One Identity Manager 8.2

API Designer User and Development Guide

Contents

API Designer

The API Designer allows you to create, record, compile, and publish a REST API (Representational State Transfer Application Programming Interface) in the quickest way possible. This API is based on the OpenAPI Specification and the One Identity Manager database model.

The main benefits of API Designer include:

- Offers easy and fast operation.
- The finished API "understands" the One Identity Manager database model.
- Modifications to the API are transparent.
- Supports the principles of good API design.
- OpenAPI support: APIs that you create using the API Designer are based on the OpenAPI specification standard. This allows you to make use of other tools, including:
  - Swagger: Use Swagger to create code, documentation, and test cases.
  - Postman: Use Postman to test the various methods of your API.

Basic principles of API development

The main components of an API created using the API Designer are files and projects.

Basic information on API files:

- Use API files to define API methods for sending data to the application or to querying data from the application.

- An API file can belong to more than one project.

- Because the API Server is stateless, save the API methods that you specify in API files, without a client specific state.
  For example, you cannot, therefore, define global variables or store session object status data. When the API Server processes are restarted, these values are not restored.

- Use asynchronous code for defining API methods. This supports more efficient usage of server resources and improves performance of the system under load. The methods of the API and the underlying object model convert this asynchronousity using the Task-based Asynchronous Pattern (TAP). For more information about TAP, see https://docs.microsoft.com/de-de/dotnet/standard/asynchronous-programming-patterns/task-based-asynchronous-pattern-tap.

- Do not use the `HttpContext.Current` method when you define API methods. You can query the current HTTP requirements with the `QBM.CompositionApi.ApiManager.Context.Current` static method.

The basics of API projects:

- API projects combine multiple API files into logical sections. The API project includes the configuration.

Basic information about the API Server:

- The API server is implemented using the Owin Platform (see http://owin.org/).

- Access to current HTTP requirements is not supported by ASP.NET APIs.

- After enabling routes, you cannot change the definition objects anymore.

Related topics

- Quick start – Creating an API on page 21

Guidelines and conventions

In this chapter, you will find general policies and conventions, which you must take into account when you create an API.

Related topics

Handling of API Server queries

In this section, you will find information about handling queries that are sent to the API server.

Authentication

When a query is sent to the API Server, there is a test to ascertain the success of the primary and, possibly, secondary login in the session for the current project (see Authentication on page 10).

NOTE: This test is not done if the API method used by the query is marked as **AllowUnauthenticated**.

The **imx-session-<API project name>** cookie is evaluated to allocate the current session.

If a cookie is passed that cannot be associated with an active session in the current process, the security token in the cookie is used to set up a new session (see Session status and security tokens on page 12).

If there is no primary login, the API Server tries to establish a database connection through one of the enabled single sign-on authentication modules.

If login cannot be carried out, the process is canceled and the HTTP error code **500** is passed to the client (see Response codes on page 14).

Authorizing method access

The API Server checks whether the currently logged in user is authorized to run the method. If the user does not have the required permissions, the process is canceled and the HTTP error code **500** is passed to the client (see Response codes on page 14).

Validating the query

The API Server calls the validators stored with the API method one by one. If one fails, the process is canceled and the HTTP error code **400** is passed to the client (see Response codes on page 14).

Processing queries (for entity methods)

- GET (for loading entities)
  - Determines the WHERE clause with internal and external filters
  - Loads data from the database
  - Enriches entities with calculated columns
- Entities in delayed logic mode can be changed with a POST query or deleted with a DELETE query. Entities in this mode are stateless and do not occupy resources on the server after the query has been processed.
  Supported HTTP methods:
  - GET (for loading entities)
  - POST (for changing entities)
  - DELETE (for deleting entities)
- Interactive entities must be created once with a PUT query and after that they obtain their own ID. Use the ID in subsequent queries (POST or DELETE).

  Supported HTTP methods:
  - PUT (for creating interactive entities)
  - POST (for changing interactive entities)
  - DELETE (for deleting interactive entities)

Authentication

User authentication is carried out on the API Server for each API project.

Running an API method requires prior authentication on an API project. If the API method is marked as `AllowUnauthenticated`, authentication is not required (you can find an example in the SDK)

Authentication has two steps:

1. Required primary authentication: Default authentication through an authentication module

2. Optional secondary authentication: Multi-factor authentication (for example, using Starling 2FA)

Related topics

- Handling of API Server queries on page 9
- Authentication (primary) on page 11
- Authentication with Starling 2FA (secondary) on page 11
- Logging out on page 12

Authentication (primary)

You can use the **imx/login/<API project name>** API method for primary authentication on the API project.

To do this, use the POST HTTP method to send the following query:

{ "Module": "RoleBasedPerson", "User": "<user name>", "Password": "<password>" }

TIP: See our SDK for examples.

Related topics

- Authentication on page 10
- Authentication with Starling 2FA (secondary) on page 11
- Logging out on page 12

Authentication with Starling 2FA (secondary)

Secondary authentication on the API project using Starling 2FA uses the following methods:

- Login/authentication using push notifications: **starling/send/push**
- Login/authentication using SMS messages: **starling/send/sms**
- Login/authentication using phone calls: **starling/send/call**

Related topics

- Authentication on page 10
- Authentication (primary) on page 11
- Logging out on page 12

Logging out

You can use the **imx/logout/<API project name>** API method to log out of the API project.

Related topics

- Authentication on page 10
- Authentication (primary) on page 11
- Authentication with Starling 2FA (secondary) on page 11

Session status and security tokens

The status a session is saved in a cookie. This cookie contains an encrypted security token which is used to restore a login to the API Server if the API Server was restarted in the mean time. The security token is cryptographically signed by the certificate selected on installation.

NOTE: If the API Server's current user restarts the browser, the cookie and its session information are reset.

Related topics

- Querying session status on page 12

Querying session status

You can use the **imx/sessions/<API project name>** API method to query the status of the session. The response contains the following information:

- Permitted authentication module and associated parameters of the respective API project.
- Type of secondary login (for example Starling 2FA).

Related topics

- Session status and security tokens on page 12

HTTP methods

HTTP requests can apply the following HTTP methods:

- **Get**: This method requests data from the application server.
- **Put**: This method changes data on the application server.
- **Post**: This method creates data on the application server.
- **Delete**: This method deletes data on the application server.

Date formats

Date values must provided in ISO 8601 format in UTC in queries that change or add objects.

Example

2016-03-19T13:09:08.123Z

Related topics

- Parameter formats on page 13

Parameter formats

HTTP requests use the following types of parameters:

- URL parameters
- Query parameters

Related topics

- Date formats on page 13

URL parameters

URL parameters extend the URL path. A forward slash is used as the delimiter.

If a query uses a URL parameters, they are given in URI format.

Example

https://<host name>/ApiServer/imx/sessions/exampleparameter

Related topics

- Query parameters on page 14

Query parameters

Query parameter are appended to the URL with a question mark (?) or an ampersand (&).

The first query parameter must be prefixed by a question mark. In this case, you must use the following format:

?parameter name=parameter value (for example, ?orderBy=LastName)

Subsequent query parameters must be prefixed by an ampersand. In this case, you must use the following format:

&parameter name=parameter value (for example, ?sortOrder=ascending)

Example

https://<host name>/AppServer/portal/person?orderBy=LastName

Related topics

- URL parameters on page 13

Response formats

Most API methods return results in JSON format (application/json). Results returned in CSV and PDF formats are also supported if the result of the API method is declared as exportable.

To obtain results in CSV format

- In the query, set **Accept header** to **text/csv**.

To obtain results in PDF format

- In the query, set **Accept header** to **applciation/pdf**.

  NOTE: To obtain results in PDF format, the **RPS** module must be installed on your system.

Related topics

- Response codes on page 14

Response codes

Responses that are sent from the REST API use the following codes. If queries fail, an explanatory error message is displayed.

Response codes	Description
200	Query successful.
204	Query successful. Response has no content.
401	Access not authorized. The session must be authorized first.
404	The given resource could not be found.
405	The HTTP method used is not allowed for this query.
500	A server error occurred. The error message is sent with the response. On the ground of security, a detailed error message is not included in the response. For more information, see the application log file on the server.

Related topics

General notes and information about entity methods

In this section, you will find advice and information for creating and implementing entity methods.

Limiting results

NOTE: Entity-based methods normally work with a limit to avoid unintentionally loading extremely large amounts of data.

The following query parameters help you to limit the amount of data that is returned by obtaining multiple data sets from sequential responses:

Query parameter	Default value	Description
PageSize	20	Specifies the maximum number of data sets that can be contained in the response. If you only determine the total number and do not want to obtains single data sets, use the value **-1**.
StartIndex	0	Specifies as from which data sets the results are returned in the response. This parameter is null-based (the first element is addressed with the value **0**).

Example

The following query returns 50 employees and starts with the 101st employee:

`https://<Host-Name>/ApiServer/portal/person?PageSize=50&StartIndex=100`

Sort order

Use the **Orderby** query parameter to sort the results returned in an response. This parameter allows you to sort the column names of the underlying database table.

Examples

The following query returns employees sorted by first name in ascending order:

`https://<Host-Name>/ApiServer/portal/person?OrderBy=FirstName`

Employees sorted by first name in descending order:
`https://<Host-Name>/ApiServer/portal/person?OrderBy=FirstName%20DESC`

Filtering

Use the **filter** query parameter to filter the results returned in an response. A filter like this consists of a JSON formatted string that must contain the following:

- **ColumnName**: Name of the column used to filter
- **CompareOp**: The operator for comparing the contents of the selected column with the expected value

  The following comparison operators are permitted:

  - **Equal**: The results only include data sets with column data that matches the comparison value.
  - **NotEqual**: The results only include data sets with column data that does NOT match the comparison value.
  - **LowerThan**: The results only include data sets with column data less than the comparison value.
  - **LowerOrEqual**: The results only include data sets with column data less than or equal to the comparison value.
  - **GreaterOrEqual**: The results only include data sets with column data greater than or equal to the comparison value.
  - **Like**: Requires the use of a percent sign (%) as a placeholder. You can enter up to two percent signs in this value. The results only include data sets with column data that matches the comparison value pattern.

- **NotLike**: Requires the use of a percent sign (%) as a placeholder. You can enter up to two percent signs in this value. The results only include data sets with column data that does NOT match the comparison value pattern.
- **BitsSet**: The value is compared to the comparison value using the AND (&) logical operator. The result must not be equal to 0.
- **BitsNotSet**: The value is compared to the comparison value using the AND (&) logical operator. The result must be equal to 0.

- **Value1**: Comparison value for comparing the contents of the column
- **Value 2**: If this second comparison value is passed down, the value of **CompareOp** is ignored and all the values that are greater or equal to **Value1** and less or equal to **Value2** are determined.

Example

The following query returns all employees with the last name "Smith":

`https://<Host-Name>/ApiServer/portal/person/all?filter=[{ColumnName: 'LastName', CompareOp: 'Equal', Value1: 'Smith'}]`

Grouping

NOTE: The API method must support grouping (by using the **EnableGrouping** parameter).

Use the **group** URL parameter to group the results returned in an response. Use the **by** query parameter to specify which attribute to use for grouping.

The result of the query contains a filter condition that you can pass to the URL parameter as filter.

Example

The following queries determine the number of employees grouped by primary location:

`https://<Host-Name>/ApiServer/portal/person/all/group?by=UID_Locality`

Response:

```
{
    {
        "Display": "(No value: Primary location)",
        "Filters": [
            {
                "ColumnName": "UID_Locality",
                "CompareOp": 0
            }
```

```
        ],
        "Count": 42
    },
    {
        "Display": "Berlin",
        "Filters": [
            {
                "ColumnName": "UID_Locality",
                "CompareOp": 0,
                "Value1": "c644f672-566b-4ab0-bac0-2ad07b6cf457"
            }
        ],
        "Count": 12
    }
}
```

Hierarchical data structure

Some data model tables are defined as hierarchical structures (**Department** for example). In principle, data is loaded from such tables from a specific hierarchy level.

Use the **parentKey** query parameter and the UID of the parent object to specify the hierarchy level.

Example

https://<host name>/ApiServer/portal/servicecategories?parentKey=QER-f33d9f6ec3e744a3ab69a474c10f6ff4

Use the **noRecursive=true** URL parameter to query the data as flat list.

Example

https://<Host-Name>/ApiServer/portal/servicecategories?noRecursive=true

Related topics

- API methods on page 87

Avoiding deadlocks

API development includes a lot of asynchronous code with async/await constructs. To avoid deadlocks, use the **ConfigureAwait(false)** method for every **await** keyword.

NOTE: When you compile the API, it checks whether **ConfigureAwait(false)** is used.

For more information, see https://blog.stephencleary.com/2012/07/dont-block-on-async-code.html and https://devblogs.microsoft.com/dotnet/configureawait-faq/.

Examples and help – Software Development Kit

To make it easier for you to start developing your API with the API Designer, One Identity provides a Software Development Kit (SDK) with lots of commented code example.

The SDK can be found on the installation medium in the directory `QBM\dvd\AddOn\ApiSamples`.

Quick start – Creating an API

The following provides a general list of the steps you must take to create an API in the API Designer:

1. Start the API Designer.

2. Create API files (in which you can define an API method, for example).

3. Create an API project or edit an existing one (for example, supplied by One Identity) to take advantage of the features already available and customize it to your own needs.

4. Configure the authentication for the API project (Single Sign-On, for example).

   NOTE: If you edit an existing API project supplied by One Identity, you will no longer be able to change the authentication.

