One Identity Manager 8.2.1

HTML5 Development Guide

Legend

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

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

One Identity Manager HTML5 Development Guide
Updated - 03 June 2022, 10:26
Version - 8.2.1

Contents

1

About this guide

This guide shows web developers how to view One Identity Manager HTML applications as code and how to understand their internal functionality.

To do this, you have the following options:

- Use existing HTML applications from the One Identity GitHub repository as templates (see Developing HTML applications with the One Identity GitHub repository on page 7).
- Use the Visual Studio Code extension (see Developing HTML applications with the Visual Studio Code extension on page 16).

Available documentation

The online version of One Identity Manager documentation is available in the Support portal under Technical Documentation. You will find videos with additional information at www.YouTube.com/OneIdentity.

2

Architecture of One Identity Manager HTML applications

One Identity Manager manages a folder structure that contains the source files for all HTML applications. In the data model, this folder structure is stored as part of the automatic software updates and it is assigned the machine role **HTML Development**.

You can store this file structure locally in a development folder. The folders for database modules are stored at the top level. Below that you will find a folder for each HTML application. See section Creating a development folder on page 18 to find out how to create a development folder.

The HTML applications are structured as nodeJS applications that use the **Angular** framework. Generally, any HTML applications that can be compiled as nodeJS applications are supported.

HTML applications use the API Client to communicate with the One Identity Manager API. The API Client is an npm library that is automatically generated and stored to the database during API compilation. The API Client controls all network access on the API Server.

Much of the logic for HTML applications is realized by plugins that can be used independently of a specific HTML application. You can also use this logic for your own HTML applications.

Developing HTML applications with the One Identity GitHub repository

You can develop your own HTML applications using the source code of a default HTML applications as a template.

The source code of default HTML applications is available in a GitHub repository at the following URL: https://github.com/OneIdentity/IdentityManager.Imx

Angular workspace architecture

The One Identity GitHub repository contains the source code for the HTML applications included in One Identity Manager.

It is a monorepo that contains the Angular workspace, which consists of applications and libraries.

Each Angular library and application belongs to a folder in the `projects` directory. The Angular workspace is defined in the `angular.json` file.

Table 1: Angular libraries

Name	Type	Dependencies within the workspace
qbm	Angular library	none
qer	Angular library	qer
tsb	Angular plugin library	qbm, qer
att	Angular plugin library	qbm, qer
rms	Angular plugin library	qbm, qer

Name	Type	Dependencies within the workspace
aad	Angular plugin library	qbm, qer, tsb
aob	Angular plugin library	qbm, qer
uci	Angular plugin library	qbm, qer
cpl	Angular plugin library	qbm, qer
dpr	Angular plugin library	qbm
o3t	Angular plugin library	qbm, qer, tsb
pol	Angular plugin library	qbm, qer

Each Angular library belongs to the One Identity Manager module of the same name.

An Angular library behaves like a regular compile-time dependency.
A plugin library is loaded dynamically at runtime. This is specified in the plugin's `imx-plugin-config.json` files.

Table 2: Angular applications

Name	Description	Project type	Static dependencies
qbm-app-landingpage	API Server landing page and server management	Angular application	qbm
qer-app-portal	Web Portal	Angular application	qbm, qer
qer-app-operationssupport	Operations Support Web Portal	Angular application	qbm, qer
qer-app-pwdportal	Password Reset Portal	Angular application	qbm, qer
arc-app-certaccess	CertAccess Web Portal	Angular app	Various

Setting up a working environment for the One Identity GitHub repository

In this section, you will discover how to set up your working environment for using the One Identity GitHub repository. This will allow you to develop your own web applications.

Prerequisites:

- You have a valid GitHub account (see https://github.com/).

To set up your working environment

1. Request access to the One Identity GitHub repository. The One Identity GitHub repository is available to you under the following URL: https://github.com/OneIdentity/IdentityManager.Imx
2. Create a fork of the One Identity GitHub repository (see https://docs.github.com/en/ge t-started/quickstart/fork-a-repo).
3. Install npm (see https://docs.npmjs.com/downloading-and-installing-node-js-and-npm).
4. Run the following command in a command line program:

```
npm install -g @angular/cli
```

Accessing the API

The Typescript API client is used to access the API.

For each One Identity Manager module that provides API services, an `imx-api-<module name>.tgz` Angular library exists as an NPM package in the `imx-modules` folder of the repository.

NOTE: The API client libraries are not dependent on Angular and thus can be used from any JavaScript program.

