-1c135dad4c05",
    new[] { "FirstName", "LastName" });
```

ChangeSingleObject

This method changes individual properties of an object.

Example:

In the example, the `Description` column of the employee with the corresponding `UID_Person` is modified.

```
var values = new[]
{
    new Q1IMService.KeyValuePair
    {
        Key = "Description",
        Value = "Created by webservice"
    }
};
svc.ChangeSingleObject(login, "Person", "UID_Person",
    "746a5662-054b-4531-a889-1c135dad4c05", values);
```

ChangeSingleObjectEx

Modifying an object with this method is done in the same way as with `ChangeSingleObject`, but here the primary key value is passed as a Key-Value-Pair-Array.

Example:

```
var values = new[]
{
    new Q1IMService.KeyValuePair
    {
        Key = "Description",
        Value = "Created by webservice"
    }
}
```

```

    };
    var keys = new[]
    {
        new Q1IMService.KeyValuePair
        {
            Key = "UID_Person",
            Value = "746a5662-054b-4531-a889-1c135dad4c05"
        }
    };
    svc.ChangeSingleObjectEx(login, "Person", keys, values);

```

DeleteSingleObject

This method deletes an object.

Example:

In this example, the employee with the corresponding UID is deleted from the database.

```

svc.DeleteSingleObject(login, "Person",
    "UID_Person", "746a5662-054b-4531-a889-1c135dad4c05");

```

DeleteSingleObjectEx

Using this method, you can delete objects with a multicolumn primary key (from example, from M:N tables).

Example:

```

svc.DeleteSingleObjectEx (
    login,
    "OrgHasApp",
    new []
    {
        new Q1IMService.KeyValuePair { Key = "UID_Org", Value = <UID> },
        new Q1IMService.KeyValuePair { Key = "UID_Application", Value = <UID>}
    });

```

CreateSingleObject

A new object is created in the database with this object.

Example:

In this example, the employee "Jon Doe" is created.

```

var values = new[]

```

```

{
    new Q1IMService.KeyValuePair {Key = "FirstName", Value = "John"},
    new Q1IMService.KeyValuePair {Key = "LastName", Value = "Doe"}
};
svc.CreateSingleObject(login, "Person", values);

```

Exists

This method checks the existence of an object.

Example:

```

bool exists = svc.Exists(login, "Person",
    "UID_Person", "746a5662-054b-4531-a889-1c135dad4c05");

```

GetSingleProperty

This method can be implemented to find a single property.

Example:

```

string description = svc.GetSingleProperty(login, "Person",
    "UID_Person", "746a5662-054b-4531-a889-1c135dad4c05",
    "Description");

```

InvokeCustomizer

The SOAP Web Service supports a `InvokeCustomizer` method, which calls a function for an object in the database. The first three parameters specify the object on which the method is called. The `customizerName` parameter provides the function name. An array of strings follows which contains the fully qualified name of the parameter data types. These are passed to the calling function. The following array of strings contains textual representation of the parameter.

How the function works

- First, the database is opened and the object specified by `objectType`, `pkName` and `pkValue` is retrieved.
- Then the runtime data types specified by `parameterTypes` are determined.
- After that, text representations of the parameters are converted from the value array to the corresponding runtime data types.
- The function is called with these values.

If the function to be called has no parameters, you can transfer the `null` value to the function for the `parameterTypes` and **parameters** parameters.

Example:

In this example, the method "TestMethod" is called for a Person type object with the primary key UID_Person and the given value. In this case, both parameters of System.String and System.Int32 type are transferred with the values "Foo" and "4711".

```
svc.InvokeCustomizer (login, "Person",  
    "UID_Person", "0000644F-C139-4B25-8D1C-5ECB93067E79",  
    "TestMethod",  
    new [] {"System.String", "System.Int32"},  
    new [] {"foo", "4711"});
```

InvokeDialogMethod

The method can call a dialog method on an object. Dialog methods do not have any parameters and no return values. The call is similar to the InvokeCustomizer call.

Example:

In this example, the "TestDialogMethod" method is called for a specific person. "TestDialogMethod" is the name of the corresponding to DialogMethod.MethodName method.

```
svc.InvokeDialogMethod (login,  
    "Person",  
    "UID_Person", "746a5662-054b-4531-a889-1c135dad4c05",  
    "TestDialogMethod");
```

FireGenEvent

A specific event is generated by this method. There is the option to enter other generating parameters.

```
public void FireGenEvent(  
    string objectType, string pkName, string pkValue,  
    string eventName, KeyValuePair[] parameters);
```

Example:

In this example, the "EXPORT_DATA" event is generated without additional parameters.

```
svc.FireGenEvent(login, "Person",  
    "UID_Person", "746a5662-054b-4531-a889-1c135dad4c05",  
    "EXPORT_DATA", new Q1IMService.KeyValuePair[] { });
```

CallFunction

This method calls a One Identity Manager script function.

Example:

In the example, the VI_BuildInitials script is called.