5. Assign the created API files to the API project.

6. Test the API.

7. Compile the API.

8. Save the changes to the database.

Related topics

- Starting the API Designer on page 22
- Creating API files on page 88
- Creating API projects on page 78
- Configuring authentication on page 81
- Assigning API files to an API project on page 89
- Testing an API on page 75
- Compiling an API on page 68
- Importing solution project changes into a database on page 25

Working with the API Designer

This section provides you with general information on working with the API Designer.

For example, you will learn:

- How the API Designer interface is structured
- How to edit your projects - locally or in the database itself
- How to edit database objects
- How to compile an API

Starting the API Designer

Before you start developing your API, you must start the API Designer.

To start the API Designer

1. Go to the installation path for API Designer.

2. Run the `ApiDesigner.Editor.exe` application.

   The API Designer opens.

3. Perform one of the following tasks:

   - To use an existing connection to the One Identity Manager database, select it in the **Select a database connection** menu.

   - To create a new connection to the One Identity Manager database, click **Add new connection** and enter a new connection.

4. Select the authentication method and enter the login data for the database under **Authentication method**.

5. Click **Connect**.

6. In the **Select project save type** dialog, perform one of the following tasks:

   - Click **Database project** to save the project in the database.

   - Click **Solution project** to save the project within a solution.

7. (Optional) To save the selected storage type for future projects, select the **Use selected project as default** option.

8. Click **OK**.

   The API Designer opens and displays the start page.

Related topics

- Start page on page 34
- Project types on page 23

Project types

The API Designer distinguishes between two different project types for saving your changes.

- Database project: The project will be stored in the database. Your saved changes are visible to others.
- Solution project: API Designer allows file-based editing of custom objects locally on your own computer. For more information, see Solution projects on page 24.

Related topics

- Changing the project type on page 23
- Solution projects on page 24
- Importing solution project changes into a database on page 25
- Menu bar on page 27

Changing the project type

You can save your project as a database project or as a solution project. For more information, see Project types on page 23.

To save the project as a database project

1. In the menu bar, click **Edit** > **Set project type**.

2. In the **Select the project type** dialog, click **Database project**.

3. (Optional) To save the selection and use it the next time you start the API Designer, enable the option **Use selected project as default**.

   | TIP: To undo this setting:

1. In the menu bar, click **Connection** > **Settings**.

2. Expand the **Editor** pane in the **Global settings** dialog.

3. In the **Editor** pane, select the **Ask for project type at each start** option.

   The next time the program is started, the **Select the project type** dialog is displayed.

4. Click **OK**.

To save the project as a solution project

1. In the menu bar, click **Edit** > **Set project type**.

2. In the **Select the project type** dialog, click **Solution project**.

3. Perform one of the following actions:

   - Click **Create solution** and use the wizard to create a new solution.
   - Click **Load solution** and enter the path to an existing solution.
   - Click the solution most recently used.

4. (Optional) To save the selection and use it the next time you start the API Designer, enable the option **Use selected project as default**.

   TIP: To undo this setting:

   1. In the menu bar, click **Connection** > **Settings**.

   2. Expand the **Editor** pane in the **Global settings** dialog.

   3. In the **Editor** pane, select the **Ask for project type at each start** option.

      The next time the program is started, the **Select the project type** dialog is displayed.

5. Click **OK**.

Related topics

Solution projects

API Designer allows for custom objects to be edited locally in a file-based format on your own computer. To do this you will need to export any custom objects stored within the database to the API Designer. In other words, you must use the "solution project" project type (see Changing the project type on page 23).

To import the changes made to the solution project back into the database, follow the instructions in Importing solution project changes into a database on page 25.

After exporting these objects from the database, you can edit and remove them on your computer, or supplement them with additional objects. Local object editing is no different than editing within the database project.

During export, any objects contained in the selected module are copied to the hard drive:

- API projects
- API files

Related topics

Importing solution project changes into a database

Changes that have been run within a solution project must be transferred into the One Identity Manager database.

To import object into the One Identity Manager database

1. In the menu bar, click **Edit** > **Import objects into database**.

2. In the **Import objects into database** dialog, enable the option **Remove deleted objects from database**. Objects that you have deleted from your solution project are now also deleted from the database.

3. Go to **Solution objects** to select the objects that you want to change and export into the database.

   TIP: To select or deselect listed objects quickly, enable the **Select all/deselect all** option.

4. Click **Continue**.

5. Click **Finished**.

Related topics

User interface

This section gives you an overview of the graphical user interface of the API Designer.

Use the mouse and keyboard to interact with the API Designer's graphical user interface. Use a screen resolution of at least 1280 x 1024 pixels and a minimum color depth of 16 bits for optimum display.

TIP: The API Designer view can be customized at any time to suit your needs by moving, closing, or hiding items.

Main user interface components

- **Top – Title bar**

  The title bar shows the program icon, program name, and connected database (including the current user).

- **Top – Menu bar**

  You can use the menus on the menu bar to access submenus and to access and run many functions in API Designer quickly. For more information, see Menu bar on page 27.

- **Top – Toolbar**

  The toolbar contains different icons that you can click to access other functions. For more information, see Toolbar on page 30.

- **Left – Navigation**

  Use the navigation to manage (create, open, delete) API configurations, API files, and API projects. You can also access the most recently edited file here. For more information, see Navigation on page 32.

- **Center – Work pane**

  The center pane is where other work panes for editing are shown, such as the Definition tree view. When you open objects, they are shown in the work pane in separate Tabs.

- **Bottom – Status bar**

  The status bar shows information such as database activity, project information, the connected database, and the current user. For more information, see Status bar on page 31.

Start page

The home page is displayed when the API Designer starts up.

On the home page, you can:

- Use compilation branches to store different versions of your API in the database.
- Test your API locally.
- Compile the API and then write it to the database.
- Use the API Designer to connect and edit C# projects.

For more information, see Start page on page 34.

Detailed information about this topic

Menu bar

You can use the menus on the menu bar to access submenus and to access and run many functions in API Designer quickly. The following table provides you with information about each menu.

Table 1: Menus and menu items on the menu bar.

Menu	Menu item	Description
Connection		
	Settings	Opens the **Global settings** dialog. You can configure the basic settings for API Designer here.
		For more information, see Global settings on page 34.
	Exit	Closes the program.
Edit		
	Set project type	Opens the **Select the project type** dialog. Select whether the project should be saved as a database or solution project here.
		For more information, see Project types on page 23 and Changing the project type on page 23.
	Import objects into database	Opens the **Import objects into database** dialog.
		From here you can import new and modified objects from your solution project into your One Identity Manager database.
		For more information, see Importing solution project changes into a database on page 25.
	Change standard project file	From here you can change the solution project that is to be used here.
	Find and replace	Opens the **Find and replace** dialog. You can search through documents using certain terms or strings and replace where necessary here.

Menu	Menu item	Description
		For more information, see Find and replace on page 51.
	Find next	Performs the search with the search parameters specified in the **Find and replace** dialog.
		The search continues without reopening the dialog.
	Captions	Opens the **Multilingual captions** dialog. You can create and edit multilingual captions here.
		For more information, see Multilingual captions on page 54.
	Import object	You can use this menu item to import API projects and API files into the API Designer.
		For more information, see Importing API projects on page 85 and Importing API files on page 90.
	Edit database queries	Opens the **Edit database queries** dialog. You can add, change, or delete database queries here.
		For more information, see Managing database queries on page 59.
View		
	Tabs	Opens the **Tabs** dialog. You can manage open tabs here. For more information, see Managing tabs on page 63.
	Restore standard layout	Restores the default layout. For more information, see Managing layouts on page 64.
	Restore standard layout (including size)	Restores the default layout and the default window size. For more information, see Managing layouts on page 64.
	Restore saved layout	Restores the layout that you saved earlier (see **Save layout** on the menu). For more information, see Managing layouts on page 64.
	Save layout	After you have customized the layout of the API Designer to suit your needs, you can save it and restore it at any time. For more information, see Managing layouts on page 64.
	Start page	Opens the start page.
		For more information, see Start page on page 34.
	Solution	Opens the **Solution** window. The solution can be used to display and manage any objects in the API Designer that can be opened from the navigation menu.

Menu	Menu item	Description
		For more information, see Managing objects in a project (solution) on page 40.
	Node editor	Opens the **Node editor** view. You can edit a node's properties here.
		For more information, see Node editor view on page 46.
	Tasks	Opens the **Tasks** view. Compilation errors and warnings can be viewed here.
		For more information, see Managing compilation errors and warnings (task window) on page 70.
	Command List	Opens the **Command list** window. You can view the change history of the selected database object here and undo any changes where necessary.
		For more information, see Displaying the change history (command list) on page 64.
	Compilation	Opens the **Compiling** window. You can view the compilation log here.
		For more information, see Opening the compiling log on page 72.
	Bookmarks	Opens the **Bookmarks** window. Bookmarks can be managed here.
		For more information, see Bookmarks on page 65.
	Navigation	Opens the navigation view. This can be used to manage (create, open, delete) API files, and API projects. You can also access the most recently edited file here.
		For more information, see Navigation on page 32.
Help		
	Community	Opens the One Identity forum.
	Support portal	Opens the support portal website.
	Training	Opens the training website.
	Online documentation	Opens One Identity's documentation website.
	Info	Opens a dialog. Detailed information on the API Designer is available here (system information, version number, information about the software producer, installed modules etc.).

Related topics

Toolbar

The toolbar contains different icons that you can click to access other functions.

Table 2: Toolbar functions

Icon	Description
Change Label:	In the list, select the label under which changes made to the One Identity Manager database are to be saved. Use the label to make it easier to identify and assign changes to the database. For more information, see Labeling changes on page 48.
Managing change labels	Opens the **Change label** dialog. You can select, add, change, or delete change labels here. For more information, see Labeling changes on page 48.
Use the current change label as default	Specify that the current change label see **Change label**) is to be used as the default change label. This change label will automatically be selected upon API Designer startup. This setting is bound to the client and does not affect other users of the One Identity Manager database. For more

Icon	Description
	information, see Using change labels on page 48.
🌐 Captions	Opens the **Multilingual captions** dialog. Here you can create and edit multilingual captions. For more information, see Multilingual captions on page 54.
📑 Testing a compilation	Tests the compilation for either the **Debug** or **Release** version. For more information, see Testing a compilation on page 69.
💾 Save	Saves changes to the object currently being edited. If you want to save changes to other database objects, select the corresponding tab in the definition tree view and click the button.
💾 Save all	Saves changes to all objects currently being edited.
↔ Previous/next node	Navigates backwards and forwards within the history of the selected objects. This displays the selected database object in the definition tree view.
	If an object or node has been deleted, the next object is shown. If an object that no longer exists is selected in the history, the previous object is shown.
	TIP: In the menu bar under **Connection** | **Settings**, you can specify the number of database objects shown in the history. The history can be deleted in the context menu using **Clear history**.

Related topics

- User interface on page 25
- Labeling changes on page 48
- Multilingual captions on page 54
- Testing a compilation on page 69

Status bar

The status bar displays different status data. Some status data is shown by way of icons. Which icons are displayed is partially dependent on the program settings selected. The status bar comes in different colors.

Table 3: Meaning of the colors

Color	Meaning
none	Development environment database is connected.
Red	Simulation mode is enabled.

Color	Meaning
Green	Test environment database is connected.
Yellow	Productive environment database is connected.

Table 4: Status bar icons

Icon	Meaning
👤	Shows the current user.
ℹ	Shows information about the project.
🗄✓	The database is connected.
🗄	Shows database access.
▦	The database must be compiled.
🗄	The program is in simulation mode.

On the status bar, the database is also shown in the following format:

<Server>/<Database (description)>

TIP: To copy the database path into the clipboard, double-click the database name in the status bar.

To see more information about the current user, double-click the user name in the status bar.

Related topics

- User interface on page 25

Navigation

Open the navigation using **View** > **Navigation** (see Opening the navigation on page 33).

Use the navigation to manage (create, open, delete) API files and API projects. You can also access the most recently edited file here.

At the bottom of the navigation view, you can select whether you would like to manage API files or API projects.

The following table provides an overview of the various features available within the navigation.

Table 5: Navigation features

Control	Description
⊞ Add	Adds a new database object. The type of database object changes according to the pane you are in (**API files** or **API projects**). For detailed information, see the following sections: • Creating API projects on page 78 • Creating API files on page 88
⊠ Delete	Deletes a database object. For detailed information, see the following sections: • Deleting API projects on page 80 • Deleting API files on page 89
⟳ Reload data	Refreshes the view/reloads the data.
☑ Properties	Opens the **Object properties** dialog. Allows you to view the properties for the selected database object. For more information, see Object properties on page 38.
Searching	Uses the search feature to find specific database objects. Enter the required search term and then press Enter.

Related topics

- User interface on page 25
- Creating API projects on page 78
- Creating API files on page 88
- Deleting API projects on page 80
- Deleting API files on page 89
- Object properties on page 38

Opening the navigation

By default, the navigation remains open (on the left side of the window) until you close it. You can open the navigation again at any time.

To open the navigation

- In the menu bar, click **View** > **Navigation**.

Related topics

- Navigation on page 32
- User interface on page 25

Start page

Open the start page from **View** > **Start page** (see Opening the start page on page 34).

The home page is displayed when the API Designer starts up.

On the home page, you can:

- Use compilation branches to store different versions of your API in the database.
- Test your API locally.
- Compile the API and then write it to the database.
- Use the API Designer to connect and edit C# projects.

Related topics

- Managing versions (compilation branches) on page 72
- Testing an API on page 75
- Compiling an API on page 68
- Linking C# projects to the API Designer on page 76
- Opening the start page on page 34
- User interface on page 25

Opening the start page

You can open the start page again at any time.

To open the start page

- In the menu bar, click **View** > **Start page**.

  The start page opens in the work pane.