The TypeScript API clients consist of several parts:

- **Endpoint-based methods**:

  The **V2Client** class of the API client contains one method for each API endpoint. The method name is generated with the following format:

```
<URL path of the API endpoint>_<HTTP method>
```

Example

The **GET portal/serviceitems** method becomes the **portal_serviceitems_get** typescript method.

- **Entity-based methods**:

  The **TypedClient** class of the API client contains a wrapper class for each entity-based API method that allows entities to be loaded and stored.

  Example of methods for interactive entities

  For the **portal/serviceitems/interactive** method there is the **TypedClient.PortalServiceItemsInteractive_byid** property of the **PortalServiceItemsInteractive_byidWrapper** class type.

  Depending on the scope of operations supported by an API method, the following methods are available for the wrapper class:

  - The **createEntity** method is used to create a new entity.
  - The **Get_byid** method is used to load an interactive entity from the API Server. The API Server only supports loading a single object per query from the database as an interactive entity. The primary key values of the object must be specified as method parameters.
  - The **Put** and **Post** methods are used to store entities with the PUT operation. These methods must not be called directly, but are controlled by the **commit()** method of the IEntity interface.

Encapsulation as a service

Access to the API is encapsulated in a separate Angular service for each Angular library. This Angular service can be imported into custom classes:

- In the **qbm** Angular library, the service is called **imx_SessionService**.
- In the **qer** Angular library, the service is called **QerApiService**.
- In all other Angular libraries, the service is called **ApiService**.

Creating and editing your own HTML applications

To create and edit your own HTML applications, you can modify Angular libraries and add plugins to the API Server.

Detailed information about this topic

- Customizing libraries on page 11
- Adding plugins on page 11

Customizing libraries

If you modify Angular library code, you must create and deploy custom versions of all Angular applications that will use the modified Angular library.

For example, if you modify the **qer** Angular library, you must also compile the **qer-app-portal**, **qer-app-operationsupport**, and **qer-app-pwdportal** Angular applications, because all of these applications contain the **qer** Angular library.

If you modify Angular plugin library code, you must create and deploy a separate version of the Angular plugin library itself and all Angular plugin libraries that depend on it.

For example, if you modify the **tsb** Angular plugin library, you must also compile the **aad** and **o3t** Angular plugin libraries because these Angular plugin libraries contain the **tsb** Angular plugin library.

Adding plugins

Plugins are Angular libraries that are dynamically loaded at runtime. The plugins are managed by the API Server. Plugins are automatically detected by the API Server by searching the program directory for files named `imx-plugin-config.json`.

The following sample file specifies that the Angular plugin library `ccc` should be loaded into the **qer-app-portal application**. The name of the Angular module to be instantiated is **CustomConfigModule**.

```
{
   "qer-app-portal":[
      {
         "Container":"ccc",
         "Name":"CustomConfigModule"
      }
```

```
    ]
}
```

To add a plugin

1. On the API Server, create the file `imxweb\<Angular plugin library name>\imx-plugin-config.json` with the following content:

```
{
   "<HTML application name>":[
      {
         "Container":"<Angular plugin library name>",
         "Name":"<Angular module name>"
      }
   ]
}
```

2. Import the file into your One Identity Manager database using the Software Loader and assign it to the **API Server** machine role. For more information on importing files using the Software Loader, see the *One Identity Manager Operational Guide*.
3. (Optional) To check if the HTML application loads the plugin correctly, open the **<API Server URL>/imx/applications** URL and check that the corresponding plugin appears at the HTML application position in the list.

Checking for missing translations

You can check HTML applications for missing translations using the ImxClient command line program. For more information about the ImxClient command line program, see the *One Identity Manager API Development Guide*.

To check an HTML application for missing translations

1. Start the ImxClient command line program.
2. In the folder you want to check for missing translations, run the **check-translations** command.

   This creates a report. The report shows you any files containing texts that have not yet been translated or have not been translated in full.
3. (Optional) To add translation keys and translations, use the Designer program. For more information about translations, see the *One Identity Manager Configuration Guide*.

Registering HTML applications

To deploy new HTML applications for use and display them on the API Server's homepage, you must add the HTML applications to the database.

To add an HTML application to the database

1. Start the Designer program.
2. Connect to the relevant database.
3. In the navigation, click the **Base data** > **Security settings** > **HTML applications** category.
4. On the menu bar, click **(Create a new object)**.
5. Click the new entry in the list.
6. In the **Properties** view, enter the HTML application data in the respective fields. Enter at least the following information:
   - **Display name**: Enter a name for the HTML application.
   - **HTML application**: Enter the path `CCC/<name of your HTML application>`.
   - **Precompiled**: Set the value to **True**.