```
svc.CallFunction(login, "VI_BuildInitials",  
    new string [] {"John", "Doe"});
```

One Identity Manager as SPML provisioning service provider

One Identity Manager enables data exchange with other vendor systems using SMPL. SPML stands for Service Provisioning Markup Language and defines a standardized interface for exchanging provisioning information. SPML version 2 (SPMLv2) was published in April 2006 by the Organization for the Advancement of Structured Information Standards (OASIS, www.oasis-open.org). The interface provides a means to simplify and standardize data exchange in the context of complex provisioning solutions and environments.

One Identity Manager can be implemented as SPML client or as SPML provider. At this point we shall only go into the One Identity Manager configuration as SPML provider. The SPML Provider supports the entire One Identity Manager schema. The objects and relations to be administrated through the SPML provider can be configured to meet customer requirements.

Detailed information about this topic

- [SPML web service](#) on page 488
- [Installing and configuring the SPML web service](#) on page 489
- [Configuring the One Identity Manager schema](#) on page 496
- [Testing SPML web service functionality](#) on page 498

SPML web service

A web service called the SPML web service is provided to function as an SPML service provider. SPML web service conforms to SPMLv2 and its implementation is based on the OASIS publication. It makes the main operations such as adding, deleting, and changing objects available as well as extensions for searching and referencing objects.

SPML Web Service supports the following defined SPMLv2 functions:

Table 190: SPMLv2 supported functions

Function	Description
listTargetsRequest	Returns the provider target system with its specific schema. The SPML provider supports the One Identity Manager schema exclusively.
addRequest	Adds a new object in the given provider target system with the given properties.
lookupRequest	Returns the properties of an object identified by a key.
modifyRequest	Changes the properties of a key identified object in the given provider target system.
deleteRequest	Deletes a key identified object in the provider target system.
searchRequest	Returns all objects in the provider target system that fulfill the search criterion.
iterateRequest	Returns other data sets from a search assuming not all of search results have been sent to the client.
closeIteratorRequest	Closes an active search and informs the provider that no further results are required.

The Reference extension allows you to maintain references between different objects from the provider's target system. There are two different types of references for this.

- Reference type **owner**
References of the type **owner** result in foreign key relations in One Identity Manager.
- Reference type **memberOf**
References of the type **memberOf** result in many-to-many assignments in One Identity Manager.

Installing and configuring the SPML web service

To install SPML web service, a server has to be made available on which the following software is already installed:

- Windows operating systems
The following versions are supported:
 - Windows Server 2022
 - Windows Server 2019

- Windows Server 2016
- Windows Server 2012 R2
- Windows Server 2012
- Microsoft .NET Framework Version 4.7.2 or later
- Microsoft Internet Information Services 10 or 8.5 or 8 or 7.5 or 7 with ASP.NET 4.7.2 and the Role Services:
 - Web Server > Common HTTP Features > Static Content
 - Web Server > Common HTTP Features > Default Document
 - Web Server > Application Development > ASP.NET
 - Web Server > Application Development > .NET Extensibility
 - Web Server > Application Development > ISAPI Extensions
 - Web Server > Application Development > ISAPI Filters
 - Web Server > Security > Basic Authentication
 - Web Server > Security > Windows Authentication
 - Web Server > Performance > Static Content Compression
 - Web Server > Performance > Dynamic Content Compression

Required permissions

- The user account that the Internet Information Service runs under, needs write access (MODIFY) to the installation directory.
- The following permissions are required for automatic updating:
 - The user account for updating requires write permissions for the application directory.
 - The user account for updating requires the **Log on as a batch job** local security policy.
 - The user account running the application pool requires the **Replace a process level token** and **Adjust memory quotas for a process** local security policies.

Detailed information about this topic

- [Installing the SPML Web Service](#) on page 490
- [Configuring the SPML web service](#) on page 493

Installing the SPML Web Service

IMPORTANT: Start the SPML web service installation locally on the server.

To install the SPML web service

1. Launch autorun.exe from the root directory of the One Identity Manager installation medium.
2. On the start page of the installation wizard:
 - a. Change to the **Installation** tab.
 - b. In the **Web-based components** pane, click **Install**.
Starts the Web Installer.
3. On the Web Installer start page, select **Install SPML web service** and click **Next**.
4. On the **Database connection** page, do the following:

TIP: It is recommended to establish a connection through the application server.

 - To use an existing connection to the One Identity Manager database, select it in the **Select a database connection** menu.
 - OR -
 - To create a new connection to the One Identity Manager database, click **Add new connection** and enter a new connection .
5. On the **Select setup target** page, configure the following settings and click **Next**.

Table 191: Settings for the installation target

Setting	Description
Application name	Name used as application name, as in the title bar of the browser, for example.
Target in IIS	Internet Information Services web page on which to install the application.
Enforce SSL	Specifies whether secure or insecure websites are available to install. If the option is set, only sites secured by SSL can be used for installing. This setting is the default value. If this option is not set, insecure websites can be used for installing.
URL	The application's Uniform Resource Locator (URL).
Install dedicated application pool	Specifies whether an application pool is installed for each application. This allows applications to be set up independently of one another. If this option is set, each application is installed in its own application pool.
Application pool	The application pool to use. This can only be entered if the Install dedicated application pool option is not set. If you use the DefaultAppPool default value, the application pool has the following syntax: <application name>_POOL

Setting	Description
Identity	<p>Permissions for running an application pool. You can use a default identity or a custom user account.</p> <p>If you use the ApplicationPoolIdentity default value, the user account has the following syntax:</p> <pre>IIS APPPOOL\<application name>_POOL</pre> <p>You can authorize another user by clicking ... next to the box, enabling the Custom account option and entering the user and password.</p>
Web authentication	<p>Type of authentication against the web application. You have the following options:</p> <ul style="list-style-type: none"> Windows authentication (single sign-on) <p>The user is authenticated against the Internet Information Services using their Windows user account and the web application logs in the employee assigned to the user account as role-based. If single sign-on is not possible, the user is diverted to a login page. You can only select this authentication method if Windows authentication is installed.</p> Anonymous <p>Login is possible without Windows authentication. The user is authenticated against the Internet Information Services and the web application anonymously, and the web application is directed to a login page.</p>
Database authentication	<p>NOTE: You can only see this section if you have selected a SQL database connection on the Database connection page.</p> <p>Type of authentication against the One Identity Manager database. You have the following options:</p> <ul style="list-style-type: none"> Windows authentication <p>The web application is authenticated against the One Identity Manager database with the same Windows user account that your application pool uses. Login is possible with a user-defined user account or a default identity for the application pool.</p> SQL authentication <p>Authentication is completed with a SQL Server login and password. The SQL Server login from the database connection is used. Use the [...] button to enter a different SQL login, for example, if the application is run with a</p>

Setting	Description
	access level for end users. This access data is saved in the web application configuration as computer specific encrypted.

- Specify the user account for automatic updating of the application server on the **Set update credentials** page.

The user account is used to add or replace files in the application directory.

- Set **Use IIS credentials for update**, if you want to use the user account that is running the application for updates.
- Set **Use other credentials for updates**, if you want to use another user account and enter the domain, user name, and password for the user.

- Installation progress is displayed on the **Setup is running** page. Once installation is complete, click **Next**.

The Web Installer generates the web application and the corresponding configuration files (`web.config`) for each folder.

- Click **Finish** on the last page to end the program.

Configuring the SPML web service

The SPML web service configuration is found in the `web.config` XML file in the installation directory. You can use any text editor to edit this file.

NOTE:

- After the default installation, make any changes required to the `AuthenticationString` in the `configuration\application`.
- Generate the `QOIM_Schema.xsd` and `QOIM_Spm1TargetSchema.xsd` schema files with the Designer's Schema Editor. For more information, see [Creating schema files](#) on page 498. Save the schema files to the SPML web service directory and declare the storage location of the schema files in the configuration file using the `ProviderSchema` and `Spm1TargetSchema` options. The files are saved by default to the `Schemas` directory in the installation directory.
- If the SPML web service should only be available over an encoded SSL connection, configure this in the Internet Information Services setting for each respective application. Look at your Internet Information Services documentation for further information.

Table 192: Configurable options in the “web.config” configuration file

Section	Option	Permitted values	Description
connectionString			Database connection parameter.
runtime	key="Cache"	value = "<path>"	Directory for storing the cache directory. Default: value="C:\inetpub\wwwroot\<web service name>\App_Data\Cache\DB"
	key="AssemblyCache"	value = "<path>"	Directory for storing the cache directory. Default: value="C:\inetpub\wwwroot\<web service name>\App_Data\Cache\Assemblies"
application	key = "ProviderSchema"	value = "<path>"	Relative path to SPML schema (QOIM_Schema.xsd). The schema defines all objects and properties the can be administered using the web service. The file is created by the Designer. All requests made to the web service are verified against this file. Default: value=". \Schemas\QOIM_Schema.xsd"
	key = "SpmlTargetSchema"	value = "<path>"	Relative path to the SPML target schema (QOIM_SpmlTargetSchema.xsd). The schema defines the response to the listTargetsRequest. The file is created by the Designer. Default: value=". \Schemas\QOIM_SpmlTargetSchema.xsd"
	key = "MaxConnections"	value = "<Integer>"	Number of possible simultaneous connections Number of clients Default: value ="1"
	key = „AuthenticationString"	value="Module;User;Password="	Authentication module and login data for carrying out login and all operations of the web service. Standard:

Section	Option	Permitted values	Description
			value="Module=DialogUser;User=DIALOGUSER;Password=PASSWORD"
	key = "DebugMode"	value = "True" value = "False"	Extended data in the log. Default: value="true"
	key = "LogAllRequests"	value = "True" value = "False"	Always log queries. Default: value="false"
	key = "LogDirectory"	value = "<path>"	Log directory. Default: value=".\\Log"
	key = "MaxSearchResults"	value = "<Integer>"	Maximum number of search results permitted for the iteration. Default: value="10000"
	key = "ConcurrentSearchResponseObjects"	value = "<Integer>"	Number of objects per iteration that may be returned to the client by the search operation. Default: value="10"
	key = "CheckForUnusedResultsInterval"	value = "<Integer>"	Interval in seconds for scanning orphaned search results. Default: value="30"
	key = "KeepSearchResultsFor"	value = "<Integer>"	Interval in seconds the client has to iterate the result set before it is discarded. Default: value="60"

NOTE: Use `aspnet_regiis.exe` to encrypt the connection parameter (ConnectionString).

Calling example:

```
c:\windows\Microsoft.NET\Framework\v4.0.30319\aspnet_regiis.exe -pe
"application" -app "/<web service name>" -prov
"DataProtectionConfigurationProvider"
```

where: <web service name> = web service path on the Internet Information Services


Uninstalling the SPML Web Service

To uninstall a web application

1. Launch `autorun.exe` from the root directory of the One Identity Manager installation medium.
2. On the start page of the installation wizard:
 - a. Change to the **Installation** tab.
 - b. In the **Web-based components** pane, click **Install**.

This starts the Web Installer.

3. On the Web Installer start page, click **Uninstall a web application** and click **Next**.
4. On the **Uninstall a web application** page, double-click the application that you want to remove.

The  icon is displayed in front of the application.
5. Click **Next**.
6. On the **Database connection** page, select the database connection and authentication method and enter the corresponding login data.
7. Click **Next**.
8. Confirm the security prompt with **Yes**.
9. The uninstall progress is displayed on the **Setup is running** page.
10. Once installation is complete, click **Next**.
11. On the **Wizard complete** page, click **Finish**.
12. Close the `autorun` program.

Configuring the One Identity Manager schema

The SPML web service supports the entire One Identity Manager schema. It is necessary to define the objects and properties to be managed as well as the relations in the One Identity Manager schema in order to manage objects and their relations using the SPML web service. The SPML web service cannot be used until the objects and properties as well as references have been defined in the One Identity Manager schema as being managed with SPML. After the definition has been made, two schema files are created that are needed for validation by the SPML web service. The files should be exchanged in the appropriate SPML web service directory.

Detailed information about this topic

- [Preparing the One Identity Manager schema for export to the SPML schema](#) on page 497
- [Creating schema files](#) on page 498

Preparing the One Identity Manager schema for export to the SPML schema

For administration of objects with individual properties and of relations between different object types with SPML web service, label the corresponding tables, columns, and table relations of the One Identity Manager schema to be exported to the SPML schema.

To manage objects and their properties with the SPML web service

1. In the Designer, select the **One Identity Manager schema** category.
2. Select the table and start the Schema Editor with the **Show table definition** task.
3. On the **Table** tab, enable the **Export for SPML schema** option.
4. Select the column in Schema Editor.
5. On the **Miscellaneous** tab, enable the **Export for SPML schema** option.

NOTE: If references between different One Identity Manager schema object types are managed with the SPML Web Service, both of the affected objects for SPML administration must be marked. Therefore, both tables must be labeled with the **Export for SPML schema** option.

References between object types are mapped by foreign key relations and many-to-many assignments in One Identity Manager.

- It is sufficient to mark the corresponding column in the One Identity Manager schema with the **Export for SPML schema** option in order to manage foreign key relations with SPML.

NOTE: Note that only one foreign key relation can be managed between two object types using SPML. Thus the business role manager (Org.UID_PersonHead) can be maintained with SPML, but not at the same time as the deputy manager (Org.UID_PersonHeadSecond).

- For the configuration of many-to-many relations for use with SPML, select the respective many-to-many tables and label the table relation with the **Export for SPML schema** option.

Related topics

- [Table definitions](#) on page 54
- [Table relations](#) on page 99

Creating schema files

Once you have labeled all tables, columns, and table relations that should be managed using SPML, you need to create the necessary schema file for SPML web service.

IMPORTANT:

- Before exporting, ensure that you have committed all the changes in the Designer in the main database and that all open calculation tasks for the DBQueue Processor have been processed.
- If you change other SPML-relevant settings on the One Identity Manager schema at a later date, you must recreate the schema file.

To create a schema file

1. In the Designer, start One Identity Manager in the **Schema Editor**.
2. Select the **Schema > Export SPML schema information** menu item.
3. Confirm the security prompt with **OK**.
4. In the **Browse for folder** dialog, enter the directory where the schema files will be created.
5. Click **OK**.

This starts the export. The export can take some time depending on the number of changes.

6. Click **OK**.

Place the QOIM_Schema.xsd and QOIM_Spm1TargetSchema.xsd schema files in the SPML web service directory. Enter the storage location for the schema files in the SPML web service configuration file. The files are saved by default to the Schemas directory in the installation directory.

Related topics

- [Configuring the SPML web service](#) on page 493

Testing SPML web service functionality

A simple test front-end is supplied in order to test the basic functionality of SPML web service. Prerequisite for using the test front-end is that SPML web service is correctly installed and configured. Use a browser to check whether SPML web service is functioning and correctly installed.

The SPML web service can be reached over a browser under:

http://<server>/<application name>

https://<server>/<application name>

TIP: You can open the web server's status display in the Job Queue Info. In the Job Queue Info, select **View > Server state** in the menu and, on the **Web servers** tab, open the web server status display from the **Open in browser** context menu.

Detailed information about this topic

- [Configuring the SPML test front-end](#) on page 499
- [Using the SPML test front-end](#) on page 499

Configuring the SPML test front-end

This configuration setting provides the specific URL of the SPML web service for use in the SPML test front-end.

1. On the installation medium, copy the `VI.SPMLTestFrontend.exe` and `VI.SPMLTestFrontend.exe.config` files from the `QBM\dvd\AddOn\SPML\Testfrontend` directory into the One Identity Manager installation directory.
2. To declare the configured web service in the test front-end, change the `VI.SPMLTestFrontend.exe.config` file.
3. You can then start the `VI.SPMLTestFrontend.exe` file.

Using the SPML test front-end

It is possible with the SPML test front-end to test and analyze SPML web service functionality. The front-end is used exclusively to analyze the SPML web service and test the functionality.

NOTE: Long term usage of the front-end for controlling the SPML Web Service is not planned.

To test the SPML web service functions

1. Start the SPML test front-end using the `VI.SPMLTestFrontend.exe` file.
2. Select the SPML web service function to test from the **Choose Request** list.
The corresponding XML queries are displayed in the **SPML Request (XML)** text field.
You can edit the XML request before sending it to the SPML Web Service. Always check the predefined section of the XML request and modify the schema defined in the target system for SPML support. The predefined sections are supposed to provide help for formulating SPML compliant requests.
3. Set the **Increment request IDs automatically** option if the request ID passed to the XML queries should be incremented. This option is disabled by default and the given request ID is used.

4. Send the query to the SPML web service using the **Access** button.

The result is displayed in the **SPML Response (XML)** text field.

If a new object is added in the target system, its key is added to the **Known Objects** list. An iterator is returned if a search is carried out with a limited result set and the result list cannot be returned in its entirety. This iterator is added to the list of known iterators (**Known Iterators**). The search can be continued with this iterator.

You will find detailed error messages in the log file. This is stored in the directory that you specified in the LogDirectory option in the SPML web service configuration file.

Related topics

- [Configuring the SPML web service](#) on page 493

Processing DBQueue tasks

The tasks queued in the DBQueue are the result of triggering, modifications to configuration parameters (for example, changes to a configuration parameter concerning inheritance) or running scheduled tasks. The DBQueue Processor processes tasks in the DBQueue. The DBQueue Processor uses several slots for running tasks in parallel.

You can select which agent type takes over the processing of internal tasks when you use the Configuration Wizard to install or update a database.

- **Database Agent Service:** Internal tasks are processed by the Database Agent Service. Ensure that the Database Agent Service is installed and configured. The Database Agent Service is deployed through the One Identity Manager Service plugin.

IMPORTANT: This is an EXPERIMENTAL function. The performance impact on production systems has not been determined. Therefore this feature is not yet covered by support. However, you are welcome to try it (preferably in non-production systems) and if you have any feedback, send it to OneIM.Beta@oneidentity.com.

- **SQL Server Agent:** The internal tasks are processed by the SQL Server agent. This creates the required database schedules. Ensure that the SQL Server agent is started.

Detailed information about this topic

- [DBQueue Processor configuration for test, development, or productive environments on page 502](#)
- [Configuring notification behavior for DBQueue Processor initialization on page 503](#)
- [Reinitializing the DBQueue Processor on page 503](#)
- [Bulk processing in the DBQueue Processor on page 504](#)
- [Processing DBQueue Processor tasks by the SQL Server Agent on page 505](#)
- [Processing DBQueue Processor tasks by the Database Agent Service on page 506](#)
- [Controlling processing of DBQueue Processor tasks on page 507](#)
- [Processing DBQueue Processor tasks on page 508](#)
- [How the central dispatcher communicates with individual slots on page 510](#)

DBQueue Processor configuration for test, development, or productive environments

You use the staging level of the One Identity Manager database to specify whether the database is a test database, development database, or a live database. A number of database settings are controlled by the staging level.

If you change the database's staging level, the following settings are configured.

Table 193: Default settings for development, test, and live environments

Setting	Development environment	Test environment	Live environment
Maximum DBQueue Processor runtime	20 minutes	40 minutes	120 minutes
Maximum number of slots for the DBQueue Processor	5	7	Maximum number of slots according to the hardware configuration

The DBQueue Processor default configuration settings are configured for normal operation and do not normally need to be modified.

If several databases are operating in a managed instance in the Azure SQL Database, you can fix the number of slots. In the Designer, adjust the following configuration parameters.

- **QBM | DBServerAgent | CountSlotAgents:** Exact number of slots. If the configuration parameter is set, the given number of slots are always set up. There is no internal calculation of the number of slots based on the hardware configuration. Changing the server's configuration has no effect. The value **15** is recommended.

NOTE: This configuration parameter is not recommended for implementing a database on an SQL Server. For implementing a database on an SQL Server, it is standard practice to use the hardware configuration to determine the slots.

The configuration settings are reduced for test environments and development environments because several databases may be located on a server. If it is necessary to change the settings for test environments or development environments for reasons of performance, you must modify the following configuration parameter settings in the Designer.

- **QBM | DBQueue | CountSlotsMax:** Maximum number of slots to be used.

Use this configuration parameter to reduce the number of slots if required. Values lower than **5** are not permitted.

Exception: Enter a value of **0** for using the maximum number of slots available based on the hardware configuration.

- **QBM | DBQueue | KeepAlive:** Maximum runtime of the central dispatcher. Tasks on slots currently in use are still processed when the timeout expires. Then the slot are stopped and the central dispatcher exits.

The lowest permitted value for runtime is **5 minutes**; the maximum permitted value is **720 minutes**.

Related topics

- [Changing the database staging level](#) on page 38

Configuring notification behavior for DBQueue Processor initialization

If errors occur during initialization of the DBQueue Processor, messages are written to the application log. You can use the results display in the Microsoft Management Console, for example, to view the application log.

Use the **QBM | DBServerAgent | CreateNotification** configuration parameter to configure in which cases error messages are written to the application log. In the Designer, you can modify the configuration parameter as required.

Permitted values are:

- **0:** No logging.
- **1:** Only success messages are logged.
- **2:** Only error messages are logged.
- **3:** All messages are logged.

Reinitializing the DBQueue Processor

IMPORTANT: Select a user that you use for migrating the database to run the SQL queries.

- You must run the QBM_PDBQueuePrepare procedure once manually when the server hardware has been extended and when custom DBQueue Processor tasks have been created.
- You must run the QBM_PDBQueuePrepare and QBM_PWatchDogPrepare procedures once when you set up a reference database for test and development.

Use a suitable program for running SQL queries to run the following procedures in the reference database just once.