Global settings

Open the global settings from **Connection** > **Settings** (see Changing global settings on page 37).

Use the **Global settings** dialog to configure basic settings for API Designer. The following table gives an overview of the various features within the **Global settings** dialog.

Table 6: Global settings

Pane	Setting	Description
Common		
	Language	
	Current	Displays the language currently in use for formatting data such as date, time, or numbers.
	After next restart	Specify which language the API Designer will use for formatting data such as date, time, or numbers. The next time the API Designer starts up, the API Designer will use this language. For more information, see Switching languages on page 37.
	Use another language for the UI	Specify whether you want to use a different language for the API Designer's user interface than the one configured for formatting values. Use the **After next restart** option to specify the language for the user interface. For example, you can continue using English for formatting data but have German as the display language.
	UI language	
	Current	Shows the language currently in use.
	After next restart	Specifies the language for the API Designer. The next time the API Designer starts up, the API Designer will use this language. For more information, see Switching languages on page 37.
	Resolution	Specifies the resolution.
	Maximum history length	Specifies how many changes can be undone.
Editor		
	Shorten long property values	If you would like to shorten long property values, enable this option.
	Show an error message if syntax errors are found after editing	Enable this option if you want to receive a notification when syntax errors have been found after editing.
	Save and load API Designer size within	If you would like to save the size settings for the API Designer view and reload them upon restarting, enable this option.

Pane	Setting	Description
	layout	
	Ask for project type at each start	Enable this option if you would like to select how you would like to save your project each time you start API Designer (see Changing the project type on page 23).
	Compilation branch identifier	Use compilation branches to store different API versions in the database. Manage your compilation branches. You can select the compilation branch that you want to use, display the existing compilation branches, create compilation branches, edit compilation branches, and delete compilation branches. For more information, see Managing versions (compilation branches) on page 72.
Compiler		
	Show following warnings as errors	Specify which warnings are displayed as errors during compilation. Enter the codes for the warnings, separated by commas. The error codes refer to the Microsoft C# compiler. For more information about compilation, see Compiling an API on page 68. TIP: For information about error codes that occur during compilation, you can refer, for example, to the **Error code** table in the **Tasks** view (see Managing compilation errors and warnings (task window) on page 70).
	Ignore following warnings	Specify which errors should be ignored during compilation. Enter the codes for the warnings, separated by commas. The error codes refer to the Microsoft C#Sharp compiler. For more information about compilation, see Compiling an API on page 68. TIP: For information about error codes that occur during compilation, you can refer, for example, to the **Error code** table in the **Tasks** view (see Managing compilation errors and warnings (task window) on page 70).
Save and backup		
	Ask for change label if none is selected	Enable this option if you would like to be asked for a change label when saving.
	Keep backup of unsaved objects on the local machine	If you would like to back up unsaved changes on your local hard drive, enable this option. This ensures that your changes are not lost if the program crashes.
Keyboard layout		

Pane	Setting	Description
	Show next bookmark	Specify the shortcut to be used for the next bookmark. TIP: To specify a Ctrl or Shift shortcut, enable the **Alt** and/or **Ctrl** options. Finally, select a button from the list.
	Set/delete bookmarks	Specify the shortcut you want to use to set or remove a bookmark. TIP: To specify a Ctrl or Shift shortcut, enable the **Alt** and/or **Ctrl** options. Finally, select a button from the list.

Related topics

- Changing global settings on page 37
- Menu bar on page 27

Changing global settings

You can change the global settings, general settings, or both for the API Designer at any time.

To adjust API Designer settings

1. In the menu bar, click **Connection** > **Settings**.
2. In the **Global settings** dialog, click a pane to collapse or expand it.
3. Configure the required settings (see Global settings on page 34).
4. Click **Apply**.

Related topics

- Global settings on page 34

Switching languages

The API Designer is available in German and English. You can switch between these languages at any time. Changing the language requires the API Designer to be restarted.

To change the API Designer language

1. In the menu bar, click **Connection** > **Settings**.
2. In the **Global settings** dialog, in the **Common** > **Common culture** section, select the desired language (for formatting and user interface) in the **After next restart**

menu.

3. (Optional) If you want to use another language for formatting values in the user interface, set the **Use another language for the UI** option and select the desired language in the **After next restart** menu.

> For example, if you want dates to be displayed in the German format (TT.MM.JJJJ) and the rest of the user interface in English, select **German (Germany)** in step 2 and **English (United States)** in step 3.

4. Click **Apply**.

5. Confirm the prompt with **Yes** in the **Restart API Designer dialog.**

   The API Designer will restart in the selected language.

Related topics

- Global settings on page 34

Database objects

API projects, and API files are defined as database objects in API Designer. Here you can learn how to work with these database objects in API Designer.

Related topics

- Editing database objects (definition tree view) on page 41
- Managing objects in a project (solution) on page 40
- Object properties on page 38

Object properties

Right-click on an object to open the **Object properties** dialog (see Displaying object properties on page 39)

The **Object properties** dialog displays the properties of objects in the API Designer. The dialog is divided into five tabs, which are explained below.

General

Under the **General** tab, the general properties of the object such as ID, status, or primary key are displayed.

Properties

Under the **Properties** tab, all columns in the database object are displayed in a table alongside their values. You can choose between a simple column view or the advanced view with additional data for column definitions.

Entitlements

On the **Permissions** tab, the database object permissions are displayed. The first entry shows the basic permissions for the table. The permissions for this particular object are displayed beneath it. The other entries show the column permissions.

TIP: Double-click the table entry, the object entry, or a column entry to display the permissions group from which the permissions were determined.

Table 7: Icon used for permissions

Icon	Meaning
✓	Permissions exist.
●	Permissions have been removed by the object layer.
☑	Permissions limited by conditions.

Label

Under the **Change labels** tab, all the change labels assigned to the database object are displayed. You can also assign a change label to the object.

Export

Under the **Export** tab, you can export the database object:

- To the clipboard
- As a file
- Using drag-and-drop
- As a SQL INSERT statement or SQL UPDATE statement

Related topics

- Displaying object properties on page 39

Displaying object properties

You can display the properties of a database object at any time.

To display the properties of a database object

1. In the menu bar, click **View** > **Navigation**.

2. In the navigation, click the database object whose properties you would like to display.

3. Click ⌨ (**Properties**).

   The **Object properties** dialog opens. This dialog contains several tabs displaying the properties of the selected database object. For more information, see Object properties on page 38.

Related topics

- Object properties on page 38

Managing objects in a project (solution)

Open the **Solution** view using **View** > **Solution** (see Opening the solution on page 41).

The solution can be used to display all the objects in the API Designer that can be opened from the navigation menu.

The solution view depends on your choice of project. If you select a solution project when starting the API Designer, the listed entries will be grouped according to their respective database modules and you will be able to disable the compilation of individual database modules. Database modules whose status has not changed, are shown as collapsed nodes. If you mark a subnode, you can run additional tasks.

You can also identify the objects whose status has changed in API Designer. Differences between changes in the API Designer and changes to resources in the database or on the hard disk are marked.

To open an object, double-click it in the solution.

The following table provides an overview of the various features available in the **Solution** window.

Table 8: Controls

Control	Description
🔄 Update list	Refreshes the list of database objects.
📥 Reload selected objects	Reloads the selected objects.
🗐 Mark all modified objects to be reloaded	Any objects that you have modified will be reloaded after you click 📥 **Reload selected objects**.
❌ Disable module compilation	Disables compilation of the selected module.
🔻 Only display modified objects	Only modified objects will be displayed in the list.

Table 9: Solution columns

Column	Description
Object identifier	Name of the object
DB status	**Up to date** is displayed when the database has been refreshed. Otherwise, the user who made the change is displayed.
Status drive	**Up to date** is displayed when the file has been updated. Otherwise, a time when the changes will be made is shown.
Status API Designer	Displays object modifications. The status is shown as **Changed** if the object has been changed.
Reload	A check box is displayed for new objects. You can reload selected objects.
Path	Shows storage location of the file on the hard drive.

Related topics

Opening the solution

You can open the solution at any time.

To open the solution

- In the menu bar, click **View** > **Solution**.

Related topics

Editing database objects (definition tree view)

Open the definition tree view over **View** > **Navigation > Click the required database object** (see Opening the definition tree view on page 42).

Use the definition tree view to display and edit different database objects in API Designer. A separate tab containing a unique definition tree view will be opened for each database object that you edit. A definition tree view opens when adding or editing database objects, for example.

A database object contains what are known as nodes, which you can display in a tree structure within the definition tree view and edit in the node editor view (see Node editor view on page 46). If you add a new database object to your project (for example, an API project), this database object is created with a predefined number of nodes and is shown in the object definition. The object definition is a type of view found within the definition tree view.

The predefined nodes contained in a database object form the basic structure of the database object. You cannot delete these nodes.

In addition to the predefined nodes, you can add extra nodes by right-clicking on the context menu. You can delete these nodes.

The following table gives you an overview of the various functions on the definition tree toolbar.

Table 10: Definition tree view toolbar

Icon	Description
Object definition	An overview of nodes contained in a database object can be found here.
Generated code	You can view the code generated for a selected node here. For more information, see Viewing and saving generated code belonging to a node on page 45.
Find and replace	You can search for terms here and replace them where necessary. For more information, see Find and replace on page 51.

Related topics

- Opening the definition tree view on page 42
- Viewing and saving generated code belonging to a node on page 45
- Find and replace on page 51

Opening the definition tree view

You can open the definition tree view at any time.

To open the definition tree view of a database object

1. In the menu bar, click **View** > **Navigation**.

2. In the navigation, double-click on the database object which holds the definitions you would like to display.

   The definition tree view for the selected object opens in a separate tab.

Related topics

- Editing database objects (definition tree view) on page 41
- Navigation on page 32

Context menu in the definition tree window

To access the context menu, right-click a node in the definition tree window. The options available in the menu depend on the type of node selected (context-sensitive).

Structure

The top section of the context menu lists the node types that can be added to the selected node.

Some node types do not allow additional node types to be pasted into them. In such cases, the **Object in extension** function is at the top of the context menu. All other functions are either available or disabled, depending on the type of node and its position in the definition tree window.

Table 11: Functions in the context menu

Function	Description
Object in extension	Allows you to create sub-elements of the selected node in an extension instead of an object.
Move to extension	Moves the selected node into a new or existing extension.
✂ Cut	Cuts the highlighted node and copies it to the clipboard.
	Any child nodes for the selected node are also cut.
	NOTE: You cannot cut any nodes that have been automatically added to the definition tree.
📄 Copy	Copies the highlighted and child nodes to the clipboard.
📋 Paste	Pastes the node contained in the clipboard.
	NOTE: You can only paste nodes contained in the clipboard if they are also permitted at the selected point. Example: You cannot paste a **Plugin** node below the **Compilation settings** node.
🗑 Delete	Deletes the highlighted node and all child nodes.
	NOTE: You cannot delete any nodes that have been automatically added to the definition tree.
	TIP: You can select multiple nodes by holding down the **Ctrl** key to apply the function once to all selected nodes.

Function	Description
Delete node as an extension	Deletes the highlighted node as an extension.
🔖 Set/remove bookmark	Sets or removes a bookmark on the highlighted node. For more information, see Bookmarks on page 65.
Expand all	Expands all nodes.
Collapse all	Collapses all nodes.
💾 Export	Saves the selected node and their child nodes in XML format to your hard drive.
📁 Import	Pastes previously exported child nodes at the selected point. NOTE: You can only paste nodes if they are also permitted at the selected point. Example: You cannot paste a **Plugin** node below the **Compilation settings** node.
↑ Move up	Moves the selected node upwards. NOTE: You cannot move any nodes that have been automatically added to the definition tree.
↓ Move down	Moves the selected node downwards. NOTE: You cannot move any nodes that have been automatically added to the definition tree.
↩ Undo	Undoes the previous action.
↪ Redo	Redoes the last action that has been undone.
🔍 Search object references	Searches for all references that refer to the selected node. NOTE: If searching for a reference within a collection, the results are grouped by related columns.
🔍 Search	Opens the **Find and replace** dialog. For more information, see Find and replace on page 51.
🔍 Search next	This function continues the search using the current search parameters and highlights the next matching node. The search is run even if the **Search** dialog is not open.

Related topics

Extensions

Use extension to indirectly edit base objects (standard database objects that are contained in your project). You can add as many extensions as you wish to the base objects in the API Designer.

Extensions are edited when you configure the base objects. This means you add an extension to the object and change the property value. The base object is subsequently compiled and the modification resulting from the extension is highlighted in the API Designer.

Base objects to which you want to add an extension can be identified as colored nodes in the **Object definitions** view in the Definition tree view.

Viewing and saving generated code belonging to a node

In the Definition tree view, you can view a node's generated code (within a database object) and copy or save where necessary.

This function is particularly useful if you are looking at code in detail and want to reuse parts of the code, or you want to examine an error in more detail.

NOTE: You cannot edit the code.

NOTE: If you want to search for a specific point in the code, you can use the shortcut **Ctrl** + **F**.

To view a node's generated code

1. In the menu bar, click **View** > **Navigation**.
2. In the navigation, double-click the database object that contains the node you require.
3. Click the node in the definition tree view whose code you want to see.
4. Click 🔒 ☑ (**Generated code**).

   The code will be displayed in a code view.

To save the generated code as a file

1. Run the steps previously described.
2. In the code view, select the code that you want to save.
3. Right-click the highlighted code.
4. Click **Copy**.
5. Click **Save as** in the context menu.
6. Select a name and location for the file in the save dialog and click **Save**.

Related topics

- Editing database objects (definition tree view) on page 41
- Resolving errors and warnings on page 71

Node editor view

Open the node editor view using **View | Node editor** (see Opening the node editor view on page 47).