```
exec QBM_PWatchDogPrepare
```

```
exec QBM_PDBQueuePrepare 0,1
```

Bulk processing in the DBQueue Processor

Table 194: Configuration parameter for bulk processing in the DBQueue Processor

Configuration parameter	Meaning
QBM DBQueue DefaultRuntime	The configuration parameter species how the length of the DBQueue Processor run. The default value is 90 seconds.
QBM DBQueue ChangeLimitMin	The configuration parameter defines the lower limit for modifications (insert, change, or delete) within a single operation. The default value is 3000 .
QBM DBQueue ChangeLimitMax	The configuration parameter defines the upper limit for modifications (insert, change, or delete) within a single operation. The default value is 50000 .

Some DBQueue Processor procedures are marked for bulk processing to reduce the total time required for processing DBQueue tasks. If a lot of entries are marked for bulk processing in the DBQueue, the DBQueue Processor switches from single to bulk processing.

There is a mechanism implemented that is used to decide whether switching to bulk processing as opposed to single processing would result in time savings. To do this, 25 single task processes are run and the processing time is recorded. All other entries for the task are processed in bulk and the minimum and maximum load time required for advantageous bulk processing is defined. A self optimizing calculation procedure updates the load times. Use of this method means that the DBQueue Processor must first stabilize, especially after an initial schema installation or after system modifications such as memory expansion in the database server.

You can use the **QBM | DBQueue | DefaultRuntime** configuration parameter to specify the length of the DBQueue Processor run. The default value is **90** seconds. This corresponds to the time period that achieves the best load for the calculation procedure.

To prevent overloading when there is large amount of data, you can define limits for the result set. Control is realized using the **QBM | DBQueue | ChangeLimitMin** and **QBM | DBQueue | ChangeLimitMax** configuration parameters.

Processing DBQueue Processor tasks by the SQL Server Agent

IMPORTANT: Do not change or delete predefined database schedules as it may lead to unexpected errors.

To process internal tasks by the SQL Server Agent, the DBQueue Processor is initialized during schema installation. The following database schedules are generated during the initialization phase:

- **QBM_PWatchDog on <database>**

This database schedule takes over several functions in One Identity Manager.

- It checks whether DBQueue Processor's central dispatcher is active and restarts it.
- It controls validation and starting of schedules.
- It starts a database schedule to remove complete processes from the DBQueue.
- It starts the database schedule for maintenance work.
- It checks at regular intervals, whether database single-user mode is still required and resets the setting if necessary.

This database schedule has an active schedule with a 1 minute interval.

- **QBM_PDBQueueProcess_Main on <database>**

This database schedule is the DBQueue Processor's central dispatcher. The central dispatcher assumes control of processing and distributes DBQueue tasks to individual slots. Each time the central dispatcher is run, the number of currently available slots required for the current run is found. The central dispatcher starts database schedules for the currently available slots just once.

Only one database schedule at most is started for the central dispatcher. The central dispatcher's database schedule does not have an active schedule, but is started by the **QBM_PWatchDog on <database>** database schedule.

- **QBM_PDBQueueProcess<SlotNumber> on <database>**

The maximum number of available slots is determined during the DBQueue Processor initialization phase.

- In the case of productive databases, the number of maximum slots depends on the number of processors on the database server.
- For test environments and development environments, the maximum number of slots is reduced.
- If the **QBM | DBServerAgent | CountSlotAgents** configuration parameter is set, the exact number of the specified slot is always set up. There is no internal calculation of the number of slots based on the hardware configuration.

For more information, see [DBQueue Processor configuration for test, development, or productive environments](#) on page 502.

An associated **QBM_PDBQueueProcess<SlotNumber> on <database>** database schedule is set up for each slot. Each database schedule is set up with a process that runs the DBQueue tasks for exactly this slot. The database schedules associated to each slot do not have any active schedules. They are started by the central dispatcher.

- **QBM_PDBQueueProcess_Del on <database>**

This database schedule removes processed DBQueue tasks. The database schedule does not have an active schedule, but is started through the **QBM_PWatchDog on <database>** database schedule.

- **QBM_PDBQueueProcess_Mnt on <database>**

This database schedule processes maintenance tasks. The maintenance tasks pass your tasks on to the database schedule instead of running them themselves. This means that nothing changes on the scheduling of maintenance tasks. The database schedule does not have an active schedule, but is started through the **QBM_PWatchDog on <database>** database schedule. For detailed information about maintenance tasks, see the *One Identity Manager Operational Guide*.

Related topics

- [Controlling processing of DBQueue Processor tasks](#) on page 507
- [Processing DBQueue Processor tasks](#) on page 508
- [How the central dispatcher communicates with individual slots](#) on page 510
- [Processing DBQueue Processor tasks by the Database Agent Service](#) on page 506

Processing DBQueue Processor tasks by the Database Agent Service

To process internal tasks by the Database Agent Service, ensure that the Database Agent Service is installed and configured. The Database Agent Service is deployed through the One Identity Manager Service plugin. The plugin should be configured on the Job server that performs the **Update server** server function. An administrative user must be used for the database connection in the Job provider.

The Database Agent Service carries out the following tasks:

- Controls processing and distribution of the DBQueue tasks to the individual slots (central dispatcher))
- Checks and starts scheduled tasks
- Removes already processed DBQueue tasks
- Starts maintenance work

- Populates and updates the Job queue overview
- Archives or deletes handled processes from the Job queue
- Archives or deletes process handling logs

NOTE: If the Database Agent Service is not working, a message is displayed in the status bar in all the administration tools. To see this message, users must have at least the configuration user access level.

Related topics

- [Controlling processing of DBQueue Processor tasks](#) on page 507
- [Processing DBQueue Processor tasks](#) on page 508
- [How the central dispatcher communicates with individual slots](#) on page 510
- [DatabaseAgentPlugin](#) on page 314
- [Processing DBQueue Processor tasks by the SQL Server Agent](#) on page 505

Controlling processing of DBQueue Processor tasks

The central dispatcher assumes control of processing and distributes DBQueue tasks to individual slots.

First, it determines the number of currently available slots available for use. The more load there is on the database, the less slots there available for use. However, at least five slots are used.

The number of currently available slots results from:

The number of currently available slots = maximum number of available slots - sum of all own database processes - sum of processes of other databases on the server

NOTE: The number of available slots can still be influenced by the **QBM | DBQueue | CountSlotsMax** configuration parameter. If the number of available slots, according to calculation, is more than the value in the configuration parameter, the configuration parameter value is used. For more information, see [DBQueue Processor configuration for test, development, or productive environments](#) on page 502.

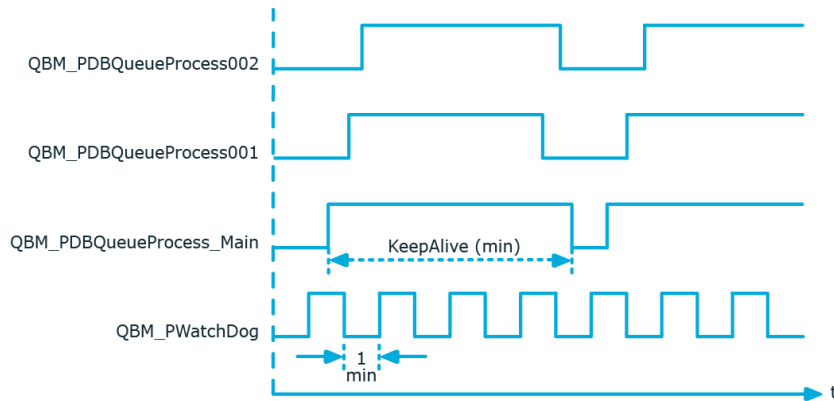
The central dispatcher starts the currently available slots just once. or each slot, a process is set up that runs tasks for exactly this slot.

Once tasks in the DBQueue are entered, the central dispatcher is notified. The central dispatcher distributes tasks to individual slots and notifies the slot processes that there are tasks are waiting to be processed. Each process processes the tasks queued for its slot. Once the task is complete, each process sends a message to the central dispatcher and waits for new tasks.

The central dispatcher checks at defined intervals whether the slots are still active and distributes new tasks to them. If there are no more tasks in the DBQueue, the central dispatcher goes into a wait state and waits for new task notifications.

Tasks on slots currently in use are still processed when the timeout expires. On completion, the slots are stopped. For more information, see [How the central dispatcher communicates with individual slots](#) on page 510.

Figure 39: Using the SQL Server Agent to control processing



Processing DBQueue Processor tasks

The central dispatcher finds entries in the DBQueue (DialogDBQueue table) and moves the tasks into the QBMDBQueueCurrent table with the assignment tasks per slot.

Example of entries in the DialogDBQueue and QBMDBQueueCurrent tables

Table 195: Entries in the DialogDBQueue (extract) table

Task name	Object
OrgRoot	A
OrgRoot	B
ADSAccountInADSGroup	X
ADSAccountInADSGroup	Y
ADSAccountInADSGroup	Z

Table 196: Entries in the QBMDBQueueCurrent (extract) table

Slot number	Task name	Object
001	OrgRoot	A
001	OrgRoot	B
002	ADSAccountInADSGroup	X
002	ADSAccountInADSGroup	Y
002	ADSAccountInADSGroup	Z

Each process processes tasks queued for its own slot in the QBMDBQueueCurrent table. Subsequent tasks resulting from processing are queued in the DialogDBQueue table.

If a process has processed its tasks and no other tasks are pending, the slot number in the QBMDBQueueCurrent table is set to **0** by the process itself. The entry initially remains in the QBMDBQueueCurrent table but is no longer taken into account (because slot 0 is not active).

All entries with the slot number 0 are deleted from the QBMDBQueueCurrent table at regular intervals.

Table 197: Meaning of slot numbers in the QBMDBQueueCurrent table

Slot number	Meaning
001 - n	Number of slot to be processed by the task.
0	State after the task is completed correctly.
-1	An error occurred during task processing or processing was deferred, for example, because synchronization is running. The central dispatcher re-enables the task. NOTE: Deferring DBQueue tasks is recorded in the system journal.
-2	An error occurred during task processing or processing was deferred, for example, because of blocking. The central dispatcher re-enables the task.
-3	An error occurred during task processing or processing was deferred, for example, because there are still entries in the Job queue. The central dispatcher re-enables the task.

Using the DBQueue buffer

To prevent blockages when processing DBQueue tasks by lengthy actions, for example, synchronization, a DBQueue buffer (QBMDBQueuePond table) is used. Synchronization initially writes DBQueue Processor tasks to the QBMDBQueuePond table. After synchronization is complete, the task are moved from the QBMDBQueuePond table to the DialogDBQueue table.

If a lengthy task does not queue anymore entries in the DBQueue tasks because, for example, synchronization did not end correctly, the remaining entries in the QBMDBQueuePond table are moved to the .DialogDBQueue table. The time period for this is

defined in the **QBM | DBQueue | BufferTimeout** configuration parameter (default: **120** minutes). The transfer is carried out by the daily maintenance tasks.

How the central dispatcher communicates with individual slots

The QBMDBQueueSlot table is responsible for communication of the central dispatcher with individual slots. The maximum number of slots available is determined during initializing of the DBQueue Processor. One entry per slot is created in the QBMDBQueueSlot table. The table contains information about each slot and its status as well as currently running tasks.

Table 198: Meaning of status in the QBMDBQueueSlot table

Status	Meaning
0	No activity required. Initial state (set by initializing) or end state (set by process).
1	The process is triggered to prepared central temporary tables, for example.
2	Ready for operation. The process has started but the currently no tasks exist. This is the state in which tasks can be queued.
3	Transfer to the QBMDBQueueCurrent table. The process has received tasks for processing and needs to begin.
4	The process has recognized the tasks and added them.
5	The process is handling the tasks.
-1	The process was prompted to quit. Stop behavior if the process timed out or errors occurred.

Stop behavior by maximum timeout

Once the maximum runtime has expired, the tasks of slots in the QBMDBQueueCurrent table currently in use are still processed. No new tasks are added from the QBMDBQueue table. In the QBMDBQueueSlot table, all slots with a slot status **2** are set to the **-1** status. This prompts the processes to finish and stop themselves. The central dispatcher checks whether all processes have completed.

Related topics

- [Example of communication during processing on page 511](#)
- [DBQueue Processor configuration for test, development, or productive environments on page 502](#)

Example of communication during processing

The following example show the entries in the QBMDBQueueSlot table during processing.

- Slot initialization

Slot number	Status	Task name
001	0	

- Starts the process using the central dispatcher.

Slot number	Status	Task name
001	1	

- The process is ready for operation. Preparations, for example, for temporary tables, are complete. The slot status is regularly tested.

Slot number	Status	Task name
001	2	

- The central dispatcher distributes tasks. The central dispatcher checks slots for readiness and enters the task from DialogDBQueue table in the QBMDBQueueCurrent table with the slot number. The status of each slots is updated once the QBMDBQueueCurrent table has taken over.

Slot number	Status	Task name
001	3	OrgRoot

- The process recognizes a task on the basis of the status, starts processing and updates its slots' status.

Slot number	Status	Task name
001	4	OrgRoot

- The process has completed the processing and sets slot number in the DialogDBQueueCurrent table to **0**. The process changes the status of its slots to operational.

Slot number	Status	Task name
001	2	

One Identity Manager Service configuration files

One Identity Manager Service is configured using a configuration file. The configuration file has to be in the same directory as the `viNetworkService.exe`. Two configuration files are supported:

Detailed information about this topic

- [Jobservice.cfg](#) on page 513
- [viNetworkService.exe.config](#) on page 514

Jobservice.cfg

The file `Jobservice.cfg` is an XML configuration file in One Identity Manager's own simplified format. The advantage of this file is that run-time loading is supported. There is a configuration section in the file for each of the different modules in the One Identity Manager Service.

NOTE: Entries are case-sensitive. Both the sections and the names of the values must be written in lower case.

The root in the XML file is always called `configuration`. Each configuration file module and its values are defined in a `category` section respectively. At the moment the program only supports the `System.Configuration.NameValueSectionHandler` section type.