The **Node editor** view allows you to edit the properties of a node that you have selected in the Definition tree view (see Editing nodes on page 47).

| NOTE:The settings shown depend on the type of node selected.

The following table provides an overview of the various features available within the **Node editor** window.

Table 12: Controls

Control	Description
📁and name / icon of the base object	Displays if a file-based database object (Solution project) has been selected.
	Click 📁 **Open corresponding folder** to open the folder on your drive where the database object is located.
	For more information, see Solution projects on page 24.
Node name/node type icon	Shows the node identifier and the node type icon that marks it in the definition tree view.
🗄Database	Displays if a database object is selected.
▼ Delete filter	Deletes the filter entered into the next field.
Filter field	Displays only fields that match the text that has been entered.
	Example: If you enter **Pe**, only the **Permission** field will be displayed.
	NOTE: The API Designer only filters the settings fields and not the values contained within them.
↓ Sort ascending alphabetically	Sorts the settings alphabetically in ascending order within each category.
	If you enable sorting, the button will be highlighted with a colored frame.
↓ Sort descending alphabetically	Sorts the settings alphabetically in descending order within each category.
	If you enable sorting, the button will be highlighted with a colored frame.

Control	Description
⇟ View grouping	Enables/disables the display of the various settings in categories.
	If you enable this option, the button will be highlighted with a colored frame.
⊟ Expand all	Expands all categories.
⊟ Collapse all	Collapses all categories.
Show extension values	Highlights settings that have had an extension added to them. Use ⊠ to switch to the extension. A node that has had an extension added to it, is indicated in the definition tree view with the ☑ icon.
	If you enable this option, the button will be highlighted with a colored frame.

Related topics

- Opening the node editor view on page 47
- Editing nodes on page 47
- Menu bar on page 27
- Editing database objects (definition tree view) on page 41

Opening the node editor view

You can open the node editor view at any time.

To open the node editor view

- In the menu bar, click **View** > **Node editor**.

Related topics

- Editing nodes on page 47
- Node editor view on page 46

Editing nodes

Node properties can be edited for any node in the node editor view.

To edit a node

1. In the menu bar, click **View** > **Navigation**.
2. In the navigation, double-click the object that you would like to edit.
3. Click **View** > **Node editor** on the menu bar.

The Node editor view opens.

4. In the Definition tree view, click the node that you would like to edit.

   The fields and options shown in the node editor view will change depending on the selected node.

5. Make the changes in the **Node editor** view.

6. On the toolbar, click 🖫 (**Save**).

Related topics

- Opening the node editor view on page 47
- Node editor view on page 46
- Editing database objects (definition tree view) on page 41
- Menu bar on page 27
- Navigation on page 32

Labeling changes

Use **Change label** to make it easier to view and assign changes in the database. Any database objects that yield a project are entered on a change label. Database Transporter is used for moving the web project. You can create and edit change labels in different One Identity Manager tools.

Related topics

- Using change labels on page 48
- Creating a change label on page 49
- Change management on page 50
- Deleting change labels on page 50

Using change labels

In order to also be able to assign a change label to changes, you need to select a change label for further use.

To use a change label

- On the toolbar, select the required change label in the **Change label** list.

  Any changes stored in the project during this session will be assigned to the selected change label.

TIP: If you always use the same change label and do not want to select it every time you start the API Designer, click ⚲ **Use the current change label as default** in the toolbar.

Related topics

- Labeling changes on page 48
- Creating a change label on page 49
- Change management on page 50
- Deleting change labels on page 50

Creating a change label

You can create as many change labels as you wish and use them to track your changes.

To edit a change label

1. In the toolbar, click ⬆ (**Manage change label**).
2. Click **Create a new change label** 🗋 in the **Change label** dialog.
3. Specify the properties of the change label in the right-hand column in the list:

   - **Change label**: Enter a name for the change label.
   - **Description**: (Optional) Enter a description for the change label.
   - **Locked**: (Optional) Select whether you would like to block the change label from further use. If a change label is locked, no further changes can be booked to this label.
   - **Comment**: (Optional) Enter a comment to monitor changes to the change label.
   - **Status**: (Optional) Select a status from the list.
   - **Status comments**: (Optional) Enter a comment in relation to the status.
   - **Parent change label**: (Optional) Select a change label from the list to be the parent.

4. Click 💾 **Save change label** above the list.
5. Click **OK**.

Related topics

- Labeling changes on page 48
- Using change labels on page 48
- Change management on page 50
- Deleting change labels on page 50

Change management

You can edit existing change labels at any time.

To edit a change label

1. In the toolbar, click 🔼 (**Manage change label**).

2. In the **Change label** dialog, click the label to be edited.

3. Click 🗹 **Show / hide edit** view.

4. Change the properties of the change label in the right-hand column in the list:

   - **Change label**: Enter a name for the change label.
   - **Description**: (Optional) Enter a description for the change label.
   - **Locked**: (Optional) Select whether you would like to block the change label from further use. If a change label is locked, no further changes can be booked to this label.
   - **Comment**: (Optional) Enter a comment to monitor changes to the change label.
   - **Status**: (Optional) Select a status from the list.
   - **Status comments**: (Optional) Enter a comment in relation to the status.
   - **Parent change label**: (Optional) Select a change label from the list to be the parent.

5. Click 💾 **Save change label** above the list.

6. Click **OK**.

Related topics

Deleting change labels

You can delete existing change labels at any time.

To edit a change label

1. In the toolbar, click 🔼 (**Manage change label**).

2. In the **Change label** dialog, click the change label you would like to delete.

3. Click ⊠ **Delete the selected change label**.

4. Confirm the prompt with **Yes** in the dialog.

5. Click **Cancel**.

Related topics

Find and replace

Open the search function over **Edit** > **Find and replace** (see Running a search on page 54).

Use the **Find and replace** dialog to search for (and replace) certain captions or items within your project. The following table gives an overview of the various features within the **Find and replace** dialog.

Table 13: Controls for find and replace

Control	Description
Find	Enter the term to be found.
	TIP: To reuse terms from previous searches, click the arrow to the right of the field and select the required term.
Find as	Select whether you would like to search using simple text or by wildcards or regular expressions.
Replace by	(Optional) Enter the text that will replace the text found.
Find scope	If you would like to further narrow the search, you can select which objects you would like to search for here:
	- **Current document**: Only the current document is included in the search.
	- **Current document and its extensions**: Only the current document and all of its extensions are included in the search.
	- **Current document and its parent documents**: Only the current document and all of its parent items are included in the search.
	- **Below the selected object**: Only the objects below the current object are included in the search.

Control	Description
	• **All API Designer objects**: All objects are included in the search.
Find options	Configure other search settings:
	• **Case sensitive**: API Designer finds only occurrences of the text that match the case that you have entered into the **Find** field.
	• **Whole word**: Only the whole word is included in the search. Example: If you enter **Person**, the search will only find "person" and not "persons."
	• **Match entire value**: API Designer finds only objects containing the exact values that match the text you have entered into the **Find** field.
	• **Filter by type**: Select the objects to which your search will be limited.
Find	Finds and opens the next occurrence of the term.
Find all	Triggers a search that lists all occurrences of the text in the **Find results** pane.
Find results	Lists all occurrences of the term.
Replace	Replaces the occurrence of the term that you have highlighted in **Find results** with the text that you have entered in the **Replace by** field.
Replace all	Replaces all occurrences of the term that you have selected in the **Replace** column under **Find results** with the text that you have entered in the **Replace by** field.

Wildcards

Use wildcards to replace a single character, or a string of characters, using a single character when searching for strings. The most common wildcards are the question mark (?), to symbolize an individual character, and the asterisk (*), to symbolize any combination of characters.

To use wildcards in the **Find and replace** dialog, enter the search term into the **Find** field and select the **Wildcards** option from the **Find as** list.

Examples

The following table gives some examples of wildcard searches:

Sample term	Result
P*n	Finds "person," "position," "plugin," and so on.
Per*	Finds "person," "personal," "perfection," and so on.
AP?.json	Finds "API.json" but not "APIProject.json."

Regular expressions

Regular expressions (also known as "regex" or "regexp") are similar to wildcards in that they also allow you to find strings. Regular expressions are more effective than wildcards, however.

To use regular expressions in the **Find and replace** dialog, enter the search term into the **Find** field and select the **Regular expressions** option from the **Find as** list.

Examples

The following table gives some examples of regular expressions:

Regular expression	Description
[a-z]	Any lowercase Latin letter
[A-Z]	Any uppercase Latin letter
\d	A digit
\D	A character that is not a digit
\w	A letter, a digit, or an underscore
\W	A character that is neither a letter, a number, nor an underscore
\s	Blank space
\S	A character that is not a blank space
{n}	The preceding term must occur exactly "n" number of times. Example: **C{3}** finds "CCC"
|	Alternatives. Example: **Regex|Regexp** finds "Regex" or "Regexp"
?	The preceding term is optional, it can occur once, but does not need to, that is, the term either occurs once or not at all.

Related topics

-
-

Running a search

You can perform a search of the whole project, or parts of it, at any time.

To run a search

1. In the menu bar, click **Edit** > **Find and replace**.

2. In the **Find and replace** dialog, enter a search term in the **Find** field.

3. In the **Find as** list, select whether you would like to search using simple text or by wildcards or regular expressions.

4. (Optional) In the **Replace by** field, enter the text to replace the text in the search.

5. In the **Find scope** list, select which objects you would like to include in the search.

6. (Optional) In the **Find options** pane, click **Expand** to configure further search settings.

7. Click **Find** or **Find all**.

   The search results are displayed in the **Find results** pane

8. In the **Find options** pane, click **Expand** to configure further search settings.

9. (Optional) Double-click a result in the results list.

   This marks and displays the corresponding nodes in the definition tree view.

10. (Optional) To replace a result, highlight it in the **Find results** pane and click **Replace**.

11. (Optional) To replace several results, enable the check box next to the relevant results and click **Replace all**.

Related topics

-

Multilingual captions

Open the **Multilingual captions** dialog using **Captions**.

Use the **Multilingual captions** dialog to add, edit, or delete captions in multiple languages. You can use this text later in your API or web application.

Before adding or editing multilingual captions, define objects in your project that output captions in your API. You create keys for these objects. You assign a text to the keys for each language. This means that the keys are translated into the different languages you wish to use.

You can create and edit keys and translations on the **Captions** tab in the **Multilingual captions** dialog The following table gives an overview of the various features within the **Multilingual captions** dialog.

Table 14: Controls

Control	Description
Add	Creates a new caption. For more information, see Creating multilingual captions on page 55.
Delete	Deletes the caption selected in the results list. For more information, see Deleting a multilingual caption on page 58.
Save	Saves the changes. For more information, see Editing a multilingual caption on page 57.
Search mask	In this pane, you can search for existing captions and edit them. For more information, see Searching for a multilingual caption on page 56.
Result list	Lists the results generated by the search. For more information, see Searching for a multilingual caption on page 56.
Handling	You can edit captions in this pane. For more information, see Creating multilingual captions on page 55 and Editing a multilingual caption on page 57.

Related topics

- Creating multilingual captions on page 55
- Searching for a multilingual caption on page 56
- Editing a multilingual caption on page 57
- Deleting a multilingual caption on page 58
- Menu bar on page 27

Creating multilingual captions

You can create multilingual captions at any time.

To create a multilingual caption

1. On the toolbar, click (**Captions**).
2. Click Add on the toolbar in the **Multilingual captions** dialog.

3. In the **Edit** pane, enter a unique value in the **Key** field to be used to reference the text.

    NOTE: If you have not added a translation for a language, the API will use the caption that has been saved under **Key**.

4. Select the caption's language in the **Language** list.

5. Enter the caption to be shown for the selected language in the **Text** field.

6. Repeat the previous two steps for all the required languages.

7. Click 🖫 **Save** on the toolbar in the **Multilingual captions** dialog.

8. Click **Apply**.

Related topics

Searching for a multilingual caption

You can search for multilingual captions at any time.

To search for a multilingual caption

1. On the toolbar, click 🌐 (**Captions**).

2. In the **Multilingual captions** dialog, enter a term into the **Search form** (such as the name of a key or parts of the caption).

3. Use the options below the search bar to limit your search:

    - **Search key and value**: Searches for the term in the key and captions.

    - **Search for key only**: Only searches for the term in the key.

    - **Search for value only**: Only searches for the term in the captions.

    - **Search in all available languages**: Enable this option if you would like to search for the term across all languages. If you disable this option, the search will only be performed in the language that is current shown in the **Language** field in the **Edit** pane.

4. Click 🔍 (**Search**).

    The search results are displayed in the **Result list** pane

    TIP: If one of the **Search key and value** or **Search for key only** options is set, the keys shown in the result list are labeled with an asterisk (*).

Related topics

Editing a multilingual caption

You can edit existing multilingual captions at any time.

To edit a multilingual caption

1. On the toolbar, click 🌐 (**Captions**).

2. In the **Multilingual captions** dialog, enter the search term in the **Search form** section.

3. Use the options below the search bar to limit your search:

   - **Search key and value**: Searches for the term in the key and captions.
   - **Search for key only**: Only searches for the term in the key.
   - **Search for value only**: Only searches for the term in the captions.
   - **Search in all available languages**: Enable this option if you would like to search for the term across all languages. If you disable this option, the search will only be performed in the language that is current shown in the **Language** field in the **Edit** pane.

4. Click 🔍 (**Search**).

5. In the **Result list** section, click the caption to be edited.

   TIP: If one of the **Search key and value** or **Search for key only** options is set, the keys shown in the result list are labeled with an asterisk (*).

6. In the **Edit** section, change the unique value in the **Key** field that is used to reference the text.

   NOTE: If you have not added a translation for a language, the API will use the caption that has been saved under **Key**.

7. Select the caption's language in the **Language** list.

8. Enter the caption to be shown for the selected language in the **Text** field.

9. Repeat the last two steps for any languages that you would like to change or add.

10. Click 💾 **Save** on the toolbar in the **Multilingual captions** dialog.

11. Click **Apply**.

Related topics

Deleting a multilingual caption

You can delete existing multilingual captions at any time.

NOTE: You cannot delete multilingual captions that have been predefined by API Designer.

To delete a multilingual caption

1. On the toolbar, click ⊕ (**Captions**).
2. In the **Multilingual captions** dialog, enter a term into the **Search parameters** (such as the name of a key or parts of the caption).
3. Use the options below the search bar to limit your search:
   - **Search key and value**: Searches for the term in the key and captions.
   - **Search for key only**: Only searches for the term in the key.
   - **Search for value only**: Only searches for the term in the captions.
   - **Search in all available languages**: Enable this option if you would like to search for the term across all languages. If you disable this option, the search will only be performed in the language that is current shown in the **Language** field in the **Edit** section.
4. Click 🔍.
5. In the **Result list** section, click the caption to be deleted.

   TIP: If one of the **Search key and value** or **Search for key only** options is set, the keys shown in the result list are labeled with an asterisk (*).
6. Click 🗑 **Delete** on the toolbar in the **Multilingual captions** dialog.
7. Confirm the prompt with **Yes** in the dialog.
8. Click **Apply**.

Related topics

Managing database queries

Open the **Edit database queries** dialog over **Edit | Edit database queries** (see Displaying database queries on page 60).

In the **Edit database queries** dialog, you can display, create, edit, delete, and test database queries.

Database queries enable secure communication between your web application and your One Identity Manager database. The actual SQL code is saved in SQL snippets, to which only certain parameters can be added later. The database queries are referenced later in the code only based on their name.

The following table gives an overview of the various features within the **Edit database queries** dialog.

Table 15: Controls

Control	Description
Add	Creates a new database query. For more information, see Creating a database query on page 60.
Delete	Deletes the selected database query. For more information, see Deleting a database query on page 62.
Save	Saves changes to the selected database query. For more information, see Editing a database query on page 61.
Save all	Saves all changes. For more information, see Editing a database query on page 61.
Discard object changes	Undoes all changes to the selected database query.
Identifier	Is a unique database query ID.
Description	Gives a description of the database query that explains the query and its function.
SQL expression	Is the actual query in SQL code.
Dialog groups	Are dialog groups that are permitted to use this database query (labeled ⊘) or forbidden from using it (labeled ⊗).
Test statement	Opens the **Test statement** dialog. You can test the database query here. For more information, see Testing a database query on page 62.

Related topics

-
-
-
-
-

Displaying database queries

You can edit all existing database queries at any time.

To show all database queries

- In the menu bar, click **Edit** > **Edit database queries**.

Related topics

-
-
-
-

Creating a database query

You can create new database queries at any time.

To create a database query

1. In the menu bar, click **Edit** > **Edit database queries**.
2. Click **Add** in the **Edit database queries** dialog.
3. Enter a unique name for the database query in the **Identifier** field.
4. (Optional) In the **Description** field, enter a description for the database query that describes the database query and its function.
5. Enter the query as a SQL code in the **SQL expression** field.
6. Double-click the dialog groups that may use the database query in the **Dialog groups** pane.

   TIP: Click **Select all** or **Deselect all** to quickly select or deselect all dialog groups.

7. (Optional) Click **Test statement** to test the SQL statement with different values. For more information, see Testing a database query on page 62.

8. Click (**Save**).

9. Click **Close**.

Related topics

- Managing database queries on page 59
- Displaying database queries on page 60
- Editing a database query on page 61
- Testing a database query on page 62
- Deleting a database query on page 62

Editing a database query

> NOTE: You cannot edit any database queries that have been predefined by the API Designer.

You can edit existing database queries at any time.

To edit a database query

1. In the menu bar, click **Edit** > **Edit database queries**.

2. Click the database query you would like to edit in the **Edit database queries** dialog.

3. Enter a unique name for the database query in the **Identifier** field.

4. (Optional) In the **Description** field, enter a description for the database query that describes the database query and its function.

5. Enter the query as a SQL code in the **SQL expression** field.

6. In the **Dialog groups** section, double-click the dialog groups for which access permissions are to be changed.

   > TIP: Click **Select all** or **Deselect all** to quickly select or deselect all dialog groups.

7. (Optional) Click **Test statement** to test the SQL statement with different values. For more information, see Testing a database query on page 62.

8. Click (**Save**).

9. Click **Close**.

Related topics

- Managing database queries on page 59
- Displaying database queries on page 60

Testing a database query

You can test database queries at any time.

To test a database query

1. In the menu bar, click **Edit** > **Edit database queries**.

2. In the **Edit database queries** dialog, click the database query you would like to test.

3. Click **Test statement**.

4. In the **Test statement** dialog, select the check box next to the SQL statement parameters that you would like to include in the test.

5. Enter the values for each parameter in the **Value** column.

6. Click **Test statement**.

7. Click **Close**.

8. Click **Close** in the **Edit database queries** dialog.

Related topics

- Managing database queries on page 59
- Displaying database queries on page 60
- Creating a database query on page 60
- Editing a database query on page 61
- Deleting a database query on page 62

Deleting a database query

NOTE: You cannot delete any database queries that have been predefined by the API Designer. The **Delete** button is disabled for such database queries.

You can delete existing database queries at any time.

To delete a query

1. In the menu bar, click **Edit** > **Edit database queries**.

2. Click the database query you would like to delete in the **Edit database queries** dialog.

3. Click (**Delete**).

4. Confirm the prompt with **Yes** in the dialog.

5. Click **Close** in the **Edit database queries** dialog.

Related topics

- Managing database queries on page 59
- Displaying database queries on page 60
- Creating a database query on page 60
- Editing a database query on page 61
- Testing a database query on page 62

Managing tabs

Use the **Tabs** dialog to enable opened tabs, close tabs, or save changes that you have made in tabs.

To manage tabs

1. In the menu bar, click **View** > **Tabs**.

2. In the **Tabs** dialog, click or more tabs (while holding down the **Ctrl** key).

3. Perform one of the following tasks:

   - Click **Enable** to enable the tab.
   - Click **Save** to save tab changes.
   - Click **Close tabs** to close the tab.

4. Click **Close**.

Related topics

- Menu bar on page 27

Managing layouts

You can adjust, save, and, where necessary, restore the layout (assign and show different menus and panes) at any time.

To save your own layout

1. Customize the layout of the API Designer to suit your needs.

2. In the menu bar, click **View** > **Save layout**.

To restore a saved layout

- In the menu bar, click **View** > **Restore saved layout**.

To restore the default layout

- In the menu bar, click **View** > **Restore standard layout**.

To restore the default layout including the window size

- In the menu bar, click **View** > **Restore standard layout (including size)**.

Related topics

- Menu bar on page 27

Displaying the change history (command list)

Open the command list using **View | Command list** (see Opening the command list on page 65).

Use the **Command list** window to display any changes made to an object and to either undo or redo them.

Commands that have already run are indicated with the ⊘ icon. This icon is not shown for commands that were undone.

The use of wizards allows numerous commands to be automatically implemented, which in the command list are displayed as composite commands and the individual commands are shown at a second level. You can only undo the composite commands.

The following table gives an overview of the various features available in the **Command list** window.

Table 16: Controls

Control	Description
↩ Undo	Undoes the last implemented command in the list.
↪ Redo	Redoes the last command in the list that was undone.

Related topics

- Opening the command list on page 65

Opening the command list

You can view a list of actions that have been taken at any time.

To open the command list

- In the menu bar, click **View** > **Command list**.

Related topics

- Displaying the change history (command list) on page 64

Bookmarks

You open the **Bookmarks** window using **View** > **Bookmarks** (see Editing bookmarks on page 66).

To make navigating around the API Designer easier and to find objects quickly, you can set bookmarks to any node in the Definition tree view. View, manage, and use these bookmarks in the **Bookmarks** window.

The following table provides an overview of the various features available within the **Bookmarks** window.

Table 17: Controls

Control	Description
📑 Delete all bookmarks	Deletes all bookmarks (see Deleting bookmarks on page 67).
✏ Edit bookmark description	Opens a dialog that allows you to change the description of the selected bookmark (see Editing bookmarks on page 66).
📄 Remove module/-	Deletes all bookmarks for the module (or component) selected in

Control	Description
component bookmark	the list (see Deleting bookmarks on page 67). NOTE: This feature is only available if you selected to group the bookmarks by module/component earlier (see **Group by modules/components**) button.
Delete current bookmark	Deletes the bookmark that you have selected in the list (see Deleting bookmarks on page 67).
Group by modules/-components	Groups the bookmarks by the API projects to which they are assigned.

Related topics

Editing bookmarks

You can show all bookmarks, delete all bookmarks, or change the description of bookmarks at any time.

To open the bookmark window

- In the menu bar, click **View** > **Bookmarks**.

To open a bookmark

1. In the menu bar, click **View** > **Bookmarks**.
2. Double-click a bookmark in the **Bookmarks** window.

To edit the bookmark description

1. In the menu bar, click **View** > **Bookmarks**.
2. Click a bookmark in the **Bookmarks** window.
3. Click **Edit bookmark description**.
4. Insert a description for the bookmark in the **Bookmark description** dialog.
5. Click **OK**.

Related topics

- Deleting bookmarks on page 67

Setting bookmarks

You can set a bookmark for database objects at any time. This allows you to do things such as quickly access frequently used nodes.

To set a bookmark

1. In the menu bar, click **View** > **Navigation**.
2. In the navigation, double-click the database object that contains the node you require.
3. In the definition tree view, right-click the node for which you would like to set a bookmark.
4. Click 📘 **Set bookmark** in the context menu.
5. Insert a description for the bookmark in the **Bookmark description** dialog.
6. Click **OK**.

Related topics

- Bookmarks on page 65
- Editing bookmarks on page 66
- Deleting bookmarks on page 67

Deleting bookmarks

You can delete single bookmarks, all bookmarks within a module, or all bookmarks within a project at any time.

To delete a single bookmark

1. In the menu bar, click **View** > **Bookmarks**.
2. Click the bookmark that you would like to delete in the **Bookmarks** window.
3. Click 📄 **Delete current bookmark**.
4. Confirm the prompt with **Yes** in the dialog.

To delete all the bookmarks in a module/component

1. In the menu bar, click **View** > **Bookmarks**.
2. Click 📑 **Group by modules/components** in the **Bookmarks** window.
3. Click the module whose bookmarks you would like to delete.

4. Click ⊠ **Remove bookmarks from module/component**.

5. Confirm the prompt with **Yes** in the dialog.

To remove all bookmarks

1. In the menu bar, click **View** > **Bookmarks**.

2. In the window, click ⊟ **Delete all bookmarks** (**Bookmarks**).

3. Confirm the prompt with **Yes** in the dialog.

Related topics

- Bookmarks on page 65
- Editing bookmarks on page 66
- Setting bookmarks on page 67

Compiling an API

TIP: See additional configuration options for compilation directly on the API project (see Configuring the compilation on page 84 and in the global settings).

Before you can use an API that has been created with the API Designer, the API must be compiled using the API Designer compiler.

The API Designer compiler can be opened from the API Designer, as well as from the Database Compiler. For more information about the Database Compiler, see the *One Identity Manager Operational Guide*.

A web project must be compiled in the following cases:

- After changing a definition (API file or API project) in the API Designer
- After changing certain system settings that require Database Compiler to be run

The API Designer compiler creates a set of DLL files from the project's XML definition and saves them in the database.

Methods of compiling

There are two ways to compile a web project:

- Run a **Release compilation** to release a specific version of the project for use.

  NOTE: Changes within the API Designer do not affect the API as long as there is no release compilation.

- Use a **Debug compilation** during the development phase for testing and debugging purposes. Debug compilation creates additional code to support the API Designer debugger. This means the DLL files are somewhat larger.

An API loads the latest compiled DLL files on startup. If these DLL files are updated, the web project reloads the new DLL files; however, only new sessions run with the code from the newly loaded DLLs.

Related topics

- Testing a compilation on page 69
- Starting a compilation on page 70
- Managing versions (compilation branches) on page 72
- Managing compilation errors and warnings (task window) on page 70
- Resolving errors and warnings on page 71
- Opening the compiling log on page 72
- Configuring the compilation on page 84
- Global settings on page 34

Testing a compilation

The two types of compilation (Debug and Release compilation) can be tested in advance.

To test the compilation

1. In the menu bar, click **View** > **Home**.
2. On the start page, expand the **Compilation** pane.
3. In the **Compilation** pane, perform one of the following actions:
   - To test a debug compilation, enable the **DEBUG** option.
   - To test a release compilation, enable the **RELEASE** option.
4. Click **Test compilation**.

   The Compilation log opens and shows compilation status and progress.

   The **Tasks** window opens. Use this to view and rectify any errors and warnings that arise during compilation. For more information, see Managing compilation errors and warnings (task window) on page 70.

Related topics

- Compiling an API on page 68
- Managing compilation errors and warnings (task window) on page 70
- Menu bar on page 27

Starting a compilation

You can start two types of compilation (Debug and Release). If compilation completes without any errors, then the compiled API is written to the database.

To start compilation

1. In the menu bar, click **View** > **Home**.

2. On the start page, expand the **Compilation** pane.

3. In the **Compilation** pane, perform one of the following actions:

   - To start a debug compilation, enable the **DEBUG** option.
   - To start a release compilation, enable the **RELEASE** option.