```
<configuration>
  <category name="serviceconfiguration">
    <value
      name="jobprovider">VI.JobService.MSSqlJobProvider,jobservice</value>
    <value name="HttpPort">1180</value>
    <value name="logwriter">VI.JobService.FileLogWriter,jobservice</value>
  </category>
</configuration>
```

Example

Simple configuration with:

- Direct connection to a SQL Server
- Only one Job destination (JobProcessor)

```
<configuration>
  <category name="serviceconfiguration">
    <value
      name="jobprovider">VI.JobService.MSSqlJobProvider,jobservice</value>
    <value name="logwriter">VI.JobService.FileLogWriter,jobservice</value>
  </category>
  <category name="sqlprovider">
    <value name="connectstring">User ID=sa;initial Catalog=<Database>;Data
      Source=<SQL-Server>;Password=<Password></value>
  </category>
  <category name="filelogwriter">
    <value name="loglifetime">0.01:00:00</value>
    <value name="logseverity">Info</value>
  </category>
  <category name="dispatcher" />
  <category name="jobdestinations">
    <value
      name="queuex">VI.JobService.JobServiceDestination,jobservice</value>
  </category>
  <category name="queuex">
    <value name="queue">\\%COMPUTERNAME%</value>
  </category>
</configuration>
```

Related topics

- [viNetworkService.exe.config](#) on page 514

viNetworkService.exe.config

The `viNetworkService.exe.config` file is the default configuration file for .NET exes and has the specified format. There is a configuration section in the file for each of the different modules in the One Identity Manager Service.

NOTE: Entries are case-sensitive.

The root in the XML file is always called configuration. All other sections of the configuration file must be in the mandatory configSections section and their type must be defined. At the moment the program only supports the System.Configuration.NameValueSectionHandler section type.

```
<configuration>
  <configSections>
    <section name="sectionname"
      type="System.Configuration.NameValueSectionHandler" />
  </configSections>
  <sectionname>
    ...
  </sectionname>
</configuration>
```

Example

Simple configuration with:

- Direct connection to a SQL Server
- Only one Job destination (JobProcessor)

```
<configuration>
  <configSections>
    <section name="serviceconfiguration"
      type="System.Configuration.NameValueSectionHandler" />
    <section name="sqlprovider"
      type="System.Configuration.NameValueSectionHandler" />
    <section name="filelogwriter"
      type="System.Configuration.NameValueSectionHandler" />
    <section name="dispatcher"
      type="System.Configuration.NameValueSectionHandler" />
    <section name="jobdestinations"
      type="System.Configuration.NameValueSectionHandler" />
    <section name="queuex"
      type="System.Configuration.NameValueSectionHandler" />
    <section name="plugins"
      type="System.Configuration.NameValueSectionHandler" />
  </configSections>
  <serviceconfiguration>
```