4. Click **Compile**.

   The Compilation log opens and shows compilation status and progress.

   The **Tasks** window opens if errors arise. Use this to view and rectify any errors and warnings that arise during compilation. For more information, see Managing compilation errors and warnings (task window) on page 70.

Related topics

- Compiling an API on page 68
- Managing compilation errors and warnings (task window) on page 70
- Testing a compilation on page 69
- Menu bar on page 27

Managing compilation errors and warnings (task window)

Open the task window with **View** > **Tasks** (see Opening the task window on page 71).

Use the **Tasks** window to view compilation errors and warnings and to rectify them where necessary (debug).

- **Compilation errors** prevent the web project from compiling properly and must therefore be eliminated. Development states cannot be released if they cannot be compiled.

- **Compilation warnings** relate to missing extensions, or to messages concerning accessibility. The compilation warnings of the relevant compiler are also taken into account. If compiler warnings are the only type of message that is generated, the development state will be successfully compiled nonetheless.

TIP: In the global settings, you can define which warnings are displayed as errors during compilation and which errors are ignored. For more information, see Global settings on page 34.

If individual messages are displayed (⊞ icon) together, it means that the errors indicated occur at various locations throughout the web project. This can happen, for example, if an extension that is referenced by a number of nodes is missing.

The following table gives an overview of the various features in the **Tasks** window.

Table 18: Controls

Control	Description
❌ Errors	Hides or displays compilation errors.
⚠️ Warnings	Hides or displays compilation warnings.
⚡ Error details	Displays the error message. This button is only active if you have highlighted a listed error message.
	Click this button to view a detailed description of the error.
	Click **Send as mail** to transfer the error message to an email and send.

Related topics

- Opening the task window on page 71
- Resolving errors and warnings on page 71

Opening the task window

You can open the task window at any time.

To open the task window

- In the menu bar, click **View** > **Tasks**.

Related topics

- Resolving errors and warnings on page 71
- Managing compilation errors and warnings (task window) on page 70

Resolving errors and warnings

If errors occur when compiling your API, the task view can be used to see and resolve them.

TIP: In the global settings, you can define which warnings are displayed as errors during compilation and which errors are ignored. For more information, see Global settings on page 34.

To open and edit objects with errors in the task window

1. In the menu bar, click **View** > **Tasks**.

2. Click either **Errors** or **Warnings** in the task window.

3. Perform one of the following tasks:

   - Double-click the listed error/warning and then resolve the error/warning within the **Node editor** window itself.

     NOTE: The node editor window opens if the node contains the error. If the error occurs when the resulting code is compiled, you can double-click it to display **Generated code (read-only)** in the definition tree view.

     - OR -

   - Click the error/warning in the list and then click 🛠 **Correct errors**.

Related topics

- Opening the task window on page 71
- Managing compilation errors and warnings (task window) on page 70

Opening the compiling log

The compiling log displays compiling status and progress.

To open the compiling log

- In the menu bar, click **View** > **Compilation**.

Related topics

- Compiling an API on page 68
- Menu bar on page 27

Managing versions (compilation branches)

To manage different versions of your compiled API and save them in the database, you use compilation branches. Compilation branches are managed in a separate pane on the start page. Here you can select the compilation branch that you want to use, display the existing

compilation branches, create compilation branches, edit compilation branches, and delete compilation branches.

Related topics

- Compiling an API on page 68
- Selecting and using compilation branches on page 73
- Creating compilation branches on page 73
- Editing compilation branches on page 74
- Deleting compilation branches on page 75

Selecting and using compilation branches

To select and use compilation branches

1. In the menu bar, click **View** > **Home**.
2. On the start page, expand the **Compilation branches** pane.
3. In the **Compilation branches** pane, in the **Compilation branch identifier** list, click the compilation branch that you want to use.

   TIP: If you do not want to use a compilation branch, click ◆ (**Use main branch**).

Related topics

- Managing versions (compilation branches) on page 72
- Creating compilation branches on page 73
- Editing compilation branches on page 74
- Deleting compilation branches on page 75

Creating compilation branches

To use a compilation branch

1. In the menu bar, click **View** > **Home**.
2. On the start page, expand the **Compilation branches** view.
3. In the **Compilation branches** view, click 🔧 **Manage name of branches**.
4. In the **Manage compilation branches** window, click 📄 **Add**.

   A new entry **Unknown** is added at the end of the list.
5. In the list, click **Unknown**.
6. In the **Identifier** field, enter a unique name for the compilation branch.

7. (Optional) In the **Description** field, enter a description for the compilation branch. This description may explain, for example, the intended purpose of the compilation branch.

8. Click ![Save icon] (**Save**).

Related topics

- Managing versions (compilation branches) on page 72
- Selecting and using compilation branches on page 73
- Editing compilation branches on page 74
- Deleting compilation branches on page 75

Editing compilation branches

To edit existing compilation branches

1. In the menu bar, click **View** > **Home**.

2. On the start page, expand the **Compilation branches** pane.

3. In the **Compilation branches** pane, click ![icon] **Manage compilation branches**.

4. In the **Manage compilation branches** dialog, click the compilation branch that you want to edit in the list.

5. In the **Identifier** field, enter a unique name for the compilation branch.

6. (Optional) In the **Description** field, enter a description for the compilation branch. This description may explain, for example, the intended purpose of the compilation branch.

7. Click ![Save icon] (**Save**).

Related topics

- Managing versions (compilation branches) on page 72
- Selecting and using compilation branches on page 73
- Creating compilation branches on page 73
- Deleting compilation branches on page 75

Deleting compilation branches

To delete compilation branches

1. In the menu bar, click **View** > **Home**.

2. On the start page, expand the **Compilation branches** pane.

3. In the **Compilation branches** pane, click **Manage compilation branches**.

4. In the **Manage compilation branches** dialog window, click the compilation branch that you want to delete from the list.

5. Click (**Delete**).

6. Confirm the **Delete?** prompt by selecting **Yes**.

Related topics

- Managing versions (compilation branches) on page 72
- Selecting and using compilation branches on page 73
- Creating compilation branches on page 73
- Editing compilation branches on page 74

Testing an API

You can test the functionality of your API locally on your PC at any time.

To test your API locally

1. In the menu bar, click **View** > **Home**.

2. Expand the **Self-hosted API Server** pane on the start page.

3. In the **Startup options** field, enter the options for starting the API.

   TIP: Alternatively, you can also select previous commands from the list.

4. Click **Start**.

   The API is compiled and will then be available locally. The corresponding web address is displayed next to the **Status**. Click the address to open this directly in the browser.

Related topics

- Opening the start page on page 34

Linking C# projects to the API Designer

To edit an API as a C# project with external programs (for example, Visual Studio), you must perform three steps:

1. Link your API Designer project with a new or existing C# project.
2. Edit the C# project in an external program.
3. Open the C# project again in API Designer and save it in the API project.

To link the API Designer project with a C# project

1. In the menu bar, click **View** > **Home**.
2. On the start page, expand the **C# development** pane.
3. In the **C# development** pane, perform one of the following actions:
   - To export and link the API Designer project as a new C# project, click 📤 **Export C# project**. Navigate to the folder where you want to export the content of your API Designer project as a C# project and click **OK**.
   - To link the API Designer project with an existing project, click 📂 **Load C# project**.

   The API Designer is linked with the C# project.

To edit the C# project in an external program

1. In the menu bar, click **View** > **Home**.
2. On the start page, expand the **C# development** pane.
3. In the **C# development** pane, click **Project file: <path to C# project>**.

   The project opens in the program that the project file is linked with.

   | TIP: To open the project folder directly, click 📂 **Open corresponding folder**.

4. Edit the project in the external program. For example, you can create, change, or delete API files.

To save changes back to the API Designer project

1. Open the API Designer.
2. In the menu bar, click **View** > **Home**.
3. On the start page, expand the **C# development** pane.
4. In the **C# development** pane, click ↔ **Synchronize C# project with base**.

   You now see the objects that you have changed outside the API Designer.

5. In the list, next to the changed objects, select the actions that you want to perform for the API Designer project.

Status message	Description	Actions
New C# file	A new API file was created in the C# project.	**Add API file**: Adds the C# file that you created in the external program to yourAPI Designer project as an API file.
		Delete file: Deletes the C# class in the C# project.
New API file	A new API file was created in the API Designer Sharp project.	**Add C# file**: Adds the API file that you created in the API Designer to your C# project as a C# file.
		Delete API file: Deletes the API file in the API Designer project.
API file is newer	An API file was modified in the API Designer project.	**Update**: Transfers the changes to the API file from the API Designer project into the C# project.
		Revert changes: Reverses the changes that you made in your API Designer project in the API file.
C# file is newer	An API file was modified in the C# project.	**Update**: Transfers the changes to the API file from the C# project into the API Designer project.
		Revert changes: Reverses the changes that you made in your C# project in the API file.

6. Select the check box next to the objects for which the selected actions should be performed.

   TIP: To quickly select or deselect all objects, select the **Select all/deselect all** check box above the list.

7. Click ✔ (**Apply selected actions**).

API projects

An API project represents the actual API application itself.

You can combine API files that you have created in one API project for a logical application. The API project includes the configuration. The API project includes the authentication on the database.

Related topics

- Creating API projects on page 78
- Editing API projects on page 79
- Deleting API projects on page 80
- Configuring authentication on page 81
- Configuring the compilation on page 84
- Assigning API files to an API project on page 89
- Importing API projects on page 85

Creating API projects

You can create new API projects at any time.

To create an API project

1. In the menu bar, click **View** > **Navigation**.

2. Click ![icon] **API projects** in the navigation.

3. Click ![icon] (**Add**) > **Add API project**.

4. In the definition tree view, click the topmost node.

5. Click **View** > **Node editor** on the menu bar.

6. In the node editor view, configure the following settings.

Setting	Description
General settings	
Identifier	Enter a unique name for the node.
Technical abbreviation	Enter a short name for the API project. This name is used later for the assignment of the project.
Control-ID	Enter a unique ID for the node.
Model version	Shows the model version being used.
Workflow control	
Allow extensions to this document	Enable this option to permit the creation of enhancements to this node.
Advanced settings	
Required database modules	(Optional) Enter the database modules required for this API project.

7. Configure further API project settings (see Configuring authentication on page 81 and Configuring the compilation on page 84).

8. On the toolbar, click 🖫 (**Save**).

Related topics

- Editing API projects on page 79
- Configuring authentication on page 81
- Configuring the compilation on page 84
- Deleting API projects on page 80
- API projects on page 78
- Navigation on page 32

Editing API projects

You can edit any API projects you have created at any time.

NOTE: You cannot edit any API projects that have been predefined by the API Designer.

To edit an API project

1. In the menu bar, click **View** > **Navigation**.

2. Click **API projects** in the navigation.

3. In the tree structure, double-click the API project to be edited.

4. Click **View** > **Node editor** on the menu bar.

5. Use the Definition tree view and the Node editor view to configure further API project settings (see Configuring authentication on page 81 and Configuring the compilation on page 84).

6. On the toolbar, click (**Save**).

Related topics

- Creating API projects on page 78
- Configuring authentication on page 81
- Configuring the compilation on page 84
- Deleting API projects on page 80
- API projects on page 78
- Navigation on page 32

Deleting API projects

You can delete any API projects you have created at any time.

NOTE: You cannot delete any API projects that have been predefined by the API Designer.

To delete an API project

1. In the menu bar, click **View** > **Navigation**.

2. Click **API projects** in the navigation.

3. In the tree structure, click on the API project to be deleted.

4. Click (**Delete**).

5. Confirm the prompt with **Yes** in the dialog.

6. Click (**Reload data**) in the tree structure.

Related topics

- Creating API projects on page 78
- Editing API projects on page 79

Configuring authentication

The **Authentication** node in the definition tree view is used to determine how clients will log in to the API.

There are three authentication options that you can configure:

- Standard: Only the authentication methods you list here are permitted. For this method you can also enable single sign-on.
- Allow all manual modules: All manual authentication modules are permitted, if these are activated for the selected product. For this method you can also enable single sign-on.
- Fixed credentials: This settings enables the user to log in to the API using fixed login data. To use this option, the login data must be stored in the `web.config` files on each API server.

To set the permitted manual authentication options

1. In the menu bar, click **View** > **Navigation**.

2. Click **API projects** in the navigation.

3. In the tree structure, double-click the API project to be edited.

4. In the Definition tree view, click the node (**Authentication**).

5. Click **View** > **Node editor** on the menu bar.

6. In the node editor view, perform the following actions:

    a. In the **Authentication type** menu, select **Standard**.

    b. Enter a unique ID for the node in the **Control ID** field.

    c. (Optional) In the **Authentication properties** field, enter the properties for the authentication. For more information about the authentication modules, see the *One Identity Manager Configuration Guide*.

    d. (Optional) If authorizations for users are controlled by a product, enable the **Product** option and select the required product in the selection list.

7. In the definition tree view, expand the node (**Authentication**).

8. Right-click the node (**Manual authentication modules**.

9. Click **Authentication module** in the context menu.

    The **Authentication module** node is placed as a child to the node.

10. Click the **Authentication module** node.

11. In the node editing view, select the required authentication module from the **Name** selection list. For more information about the authentication modules, see the *One Identity Manager Configuration Guide*.