```

        <add key="jobprovider"
        value="VI.JobService.MSSqlJobProvider,jobservice" />
        <add key="logwriter" value="VI.JobService.FileLogWriter,jobservice" />
    </serviceconfiguration>
    <sqlprovider>
        <add key="ConnectionString" value="User ID=sa;initial
        Catalog=<Database>;Data Source=<SQL-Server>;Password=<Password>" />
    </sqlprovider>
    <filelogwriter>
        <add key="LogLifeTime" value="0.01:00:00" />
        <add key="LogSeverity" value="Info" />
    </filelogwriter>
    <dispatcher />
    <jobdestinations>
        <add key="QueueX"
        value="VI.JobService.JobServiceDestination,jobservice" />
    </jobdestinations>
    <queuex>
        <add key="queue" value="\%COMPUTERNAME%" />
    </queuex>
</configuration>

```

Related topics

- [Jobservice.cfg](#) on page 513

One Identity solutions eliminate the complexities and time-consuming processes often required to govern identities, manage privileged accounts and control access. Our solutions enhance business agility while addressing your IAM challenges with on-premises, cloud and hybrid environments.

Contacting us

For sales and other inquiries, such as licensing, support, and renewals, visit <https://www.oneidentity.com/company/contact-us.aspx>.

Technical support resources

Technical support is available to One Identity customers with a valid maintenance contract and customers who have trial versions. You can access the Support Portal at <https://support.oneidentity.com/>.

The Support Portal provides self-help tools you can use to solve problems quickly and independently, 24 hours a day, 365 days a year. The Support Portal enables you to:

- Submit and manage a Service Request
- View Knowledge Base articles
- Sign up for product notifications
- Download software and technical documentation
- View how-to videos at www.YouTube.com/OneIdentity
- Engage in community discussions
- Chat with support engineers online
- View services to assist you with your product

#

#LD notation 354

\$

\$ notation 344
data type 344

*

.CustomForms.vif 150
.Forms.vif 153

A

application server 18
connection data 270
search index
update 87
AppServerJobProvider 294
AuthenticationString 294
ConnectionString 294
RequestQueueLimit 294
ResultQueueLimit 294
assignment form 149
configuration data 156
tab 156
authentication module
default 33

C

collection 22

column

#LD content 212
alternative primary key 91
assign to employee 84
base column 91
bitmask 86, 91
compulsory field 80, 91
data type 91
default configuration
change 28
restore 28, 30
default value 91
display name 91
dynamic foreign key 91, 102
edit 74, 90
allow 30
lock 30
encrypted 91
export for SPML 91, 497
foreign key 91
format 91
formatting script 83
formatting types 81
group 91, 146
hierarchy data 91
index weighting 87, 91
language dependent 212
length 91
log 91
mapping 74
maximum length 80, 91

- minimum length 65, 80, 91
- multi-value specification 86, 91
- multiline 91, 146
- multilingual 91, 212
- MVP column 86, 91
- name components 91
- no DB transport 91
- no log 91
- number of decimal places 80, 91
- permitted value 84
- preprocessor condition 91
- primary key 91
- properties 74
- proxy view 91
- recipient 75
- recursive key 91
- sender 75
- sort order 91, 146
- syntax 91
- table 91
- table lookup support 91
- template 75, 91
- text buffer 214
- threshold 91
- translate 212
- translation source 212
- translation target 212
- unique 65
- value list 84, 91
- column relation 99
- Common Table Expressions 447
- compiling
 - conditional 335
 - error message 244
- configuration file 283
- configuration parameter
 - create 41
 - deactivate 42
 - display name 42
 - edit 40
 - enable 40, 42
 - encrypted 42
 - option 43
 - preprocessor condition 336
 - preprocessor Expression 43, 336
 - preprocessor relevant 42
 - value 40, 42
- configuration repository 28
- configurations
 - ComponentDebugMode 302
 - DebugMode 302
 - DoNotProtectCryptedValues 302
 - DoNotProtectPrivateKeys 302
 - DoNotWriteConfigBack 302
 - HTTPAddress 302
 - HTTPHeaders 302
 - HTTPPort 302
 - language 302
 - LogDestinationAndProviderId 302
 - RetriesOnFailedStart 302
 - SecretAllowList 302
 - SecretFolder 302
 - UseSSL 302
 - VerboseLogging 302
 - WaitTimeOnFailedStart 302
- connection
 - CacheReloadInterval 309
 - CacheType 309
 - DirectConnection 309
 - JobGenLogDir 309

- LogBlobReads 309
- NoReloadBeep 309
- ObjectDumpStackExpression 309
- SupportReadScaleOut 309
- TokenCertificateFile 309
- TokenCertificateThumbprint 309

country

- edit 205
- enable 203
- language 205, 207
- public holiday 205, 207
- time zone 205, 207
- work hours 207

country information 203

- country 205, 207
- public holiday 204, 209
- state 206, 208
- time zone 202

Customizer 18, 74

D

data change

- delete 331
- record 318-319
- retention period 332

database

- authentication module 33
- connection data 33, 36
- country 33, 39
- customer name 33
- customer prefix 33
- database ID 33
- development database 37, 502
- file groups 104
- language 33

- main database 33
- production database 37, 502
- public key 33
- revision 33
- SQL customization 456
- staging level 33, 37, 502
- test database 37, 502

Database Agent Service 314, 506

database layer 18

DatabaseAgentPlugin

- provider ID 314

DBQueue 18

DBQueue Processor 501

- bulk processing 504
- central dispatcher 505, 510
- communication 510
- configurations 502
- database schedule 505
- duration 502, 504
- GenProcID 325
- initialization 505
- process monitoring 325
- process task 508
- processing 507
- QBM_PDBQueuePrepare 503
- QBM_PDBQueueProcess 505
- QBM_PDBQueueProcess_Del 505
- QBM_PDBQueueProcess_Main 505
- QBM_PDBQueueProcess_Mnt 505
- QBM_PWatchDog 505
- QBM_PWatchDogPrepare 503
- slots 502, 505, 510

DebugMailPlugin 313

- DropFolder 313

deferred deletion 22

- dialog notification 75
- DialogColumn 44
- DialogMultiLanguage 211
- DialogTable 44
- dispatcher

- IsProxy 308
 - ProxyIntervall 308

E

- encryption

- PrivateKey 295

- entity 18

- Delayed Logic 22
 - discard 22
 - EntityLogic 18
 - EntitySource 18, 22
 - interactive 22
 - MarkForDeletion 22
 - read only 22
 - UnitOfWork 18, 22

- event

- assign 238
 - delete 238
 - edit 238-239
 - insert 238
 - object event 238-239
 - permission 240
 - process information 239, 323
 - program function 238-240
 - remove 238
 - run 238
 - sort order 239
 - update 238

- EventLogLogWriter 305
 - category 305

- EventID 305
- EventLog 305
- LogSeverity 305
- source 305

F

- file groups 104

- FileJobDestination 297

- AutoUpdateSubDirectories 297
 - BackupFiles 297
 - CheckInputIndex 297
 - EventTypes 297
 - host name 297
 - InputDirectory 297
 - MaxListCount 297
 - OutputDirectory 297
 - port 297
 - provider ID 297
 - SubDirectories 297
 - TimerInterval 297
 - UseEncryption 297

- FileJobProvider 289

- AutoSubDirectories 289
 - BackupFiles 289
 - CheckInputIndex 289
 - EventTypes 289
 - host name 289
 - InputDirectory 289
 - MaxListCount 289
 - OutputDirectory 289
 - port 289
 - SubDirectories 289
 - TimerInterval 289
 - UseEncryption 289

- FileLogWriter 306
 - AddServerName 306
 - HistorySize 306
 - JobLogLifeTime 306
 - LogLifeTime 306
 - LogSeverity 306
 - MaxLogSize 306
 - OutPutFile 306
 - ParamMaxLength 306
- foreign key column 49
 - dynamic 49
- form archive 153
- form definition 151, 153
 - configurations 155-156
 - form sequence 155
 - required tables 155
 - VI_Common_ChildRelation_Grid 147
 - VI_ElementNavigation 147, 166
 - VI_Generic_MasterData 146-147
 - VI_Report 147, 433
 - VI_Wizard 147
- Form Editor 141
- form template 153
 - form archive 153
 - form source type 153
 - form type 153
- FrmCommonChildRelationGrid 147
- FrmCom-
 - monOneChildAndMemberRelation 147
 - FrmCommonOneChildRelation 147
 - FrmCommonOneDynamicRelation 147
 - FrmCommonOneGenericRelation 147
 - FrmCommonOneMemberAndChildRelation 147
 - FrmCommonOneMemberRelation 147
 - FrmElementNavigation 147
 - frmGeneric 147
 - ReportForm 147
 - usage 153
 - WizardForm 147
- form type 153
 - edit 153
 - grid 153
 - info 153
 - MemberRelation 149, 153
 - report 153
 - virtual 153
 - wizard 153
- formatting script 83
 - test 374
- formatting types 81
- FTP Server 291
- FTP user 291
- FTPJobDestination 299
 - AutoUpdateSubDirectories 297
 - BackupFiles 297
 - CheckInputIndex 297
 - EventTypes 297
 - FTPPassword 299
 - FTPPort 299
 - FTPServer 299
 - FTPUser 299
 - host name 297
 - InputDirectory 297
 - MaxListCount 297
 - OutputDirectory 297
 - port 297
 - provider ID 297

- SubDirectories 297
- TimerInterval 297
- UseEncryption 297
- FTPJobProvider 291
 - AutoSubDirectories 289
 - BackupFiles 289
 - CheckInputIndex 289
 - EventTypes 289
 - FTPPassword 291
 - FTPPort 291
 - FTPServer 291
 - FTPUser 291
 - host name 289
 - InputDirectory 289
 - MaxListCount 289
 - OutputDirectory 289
 - port 289
 - SubDirectories 289
 - TimerInterval 289
 - UseEncryption 289
- full text search
 - application server 87
 - configure 87

G

- GenProcID 317
 - replacement 325
- Globally Unique Identifier (GUID) 49, 64
- GUID module 64

H

- HTTP authentication module
 - BasicHttpAuthentication 310
 - WindowsHttpAuthentication 310

- HTTP Server 302
- HTTPJobDestination 301
 - ChildPort 301
 - provider ID 301
 - RemoteDomain 301
 - RemotePassword 301
 - RemoteUser 301
 - retries 301
 - RetryDelay 301
- HTTPJobProvider 293
 - ParentPort 293
 - ParentServer 293
 - RemoteDomain 293
 - RemotePassword 293
 - RemoteUser 293
 - retries 293
 - RetryDelay 293
- HTTPLogPlugin
 - log file 311
 - LogFile 311

I

- info system
 - bar chart 179
 - line diagram 179
 - pie chart 179
 - statistic definition 171
 - statistics 176
 - table 179
 - tachometer 179
 - thermometer 179
 - traffic light 179
- input value
 - define 132
- InstallState.config 265

IsChanged 349

IsDeleted 349

IsLoaded 349

J

Job queue 18

queue 277, 295

Job server

configure 284

connection data 270, 272

create 262

edit 261-262

executing server 263

fallback connection 270

Machine role 267

no direct database connection 270,
272

queue 262-263

server function 262-263, 266

server operating system 263

service account 263

start HTTP request 284

statistics 269

status 284

transfer configuration 284

Job Service Configuration

module type 286

verification test 287

JobDestination 279

FileJobDestination 297

FTPJobDestination 299

HTTPJobDestination 301

Jobgate 279

JobGenLogDir 309

Jobprovider 279

AppServerJobProvider 294

FileJobProvider 289

FTPJobProvider 291

HTTPJobProvider 293

MSSQLJobProvider 288

Jobservice.cfg 283, 513

JobServiceDestination

EncryptionScheme 295

ExternalSlotEnvironment 295

ExternalSlotEnvironment32 295

ExternalSlots 295

ExternalSlots32 295

InternalSlots 295

MaxExternalSlotReuse 279, 295

PrivateKey 295

PrivateKeyId 295, 314

provider ID 295

queue 295

RequestTimeout 295

StartInterval 295

StatisticInterval 295

K

key file 295

L

language 208

country 205, 207

state 206

language pack

import 219

language setting

default language 38

- login language 38
- Launchpad 187
 - action 189-190
 - extend 187
 - menu item 190
 - NavigationNodeState 187-188
- link
 - set up 133
- lists
 - condition 130
 - display pattern 130, 132
 - edit 130
 - input value 132
 - insert values 130
 - object 130
 - sort by 130
- LogWriter
 - EventLogLogWriter 305
 - FileLogWriter 306

M

- Machine role 265, 267
- many-to-many table 68
- MarkForDeletion 22
- menu item
 - assign application 124
 - assign permissions group 125
 - assign user interface form 146
 - condition 125, 128
 - configuration switch 125
 - copy 116, 122
 - create 116, 121, 123
 - data dependent 113, 128
 - data source 128
 - database query 128

- deactivate 125
- diagram type 176, 179
- display text 125
- edit 116
- fixed 113
- free 113
- icon 125
- item type 125
- Launchpad 190
- link 113, 133
- lists 130
- load 118
- main form element 113
- menu category 113, 116, 121, 123
- overlay icon 125, 187-188
- preprocessor condition 125
- recursive 128, 130
- sort by 128
- sort order 125
- statistics 113, 176
- task 113
- task category 113
- unique 128
- variable definition 135, 137
- method definition
 - behavior 193
 - deactivate 192-193
 - display text 193
 - edit 191
 - enabled for 193
 - icon 193
 - name 193
 - object 193
 - object definition 191
 - permission 191

- permissions group 191, 193
- program function 191, 193
- script 193
- test 375

MSSQLJobProvider 288

- ConnectionString 288
- FlushTimeoutSeconds 288
- RequestQueueLimit 288
- ResultQueueLimit 288

N

navigation

- copy 120
- create 121
- load 116-117, 119
- select 116

NavigationNodeState class 187-188

O

object

- assign 22
- base. 350
- change 22, 25
- deferred 22
- delete 22, 25
- discard 22
- EntitySource 22
- handling 22
- insert 22, 25
- interactive 22
- IsChanged 349
- IsDeleted 349
- IsLoaded 349
- lifecycle 22

- load 22
- marked for deletion 22
- MarkForDeletion 22
- remove 22
- transaction 25
- UnitOfWork 22
- update 22
- XMarkedForDeletion 22

object class 22

object definition

- condition 108, 111
- display name 111
- display pattern 111
- display text 111
- display text (form) 109
- display text (list) 109
- edit 110
- input value 132
- insert values 111
- object 111
- preprocessor condition 111
- selection script 108, 111
- table 111
- universal 111

object event 238-239

- permission 240
- program function 238-240

object layer 18

One Identity Manager

- data model 44
- software architecture 18
- system configurations 33
 - configuration parameter 40
 - database connection 36
 - documentation 31

- enable country 203
 - language setting 38
 - report 31
 - time zone 202
- One Identity Manager query
 - language 381
- block comment 382
- comment 382
- compare columns 394
- compare date differences 396
- compare date ranges 396
- compare fixed values 397
- compare parameters 398
- condition 392-393
- date value 384
- decimal value 384
- display 390
- display value 390
- identifier 382
- in clause 395
- integer value 383
- language elements 381
- line comment 382
- literals 383
- LongDisplay 390
- not in clause 395
- order by clause 389
- paging 389
- parameter 385, 398
- predefined where clause 385, 398
- query header 386
- query info 392
- query parameter 391
- search clause 387
- select clause 387
- skip clause 389
- string value 383
- take clause 389
- time value 384
- where clause 387
- One Identity Manager schema 44
 - schema overview 45
- One Identity Manager Service
 - AppServerJobProvider 294
 - configuration file 283, 513-514
 - configurations 302
 - template 285
 - configure 283-284
 - connection 309
 - DatabaseAgentPlugin 314
 - DebugMailPlugin 313
 - dispatcher 308
 - event log 305
 - EventLogWriter 305
 - FileJobDestination 297
 - FileJobProvider 289
 - FileLogWriter 306
 - FTPJobDestination 299
 - FTPJobProvider 291
 - generation log 309
 - HTTP authentication 310
 - HTTPJobDestination 301
 - HTTPJobProvider 293
 - HTTPLogPlugin 311
 - install 272
 - JobDestination 294
 - JobServiceDestination 295
 - language 302
 - linux share 313
 - log file 306

- LogWriter 305
- MSSQLJobProvider 288
- PerformanceCounterPlugin 312
- plugins 310
- procedure 276-279
- process collection 288
- process component 276
- queue 262-263
- remote install 272
- RequestWatchDogPlugin 312
- ScheduleCommandPlugin 311
- set up 276
- ShareInfoPlugin 313
- statistics info 295
- StdioProcessor.exe 279, 295
- StdioProcessor.log 279
- One Identity Manager Service module
 - configurations 302
 - connection 309
 - dispatcher 308
 - HTTP authentication module 310
 - JobDestination 294
 - LogWriter 305
 - plugins 310
 - private key files 314
 - process collection 288
- overview form 162
 - alignment 167
 - background color 167
 - columns 163, 167
 - create 163
 - deactivate 169
 - delete 170
 - delete form element 170
 - design view 163
 - disable form element 169
 - display text 163
 - form definition 166
 - form element 163, 166
 - header 167
 - insert form element 165
 - lists 166-167
 - main form element 163, 166
 - menu item 163
 - object 163
 - OverviewNode 166
 - permissions group 163
 - preview 163, 169
 - product assignment 163
 - VI_ElementNavigation 166
- Overview Form Editor
 - design view 163
 - preview 163, 169

P

- PerformanceCounterPlugin 312
 - category 312
 - CounterType 312
 - PollingInterval 312
- permissions group
 - copy 120
- plugins
 - DatabaseAgentPlugin 314
 - DebugMailPlugin 313
 - HTTPLogPlugin 311
 - PerformanceCounterPlugin 312
 - RemoteConnectPlugin 313
 - RequestWatchDogPlugin 312
 - ScheduleCommandPlugin 311
 - ShareInfoPlugin 313

- preprocessor condition 337-338
 - column 91
 - configuration parameter 336
 - evaluate 339
 - menu item 125
 - object 111
 - process 230
 - process step 231
 - report 406
 - statistics 172
 - table 68
 - user interface forms 151
- preprocessor Expression 336
- presentation Layer 18
- primary key column 49
- process
 - base object 230
 - compare 229
 - compile 244
 - copy 224-225
 - create 224
 - do not generate 230
 - edit 224
 - error checking 242
 - event 238-239
 - export 229
 - generating condition 230
 - identifier 230
 - import 229
 - notification 247
 - over limit 247
 - pre-script for generating 230, 351
 - preprocessor condition 230
 - process information 230, 322
 - simulation 241
 - table 230
 - threshold 230, 247
 - UID 230
 - validity check 242
 - variable 245
- process automation 251
- process component 254, 276, 295
 - AutoUpdateComponent 254
 - CommandComponent 254
 - ControlFilesComponent 254
 - DelayComponent 254
 - display name 258
 - FileComponent 254
 - FtpComponent 254
 - HandleObjectComponent 254
 - LogComponent 254
 - MailComponent 254
 - maximum instance 257-258
 - PowerShellComponent 254
 - process task 258
 - ProjectorComponent 254
 - ReportComponent 254
 - ScriptComponent 254
 - SQLComponent 254
 - WakeOnLanComponent 254
 - ZipComponent 254
- Process Editor
 - layout position 222
 - process document 222
 - process element 222
 - process step element 222
- process generating
 - simulate 241
 - test 241
- process handling 220

- process history
 - delete 331
 - record 324
 - retention period 332
- process information
 - delete 331
 - depth of detail 322
 - for process 322
 - for process step 322
 - for result 323
 - record 320
 - retention time 332
- process monitoring 316-317
 - data change 318-319
 - DBQueue Processor 325
 - delete 331
 - enable 317
 - process handling 320
 - process information 320
 - process trigger 317
 - retention period 332
- process parameter template 258
- process plan 251
 - base object 253
 - calculation schedule 253
 - condition 253
 - event 253
 - parameter 253
 - run 252-253
 - set up 252-253
 - status 251
- process step
 - copy 226-227
 - create 226
 - edit 226
- generating condition 231
- identifier 231
- ignore errors 231
- import 226-227
- log error 231
- log mode 231
- notification 231, 249
- parameter 234
 - edit 235
 - encrypted 235
 - hidden 235
 - identifier 235
 - IN 235
 - INOUT 235
 - OUT 235
 - out parameter 236
 - Parametercollection 236
 - set value 236
 - template 258
 - type 235
 - value template 235
- pre-script for generating 231, 351
- preprocessor condition 231
- priority 231
- process information 231, 322
- process task 231
- retention time 231