12. Enter a unique ID for the node in the **Control ID** field.

13. Repeat steps 8 to 12 until to add each authentication method you wish to use.

14. On the menu bar, click 🖫 (**Save**).

To permit all manual authentication options

1. In the menu bar, click **View** > **Navigation**.

2. In the navigation, click 🧊 (**API projects**).

3. In the tree structure, double-click the API project to be edited.

4. In the Definition tree view, click the 🔒 node (**Authentication**).

5. Click **View** > **Node editor** on the menu bar.

6. In the node editor view, perform the following actions:

   a. In the **Authentication type** selection list, select **Allow all manual modules**.

   b. Enter a unique ID for the node in the **Control ID** field.

   c. (Optional) In the **Authentication properties** field, enter the authentication properties.

   d. (Optional) Enable the **Product** option and select the product from the list.

7. On the menu bar, click 🖫 (**Save**).

To allow single sign-on

1. In the menu bar, click **View** > **Navigation**.

2. In the navigation, click 🧊 (**API projects**).

3. In the tree structure, double-click the API project to be edited.

4. In the definition tree view, expand the 🔒 node (**Authentication**).

5. Right-click the 🔒 **Authentication modules for single sign-on** node.

   NOTE: This node is only available if you have manually set the authentication options yourself or if you have enabled all manual authentication options.

6. Click **Authentication module** in the context menu.

   The **Authentication module** node is placed as a child to the node.

7. Click the **Authentication module** node.

8. Click **View** > **Node editor** on the menu bar.

9. In the node editing view, select the required authentication module from the **Name** selection list. For more information about the authentication modules, see the *One*

Identity Manager Configuration Guide.

10. On the menu bar, click 🖫 (**Save**).

Log in with stored login data

To approve a login with stored login data, perform the following two steps:

1. Store the login data for the users for whom you want to allow access in the `web.config` file of each API server.

2. Configure the authentication in the API Designer for the API project.

To store login data in the API Server

1. Connect to your API Server.

2. Open the `web.config` file in a text editor.

   | NOTE: If the file is encrypted, you must decrypt it before editing.

3. In the `<connectionStrings>` section, add the following entry:

   ```
   <add name="sub:<NAME>" connectionString="Module=DialogUser;User=<USER>;
   (Password)Password=<PASSWORD>" />
   ```

   - `<NAME>` stands for the name/ID of the API project.
   - `<USER>` stands for the login name of the user.
   - `<PASSWORD>` stands for the user's password.

4. Save your changes to the file.

5. (Optional) encrypt the file.

To configure the login with the saved login data on the API project

1. Start the API Designer.

2. In the menu bar, click **View** > **Navigation**.

3. In the navigation, click 🟦 (**API projects**).

4. In the tree structure, double-click the API project to be edited.

5. In the Definition tree view, click the 🔒 node (**Authentication**).

6. Click **View** > **Node editor** on the menu bar.

7. In the node editor view, perform the following actions:

   a. In the **Authentication type** selection list, select **Fixed credentials**.

   b. Enter a unique ID for the node in the **Control ID** field.

   c. (Optional) In the **Authentication properties** field, enter the authentication properties.

   d. (Optional) Enable the **Product** option and select the product from the list.

8. On the menu bar, click 🖫 (**Save**).

Related topics

- Creating API projects on page 78
- Editing API projects on page 79
- API projects on page 78

Configuring the compilation

Use the **Compilation settings** node in the definition tree view to configure the settings for compiling the API.

If you want to use code from other DLLs in your API code (for example, API files from other projects), you need to create an assembly reference so that the compiler recognizes these DLLs.

To create an assembly reference

1. In the menu bar, click **View** > **Navigation**.
2. Click ![icon] **API projects** in the navigation.
3. In the tree structure, double-click the API project to be edited.
4. In the definition tree view, right-click the **Compilation settings** node.
5. In the context menu, click **Assembly reference**.

   The **Assembly reference** node is added under this node.
6. In the definition tree view, click **Assembly reference**.
7. Click **View** > **Node editor** on the menu bar.
8. In the node editor view, enter the name of the assembly you wish to reference into the **Referenced assembly** field.
9. On the menu bar, click ![icon] (**Save**).

Related topics

- Compiling an API on page 68
- Creating API projects on page 78
- Editing API projects on page 79
- API projects on page 78
- Configuring authentication on page 81

Assigning API files to an API project

In order for the API files you have created to be used practically, they must be assigned to an API project.

To assign API files to an API project

1. In the menu bar, click **View** > **Navigation**.

2. Click ⬢ **API projects** in the navigation.

3. In the tree structure, double-click the API project to be edited.

4. In the definition tree view, right-click the topmost node.

5. In the context menu, click **API file reference**.

   The new **API file reference** node is added.

6. In the definition tree view, click **API file reference**.

7. Click **View** > **Node editor** on the menu bar.

8. In the node editor view, enter a unique ID for the node into the **Control ID** field.

9. Select the required file in the **Name** list.

   | TIP: To skip directly to the file definition, click ⬀ (**Show definition object**).

10. On the toolbar, click 💾 (**Save**).

Related topics

- Creating API files on page 88
- Editing API files on page 88
- Importing API files on page 90
- Deleting API files on page 89
- Creating API projects on page 78
- Editing API projects on page 79
- Deleting API projects on page 80
- API projects on page 78

Importing API projects

You can import API projects into the API Designer. The API Designer independently detects whether the projects are API projects, and imports them as such.

To import an API project

1. In the menu bar, click **Edit** > **Import object**.

2. In the file browser, select the required API project and click **Open**.

   The API project is imported.

Related topics

- API projects on page 78
- Menu bar on page 27

API methods

You can use an API method to create a call for exchanging data with the server (or database) relating to a specific application scenario. An API method can belong to more than one project. In an API method, you can define an API method, for example.

NOTE: If you create a new API method to provide a new API method, you must name the API method the same as the class. Ensure you use the correct case.

Using the API Designer, you can define the following types of API methods in API methods

Entity methods

Entity methods work with small parts of the object model in order to read data from the database or write data to the database. When you create an entity method, you only need to enter the table and column name and, if required, a filter condition (WHERE clause). Internal processing is handled by the API Server. The data schema for the input and output also has a specific format.

For examples for the definition of entity methods, see the SDK under `Sdk01_Basics\01-BasicQueryMethod.cs`.

User-defined methods

User-defined methods are methods for which you fully define the processing, input, and output data in code. This type therefore offers the greatest flexibility.

For examples for the user-defined methods, see the SDK under `Sdk01_Basics\03-CustomMethod.cs`.

SQL methods

SQL methods are methods that provide data from a predefined SQL query through the API. Create the parameters of a query as SQL parameters.

For examples for the definition of SQL methods, see the SDK under `Sdk01_Basics\02-BasicSqlMethod.cs`.

Related topics

Creating API files

You can create new API files at any time.

NOTE: If you create a new API file to provide a new API method, you must name the API file the same as the class. Ensure you use the correct case.

To create an API file

1. In the menu bar, click **View** > **Navigation**.

2. In the navigation, click ⚙ **API files**.

3. Click ➕ **Add** > **Add API file**.

4. In the new window, create the file definition.

5. On the toolbar, click 💾 (**Save**).

Related topics

Editing API files

You can edit the API files you have created at any time.

NOTE: You cannot edit API files that have been predefined by the API Designer.

To edit an API file

1. In the menu bar, click **View** > **Navigation**.

2. In the navigation, click ⚙ **API files**.

3. In the tree structure, click the API file that you want to edit.

4. In the new window, create the file definition.

5. On the toolbar, click 🖫 (**Save**).

Related topics

- API methods on page 87
- Creating API files on page 88
- Deleting API files on page 89
- Navigation on page 32

Deleting API files

You can delete the API files you have created at any time.

NOTE: You cannot delete API files that have been predefined by API Designer.

To delete an API file

1. In the menu bar, click **View** > **Navigation**.

2. In the navigation, click 🔘 (**API files**).

3. In the tree structure, click the API file that you want to delete.

4. Click 📄 (**Delete**).

5. Confirm the prompt with **Yes** in the dialog.

6. Click 🔄 (**Reload data**) in the tree structure.

Related topics

- API methods on page 87
- Creating API files on page 88
- Editing API files on page 88
- Navigation on page 32

Assigning API files to an API project

In order for the API files you have created to be used practically, they must be assigned to an API project.

To assign API files to an API project

1. In the menu bar, click **View** > **Navigation**.

2. Click ![icon] **API projects** in the navigation.

3. In the tree structure, double-click the API project to be edited.

4. In the definition tree view, right-click the topmost node.

5. In the context menu, click **API file reference**.

   The new **API file reference** node is added.

6. In the definition tree view, click **API file reference**.

7. Click **View** > **Node editor** on the menu bar.

8. In the node editor view, enter a unique ID for the node into the **Control ID** field.

9. Select the required file in the **Name** list.

   | TIP: To skip directly to the file definition, click ![icon] (**Show definition object**).

10. On the toolbar, click ![icon] (**Save**).

Related topics

Importing API files

You can import API files into the API Designer. The API Designer automatically detects that the files are API files and imports them as such.

To import an API file

1. In the menu bar, click **Edit** > **Import object**.

2. In the file browser, select the API file and click **Open**.

   The API file is imported.

Related topics

- API methods on page 87
- Menu bar on page 27

API Server

The API that you created in the API Designer is available in the API Server.

To call the API Server web interface

- In a web browser, open the web address (URL) of your API Server.

  From here, you can:

  - Call the swagger documentation for your API.
  - Open the Operations Support Web Portal.
  - Call all installed web applications.

ImxClient command line program

The ImxClient command line tool can be used to run all API Designer API functions from a command line without a graphical interface.

Related topics

- Starting the ImxClient command line program on page 93
- ImxClient command overview on page 93

Starting the ImxClient command line program

You can start the ImxClient command line tool at any time using any command line interface.

To start the lmxClient command line program

1. Open a command line interface (for example, Windows Powershell).
2. In the command line program, go to the installation path for the API Designer.
3. Run the `ImxClient.exe` application.

Related topics

- ImxClient command line program on page 93

ImxClient command overview

The following chapters contain a list of all ImxClient commands that you can run.

Related topics

help

Displays a list of available commands.

Parameters

To view help for a specific command, add the command as a parameter.

Example: `help fetch-files`

Related topics

compile-app

Runs HTML5 package compilation.

This command performs the following steps:

1. Runs the **npm install** command in the application folder.

2. Runs the **npm run build** command in the package folder.

3. Creates the output in subdirectory `dist`
.The output is stored as a zip file in the database.

Parameters

Login parameters:

- `/conn <database connection>`: Specifies the database to connect to.

- `/dialog <dialog authentication>`: Specifies the dialog authentication.

Optional parameter:

- `/conndialog <option>`: Specifies whether a login window is displayed for the database connection. The following options are possible:

  - `off`: The login window is not shown. If the database is not connected, an attempt is made to establish a connection.

  - `show`: The login window is shown (even is a database is already connected) and the new connection replaces the old one.

  - `fallback` (default): The current database connection is used. If the database is not connected, an attempt is made to establish a connection.

- `/factory <target system>`: Specifies the target system for the connection. Enter this parameter if you want to establish a connection to the application server.

- `/workspace <path to working directory>`: Specifies the working directory. This folder contains the application to be compiled. This folder normally contains the `package.json` file of the application. If you do not enter anything here, the current directory is used.

- `/app <application project name>`: Specifies which application project to compile. If you do not specify anything here, all application projects are compiled.

- `/branch <compilation branch ID>`: Saves the compiled result under a specified compilation branch. You must also specify the parameter `-D`.

- `-D`: Runs debug compilation.

- `N`: Prevents saving to the database.

- `/copyto <file path>`: Saves the result of the compilation as ZIP files in a folder.

- `/exclude <module name>`: Omits packages of a module at compile time (for example, **AOB**).

Related topics

- ImxClient command line program on page 93
- ImxClient command overview on page 93

compile-api

Compiles the API and saves the result to the database.

Parameters

Login parameters:

- `/conn <database connection>`: Specifies the database to connect to.
- `/dialog <dialog authentication>`: Specifies the dialog authentication.

Optional parameter:

- `/conndialog <option>`: Specifies whether a login window is displayed for the database connection. The following options are possible:
  - `off`: The login window is not shown. If the database is not connected, an attempt is made to establish a connection.
  - `show`: The login window is shown (even is a database is already connected) and the new connection replaces the old one.
  - `fallback` (default): The current database connection is used. If the database is not connected, an attempt is made to establish a connection.
- `/factory <target system>`: Specifies the target system for the connection. Enter this parameter if you want to establish a connection to the application server. Example: `QBM.AppServer.Client`.
- `/solution <solution project file path>`: Specifies the path to the solution project file to use. If you leave this parameter empty, a database project is used.
- `/mode <compile mode>`: Specifies the compilation mode:
  - `normal`: Runs a complete compilation (default mode).
  - `nostore`: Assemblies are not saved in the database.
  - `nocompile`: Creates only C# code. The compilation is not run.
  - `nocodegen`: Runs only the compilation. No C# code is generated.
- `-E`: Enables extended checks.
- `-D`: Compiles with debug information.
- `/csharpout <folder path>`: Specifies the path to the folder in which you want to save the C# files.
- `/copyapi <folder path>`: Specifies where to copy the `imx-api.tgz` to.
- `/copyapidll <API DLL path>`: Specifies which API DLL file to use. The `/solution` and `/branch` parameters are ignored if you use this parameter.
- `/branch <compilation branch ID>`: Saves the compiled result under a specified compilation branch. You must also specify the parameter `-D`.

- `/nowarn <error1,error2,...>`: Specifies which errors are ignored during compilation. Enter the codes for the warnings, separated by commas.
- `/warnaserror <error1,error2,...>`: Specifies which warnings are displayed as errors during compilation. Enter the codes for the warnings, separated by commas.

Related topics

- ImxClient command line program on page 93
- ImxClient command overview on page 93

repl

Starts the ImxClient command line tool in REPL mode.