- retries 231
- script for server selection 231, 248
- search 228
- server 248
- server function 231, 248
- split processing 231
- stop on error 231
- wait mode on error 231

- process task 254
 - exe type 256, 258
 - identifier 258
 - maximum instance 257-258
 - operating system class 258
 - process component 258
- program function
 - object event 238
- proxy server 308
- public holiday
 - country 205, 207
 - edit 204, 209
 - state 206, 208
- PWatchDogPrepare 503

Q

- QBM_PDBQueuePrepare 104, 503
- QBM_PDBQueueProcess 505
- QBM_PDBQueueProcess_Del 505
- QBM_PDBQueueProcess_Main 505
- QBM_PDBQueueProcess_Mnt 505
- QBM_PDiskStorePhysicalSync 104
- QBM_PTableMove 104
- QBM_PWatchDog 505
- QBM_PWatchDogPrepare 104
- QBMRelation 44
- QBMTranslationAddOnSource 214
- QERCentralAccount 91
- QERMailAddress 91

R

- RemoteConnectPlugin 313
 - ADGroupAuthPermittedGroup 313
 - AuthenticationMethod 313

- port 313
- report
 - base table 406
 - bind 433
 - copy 404
 - create 404
 - data field 426
 - data query
 - historical assignment 415
 - multiple object history 413
 - object 410
 - simulation 417
 - single object history 411
 - SQL 408
 - test 407
 - view 409
 - data source
 - delete 407
 - edit 407
 - virtual 425
 - display name 406
 - edit 404
 - export 425, 434
 - identifier 406
 - import 425
 - load 404
 - multilingual 432
 - preprocessor condition 406
 - repo 425
 - report parameter 418
 - can be overwritten 421
 - condition (calc.) 424
 - condition (query) 422
 - data source 422
 - data type 422

- default value 422
 - delete 420
 - description 421
 - display name 421
 - edit 420
 - list of permitted values 422
 - mandatory parameter 421
 - multiline 422
 - multivalue 422
 - overwrite empty value 422
 - parameter definition 422
 - parameter type 421
 - parameter value 422
 - script 424
 - sort order 421
 - table column (calc.) 424
 - table column (query) 422
 - value calculation 424
 - viewable 421
 - ReportAlias 406
 - ReportName 433
 - ReportParameter 433
 - translate 425, 432
 - user interface forms 433
 - Report Editor 399
 - globalization editor. 425
 - program setting 402
 - SQL log 403
 - RequestWatchDogPlugin 312
 - interval 312
 - MinRequests 312
- S**
- ScheduleCommandPlugin 311
 - command 311
 - interval 311
 - LogSeverity 311
 - OutputToLog 311
 - StartCommand 311
 - StopCommand 311
 - Schema Editor
 - schema overview 45
 - schema extension
 - add 453
 - assignment table
 - index column 450
 - authorizations 452
 - change label 452
 - column
 - base column 442
 - column name 439-440, 442
 - comment 443
 - compulsory field 443
 - configure 443
 - create 439-440
 - data type 443
 - define 438
 - delete restrictions 443
 - display name 443
 - foreign key column 440
 - from table 440
 - initial value 443
 - insert restrictions 443
 - name 443
 - permission 443
 - primary key 436, 443
 - simple column 439
 - UID 443
 - unicode 443
 - x-column 436

- create index 450
- create table 436
- custom 435
- database view
 - column 442
 - create 445, 447
- DDL statements 453
- extend table 438
- foreign key column
 - column name 440
 - create 440
 - from table 440
- Index 450
 - index name 450
- permissions group 452
- read-only database view
 - description 445
 - display name 445
 - name 445
 - view definition 445
- remove 450
- save to file 453
- Schema Extension 435
- simple column
 - column name 439
 - create 439
- table
 - create 436
 - description 436
 - display name 436
 - extend 438
 - name 436
 - primary key 436
 - x-column 436
- union view
 - description 447
 - display name 447
 - name 447
 - view definition 447
- Schema Extension 435
- SCIM plugin 468
- script 341
 - #LD notation 354
 - \$ notation 344
 - accessibility 359
 - auto-completion 359
 - base. 350
 - compile 368
 - copy 366
 - create 365
 - data type 344
 - date value 342
 - edit 365, 372
 - function 350
 - message 342
 - overridable 368
 - override 368
 - overwrite 368
 - permission 369
 - pre-script 351
 - program function 369
 - RaiseMessage 342
 - save 376
 - Session Services 351
 - syntax 341
 - test 367, 370, 373
 - VID_Write2Log 342
 - web service call 462

- script library 358
 - load 370
- semaphore 28
- server
 - specify 248
- server function 265-266
- session
 - Session.Config.GetConfigParm 352
 - Session.Source.Exists 353
 - Session.Variables 353
- Session Services 351
- ShareInfoPlugin 313
- SOAP Web Service 474
 - ChangeSingleObject 474, 482
 - ChangeSingleObjectEx 474, 482
 - configuration file 479
 - configure 479
 - CreateSingleObject 474, 482
 - DeleteSingleObject 474, 482
 - DeleteSingleObjectEx 474, 482
 - exists 474, 482
 - ExistsEx 474, 482
 - FireGenEvent 474, 482
 - FireGenEventEx 474, 482
 - GetCompleteSingleObject 474, 482
 - GetCompleteSingleObjectEx 474, 482
 - GetListObject 474, 482
 - GetListObjectWithDisplays 474, 482
 - GetSingleObject 474, 482
 - GetSingleObjectEx 474, 482
 - GetSingleProperty 474, 482
 - GetSinglePropertyEx 474, 482
 - install 476
 - InvokeCustomizer 474, 482
 - InvokeCustomizerEx 474, 482
 - InvokeDialogMethod 474, 482
 - InvokeDialogMethodEx 474, 482
 - status display 481
 - web.config 479
- software architecture 18
- SPML Provisioning Service Provider 488
- SPML schema
 - configure 497
 - export 498
- SPML Web Service 488
 - configuration file 493
 - configure 489
 - install 489
 - schema file 498
 - test 498
 - web.config 493
- Standard GUID 64
- state
 - edit 206, 208
 - enable 208
 - language 206, 208
 - public holiday 206, 208
 - time zone 206, 208
 - work hours 208
- statistic
 - traffic light 179
- statistics 171
 - aggregate function 172
 - bar chart 179
 - calculation 172
 - calculation schedule 172
 - condition 174
 - deactivate 175
 - diagram type 179

- display name 172
- edit statistics definition 171
- ElementName 174, 179
- ElementObjectKey 174
- ElementObjectKey2 174
- ElementOrder 174
- ElementValue 174, 179
- history 172
- imported statistics data 172
- instant calculation 172
- line diagram 179
- measurement value 174
- not set 172
- pie chart 179
- preprocessor condition 172
- query 174
- table 179
- tachometer 179
- thermometer 179
- threshold 172, 174
- use report 177-178
- StdioProcessor.exe 279, 295
- SwitchToModuleGuid() 64
- SwitchToNormalGuid() 64
- System Debugger 370
 - SQL log 372

T

- table
 - assign by event 68
 - base table 54, 68
 - cache information 68
 - condition for transport 68
 - Customizer 68
 - deferred deletion 67-68

- display name 68
- display pattern 68
- edit 54, 68
- export for SPML 68, 497
- file group 104
- GUID module 68
- hierarchy path 68, 149
- icon 68
- ignore empty values 65
- input value 132
- logical storage 68, 104
- M-to-all table 49
- many-to-many table 49, 68
- mapping 54
- module GUID permitted 64
- module GUID required 64
- multicolumn uniqueness 65, 68
- physical storage 104
- preprocessor condition 68
- properties 54
- proxy 54, 57
- proxy view 68
- read-only 54, 61
 - Common Table Expressions 447
- retain in memory 68
- scope hierarchy 68
- simple table 49
- statistic information 68
- table 54
- table lookup support 68
- table scripts 63, 68
- type 49, 54
- union 54, 59
- unique group 65
- usage type 68

- view 54-55
- view definition 68
- worktable 49
- table relation 99
 - only transport as group 99
 - update dependencies modification date 99
- table scripts 63, 68
 - test 375
- task
 - deactivate 192
- template
 - assign to employee 48
 - cross-object 79
 - edit 76
 - limit implementation 78
 - local 79
 - overwritten 76
 - prevent change 77
 - recipient 75
 - sender 75
 - shorten value 76
 - test 374
 - thresholds 78
- time zone 202
 - country 205, 207
 - state 206, 208
- transaction 25
- translation 210
 - basics 211
 - change entry key 218
 - column 212
 - edit 214, 216-217
 - fallback 214
 - import 219

- language pack 219
- text buffer 214
- translation source 212
- translation target 212

U

- use case
 - authentication module 196
 - configuration data 196
 - form 196
 - menu item 196
 - permissions group 196
 - set up 195-196
 - start menu item 196
 - system user 196
- user interface
 - form 138
 - icons 198
 - images 198
 - method definition 191
 - navigation 112
 - object definition 108
 - statistics 176
 - translation 210
 - use case 195-196
- user interface forms
 - assign application 143
 - assign menu item 146
 - assign object definition 144
 - assign permissions group 144
 - configurations 151
 - copy 142
 - create 142
 - custom 146-147
 - deactivate 141, 151

- display text 151
- edit 140, 151
- form definition 151
- form name 151
- form preview 141
- generic form 146
- input value 132
- insert values 151
- main data form 146
- navigation structure display 146
- object definition 145
- overview form 162
- preprocessor condition 151
- replace 150
- repo 433
- sort order 151
- tab 146

V

variable

- define 135
- DialogUserID 135, 353
- EnvUserName 135, 353
- Feature_ 353
- FullSync 353
- GenProcID 353
- LogonUser 135, 353
- ManageOutstandingOperation 353
- pre-script 245
- process definition 245
- SessionType 135, 187-188
- ShowCommonDialog 135, 353
- UserName 135, 353
- UserID 135, 353

VI.DB.DLL 18

- viNetworkService.exe 283
- viNetworkService.exe.config 283, 514

W

web service

- change proxy code 466
- create script 462
- integrate 458
- method call
 - direct 461
 - generic 459
 - self-defined 461
- script parameter 462
- SOAP 462
- synchronization by script 459
- WCF 462
- WSDL file 462

web service integration wizard 462

X

- XDateInserted 49
- XDateSubItem 49
- XDateUpdated 49
- XIsInEffect 49
- XMarkedForDeletion 22, 49
- XObjectKey 49
- XOrigin 49
- XTouched 49
- XUserInserted 49
- XUserUpdated 49