In this mode, the following actions are performed in an infinite loop:

- Read commands from **stdin.**
- Forward commands to the relevant plugin.
- Output the results of processing to **stdout.**

Related topics

- ImxClient command line program on page 93
- ImxClient command overview on page 93

branch

Manages compilation branches in the database.

Parameters

If the command is called without specifying any parameters, this command outputs the total number of compilation branches and their IDs from the database.

Login parameters:

- `/conn <database connection>`: Specifies the database to connect to.
- `/dialog <dialog authentication>`: Specifies the dialog authentication.

Optional parameter:

- `/conndialog <option>`: Specifies whether a login window is displayed for the database connection. The following options are possible:

- off: The login window is not shown. If the database is not connected, an attempt is made to establish a connection.

- show: The login window is shown (even is a database is already connected) and the new connection replaces the old one.

- fallback (default): The current database connection is used. If the database is not connected, an attempt is made to establish a connection.

- /id <compilation branch ID>: Queries the field name and corresponding values of a compilation branch.

- -c /id <new compilation branch ID>: Creates a new compilation branch.

  To also specify a description of the compilation branch, use /description <description>.

- -d /id <compilation branch ID>: Deletes a compilation branch.

Related topics

connect

Establishes a database connection.

If a connection to a database has already been established, this is closed and a new connection is then established.

Parameters

Login parameters:

- /conn <database connection>: Specifies the database to connect to.

- /dialog <dialog authentication>: Specifies the dialog authentication.

Optional parameter:

- /conndialog <option>: Specifies whether a login window is displayed for the database connection. The following options are possible:

  - off: The login window is not shown. If the database is not connected, an attempt is made to establish a connection.

  - show: The login window is shown (even is a database is already connected) and the new connection replaces the old one.

  - fallback (default): The current database connection is used. If the database is not connected, an attempt is made to establish a connection.

- /factory <target system>: Specifies the target system for the connection. Enter this

parameter if you want to establish a connection to the application server. Example: `QBM.AppServer.Client`.

Related topics

install-apiserver

Installs an API Server on the local Internet Information Services (IIS).

Parameters

Login parameters:

- `/conn <database connection>`: Specifies the database to connect to.
- `/dialog <dialog authentication>`: Specifies the dialog authentication.

Required parameters:

- `/app <application name>`: Specifies which name is used for the application (for example, in the brower's titlebar).
- `/sessioncert <certificate thumbprint>`: Specifies which (installed) certificate is used for creating and verifying session tokens.

  TIP: For example, to obtain a certificate thumbprint, you can use the **Manage computer certificates** Windows function and find the thumbprint through the certificate's detailed information.

Optional parameter:

- `/conndialog <option>`: Specifies whether a login window is displayed for the database connection. The following options are possible:
  - `off`: The login window is not shown. If the database is not connected, an attempt is made to establish a connection.
  - `show`: The login window is shown (even is a database is already connected) and the new connection replaces the old one.
  - `fallback` (default): The current database connection is used. If the database is not connected, an attempt is made to establish a connection.
- `-u`: Allows insecure HTTP connections to the API Server website. By default, the API Server website can only be opened over an encrypted connection.
- `/site <site name>`: Specifies the website on the IIS under which the web application will be installed. If you do not enter anything, the website is found automatically (normally **Default website**).

- `/searchservice <URL>`: Specifies the application server's URL that the search service you want to use is hosted on.

  NOTE: If you would like to use the full text search, then you must specify an application server. You can enter the application server in the configuration file at a later date.

Related topics

run-apiserver

Starts or stops a self-hosted API Server.

This command requires a database connection.

Parameters

Login parameters:

- `/conn <database connection>`: Specifies the database to connect to.
- `/dialog <dialog authentication>`: Specifies the dialog authentication.

Optional parameter:

- `/conndialog <option>`: Specifies whether a login window is displayed for the database connection. The following options are possible:
  - `off`: The login window is not shown. If the database is not connected, an attempt is made to establish a connection.
  - `show`: The login window is shown (even is a database is already connected) and the new connection replaces the old one.
  - `fallback` (default): The current database connection is used. If the database is not connected, an attempt is made to establish a connection.
- `/factory <target system>`: Specifies the target system for the connection. Enter this parameter if you want to establish a connection to the application server. Example: `QBM.AppServer.Client`.
- `-S`: Stops the API Server.
- `/baseaddress <URL with port>`: Specifies the web application's root URL and port.
- `/baseurl <root URL>`: Specifies the web application's URL.
- `/branch <compilation branch ID>`: Specifies the compilation branch with the API you want to start the API Server for.

- `/asmdir <filepath>`: Loads the ZIP files containing the API DLLs and HTML files from the specified directory.
- `/apidll <filename>`: Loads the API from the specified file instead of the database.
- `-D`: Loads debug assemblies.
- `-C`: Compiles the API with source data from the database.
- `-T`: Queries the status of the current API Server.
- `-B`: Locks the console.
- `/compile <solution file path>`: Compiles the API with source data from the given (local) solution project.
- `/excludedMiddlewares <middleware name1, middleware name2, ...>`: Specifies which middleware services are not be made available. Enter multiple values separated by a comma.

Related topics

- ImxClient command line program on page 93
- ImxClient command overview on page 93

fetch-files

Loads all files of the HTML Development machine role from the database and saves them in a local file.

Parameters

Login parameters:

- `/conn <database connection>`: Specifies the database to connect to.
- `/dialog <dialog authentication>`: Specifies the dialog authentication.

Optional parameter:

- `/conndialog <option>`: Specifies whether a login window is displayed for the database connection. The following options are possible:
  - `off`: The login window is not shown. If the database is not connected, an attempt is made to establish a connection.
  - `show`: The login window is shown (even is a database is already connected) and the new connection replaces the old one.
  - `fallback` (default): The current database connection is used. If the database is not connected, an attempt is made to establish a connection.
- `/factory <target system>`: Specifies the target system for the connection. Enter this parameter if you want to establish a connection to the application server.

Example: `QBM.AppServer.Client`.

- `/workspace <working directory path>`: Specifies the working directory where the files should be placed. If you do not enter anything here, the current directory is used.

- `/targets <target1;target2;...>`: Specifies which machine roles you want to use. If you leave this empty, the **HTML Development** machine role is used.

Related topics

push-files

Saves files that you have changed locally back to the database.

Parameters

Login parameters:

- `/conn <database connection>`: Specifies the database to connect to.

- `/dialog <dialog authentication>`: Specifies the dialog authentication.

Optional parameter:

- `/conndialog <option>`: Specifies whether a login window is displayed for the database connection. The following options are possible:

  - `off`: The login window is not shown. If the database is not connected, an attempt is made to establish a connection.

  - `show`: The login window is shown (even is a database is already connected) and the new connection replaces the old one.

  - `fallback` (default): The current database connection is used. If the database is not connected, an attempt is made to establish a connection.

- `/factory <target system>`: Specifies the target system for the connection. Enter this parameter if you want to establish a connection to the application server. Example: `QBM.AppServer.Client`.

- `/targets <target1;target2;...>`: Specifies which machine roles you want to use. If you leave this empty, the **HTML Development** machine role is used.

- `/workspace <folder path>`: Specifies the working directory where the files are located that have been modified and are now to be stored in the database.

- `/tag <uid>`: Specifies the UID of a change tag.

- `/add <file1;file2;...>`: Specifies which new database files are added. Use relative paths.

- `/del <file1;file2;...>`: Specifies which database files are deleted. Use relative paths.
- `-C`: Prevents the saving of changed files and saves only new files, and deletes files from the database.

Related topics

get-apistate

Queries the compilation status of the API in the database.

Parameters

Login parameters:

- `/conn <database connection>`: Specifies the database to connect to.
- `/dialog <dialog authentication>`: Specifies the dialog authentication.

Optional parameter:

- `/conndialog <option>`: Specifies whether a login window is displayed for the database connection. The following options are possible:
  - `off`: The login window is not shown. If the database is not connected, an attempt is made to establish a connection.
  - `show`: The login window is shown (even is a database is already connected) and the new connection replaces the old one.
  - `fallback` (default): The current database connection is used. If the database is not connected, an attempt is made to establish a connection.
- `/factory <target system>`: Specifies the target system for the connection. Enter this parameter if you want to establish a connection to the application server.
  Example: `QBM.AppServer.Client`.
- `/branch <compilation branch ID>`: Queries the compilation status of the API saved under this compilation branch.
- `/htmlapp <name of the HTML package>`: Returns data for the specified HTML package.
- `-D`: Returns data for debug assemblies.
- `-R`: Returns data for release assemblies.

Related topics

get-filestate

Compares the local file structure with the file structure in the database.

Using the **QBM | ImxClient | get-filestate | NewFilesExcludePatterns** configuration parameter, you can define which files are excluded from the synchronization. This prevents excessive load during synchronization. The node_modules and imx-modules folders are excluded from the synchronization by default.
You can adjust the configuration parameters in the Designer. When setting the rules, use the formats defined here:
https://docs.microsoft.com/en-us/dotnet/api/microsoft.extensions.filesystemglobbing.matcher
Use the **|** character to delimit multiple entries.

> NOTE: This configuration parameter is generally only used to exclude new files from the synchronization. Files that already exist in the database are not taken into account.

Parameters

Login parameters:

- /conn <database connection>: Specifies the database to connect to.
- /dialog <dialog authentication>: Specifies the dialog authentication.

Optional parameter:

- /conndialog <option>: Specifies whether a login window is displayed for the database connection. The following options are possible:
  - off: The login window is not shown. If the database is not connected, an attempt is made to establish a connection.
  - show: The login window is shown (even is a database is already connected) and the new connection replaces the old one.
  - fallback (default): The current database connection is used. If the database is not connected, an attempt is made to establish a connection.
- /factory <target system>: Specifies the target system for the connection. Enter this parameter if you want to establish a connection to the application server. Example: QBM.AppServer.Client.
- /targets <target1;target2;...>: Specifies which machine roles you want to use. If you leave this empty, the **HTML Development** machine role is used.
- /workspace <directory path>: Specifies the working directory where the files you

want to match are located. If you do not enter anything here, the current directory is used.

Related topics

- ImxClient command line program on page 93
- ImxClient command overview on page 93

setup-web

Installs necessary files for the development of TypeScript clients.

Parameters

Login parameter:

- `/conn <database connection>`: Specifies the database to connect to.
- `/dialog <dialog authentication>`: Specifies the dialog authentication.

Optional parameter:

- `/conndialog <option>`: Specifies whether a login window is displayed for the database connection. The following options are possible:
    - `off`: The login window is not shown. If the database is not connected, an attempt is made to establish a connection.
    - `show`: The login window is shown (even is a database is already connected) and the new connection replaces the old one.
    - `fallback` (default): The current database connection is used. If the database is not connected, an attempt is made to establish a connection.
- `/factory <target system>`: Specifies the target system for the connection. Enter this parameter if you want to establish a connection to the application server. Example: `QBM.AppServer.Client`.
- `/workspace <API project path>`: Specifies the working directory where the files are to be installed. If you do not enter anything here, the current directory is used.
- `/app <application project name>`: Specifies which application to compile. If you do not specify anything here, all application projects are compiled.
- `/branch <Compilation branch ID>` Specifies the compilation branch for installing the files.

If you do not use any of the following parameters, all the install steps are carried out. Once you use one of the parameters, only the corresponding install steps are carried out. You can uses multiple parameters.

- `-A`: Create a link to the `assets` folder.
- `-L`: Initialize the libraries.

- `-P`: Binds Plugins to the web application.

Related topics

- ImxClient command line program on page 93
- ImxClient command overview on page 93

setup-workspace

Sets up the Angular working directory.

Parameters

Optional parameter:

- `/path <working directory path>`: Specifies the working directory. If you do not enter anything here, the current directory is used.

Related topics

- ImxClient command line program on page 93
- ImxClient command overview on page 93

workspace-info

Queries the state of the Angular working directory (existing applications and last API client update).

Parameters

Optional parameter:

- `/workspace`: Specifies which working directory to query. If you do not enter anything here, the current directory is used.

Related topics

- ImxClient command line program on page 93
- ImxClient command overview on page 93

check-translations

Search for captions (multilingual captions) with missing translations in a particular folder and its subfolders.

Parameters

Login parameters:

- `/conn <database connection>`: Specifies the database to connect to.
- `/dialog <dialog authentication>`: Specifies the dialog authentication.

Required parameters:

- `/path <path to folder>`: Specifies the path to the folder you want to check.

Optional parameter:

- `/conndialog <option>`: Specifies whether a login window is displayed for the database connection. The following options are possible:
  - `off`: The login window is not shown. If the database is not connected, an attempt is made to establish a connection.
  - `show`: The login window is shown (even is a database is already connected) and the new connection replaces the old one.
  - `fallback` (default): The current database connection is used. If the database is not connected, an attempt is made to establish a connection.
- `/factory <target system>`: Specifies the target system for the connection. Enter this parameter if you want to establish a connection to the application server. Example: `QBM.AppServer.Client`.

Related topics

- ImxClient command line program on page 93
- ImxClient command overview on page 93

version

Shows the version of the ImxClient command line program in use.

Related topics

- ImxClient command line program on page 93
- ImxClient command overview on page 93

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

Index

SQL files 87

SSO 81

start 22

start page 34

open 34

StartIndex 15

status bar 31

meaning of the colors 31

Swagger 92

T

tab 63

close 63

manage 63

open 63

save 63

task window 70

display 71

open 71

tasks 70

test

API 75

compiling 69

database query 62

SQL expression 62

text 54-58

token 12

toolbar 30

type member 84

U

UI 25

undo 34, 64

URL parameter 13

usage

change label 48

user interface 25

V

version control 72

Visual Studio 76