Compiling and deploying Angular projects

To make an Angular project available through the API Server, you must compile the Angular project and compress it into a ZIP file.

To compile and deploy an Angular project

1. Start a command line program.
2. Change to the Angular workspace directory.
3. Run the following command:

   ```
   ng build <project name>
   ```

4. Compress the contents of the directory containing the compilation (usually `dist/<project name>`) into a ZIP file named `Html_<project name>.zip`.
5. Copy the ZIP file to the `imxweb\custom` subfolder of your workspace.
6. Import the ZIP file into your One Identity Manager database using the Software Loader and assign it to the **API Server** machine role. For more information on importing files using the Software Loader, see the *One Identity Manager Operational Guide*.

Debugging

Running and debugging HTML applications is possible with the Angular CLI toolchain's standard tools.

For example, you can use the `ng serve qer-app-portal` command to debug the Web Portal HTML application.

To debug an HTML application

1. Host the API Server locally (see Hosting API Servers locally on page 14).
2. Start a command line program.
3. Change to the Angular workspace directory.
4. Run the following command:

   ```
   npm run start <HTML application>
   ```

   This starts a web server that is accessible by default under `http://localhost:4200` and hosts the HTML application.
5. Start debugging in an appropriate development environment (for example, Visual Studio Code).

Hosting API Servers locally

To debug and develop an HTML application you need an instance of an API Server for connecting HTML applications. To do this, you can host an API Server locally.

NOTE: HTML applications connect with the API Server through the URL defined in the HTML application's `environment.ts` file. The default URL that runs under a locally hosted API Server, is `http://localhost:8182`.

To host an API Server locally

1. Start a command line program.
2. Change to the Angular workspace directory.
3. Run the following command:

   ```
   imxclient.exe run-apiserver -B
   ```

Debugging with plugins

You can also debug with plugins. Debugging with plugins only works if the local API Server can find the plugin.

To debug a static Angular library

1. Host the API Server locally (see Hosting API Servers locally on page 14).
2. Start a command line program.
3. Change to the Angular workspace directory.
4. Run the following command:

   ```
   npm run build:watch <Angular library>
   ```

5. Start another command line program.
6. Change to the Angular workspace directory.
7. Run the following command:

   ```
   npm run start <HTML application>
   ```

   This starts a web server that is accessible by default under `http://localhost:4200` and hosts the HTML application.
8. Start debugging in an appropriate development environment (for example, Visual Studio Code).

To debug an Angular plugin library

1. Host the API Server locally (see Hosting API Servers locally on page 14).
2. Start a command line program.
3. Change to the Angular workspace directory.
4. Run the following command:

   ```
   npm run build:watch:dynamic <Angular plugin library>
   ```

5. Start another command line program.
6. Change to the Angular workspace directory.
7. Run the following command:

   ```
   npm run start <HTML application>
   ```

   This starts a web server that is accessible by default under `http://localhost:4200` and hosts the HTML application.
8. Start debugging in an appropriate development environment (for example, Visual Studio Code).

4

Developing HTML applications with the Visual Studio Code extension

In this section, you will find information on how to use with the One Identity Visual Studio Code extension.

You can use the Visual Studio Code extension to launch the ImxClient command line program directly in Visual Studio Code without any in-depth knowledge of the syntax:

- Connect to the database
- Create the development folder
- Manage and use compilation branches
- Update HTML applications with the same setup
- Compile a HTML application
- Update the base libraries (API Client)
- Check for missing translations

The Visual Studio Code extension also provides you with the following helpful information:

- Status of the current database connection
- Information about the ImxClient used
- Information about the API Client used
- Status of the development order
- Compilation branch display
- Information on the last translation check

Related topics

- Installing the One Identity Visual Studio Code extension on page 17
- Setting up a working environment with the Visual Studio Code extension on page 17
- Using the One Identity Visual Studio Code extension on page 19
- Creating applications on page 22

Installing the One Identity Visual Studio Code extension

To install the One Identity Visual Studio Code extension

1. Start the Visual Studio Code program.
2. In Visual Studio Code, click **Extensions** in the left navigation.
3. In **Extensions**, click **...** | **Install from VSIX**.
4. In the file browser, select the file `vcode-extension.vsix` in the One Identity Manager installation folder and click **Install**.
5. Restart Visual Studio Code.
6. In Visual Studio Code, click **Explorer** in the left navigation.
7. In the **Explorer**, navigate to **One Identity** | **Configuration**.
8. Next to **Imx client**, click **One Identity Manager: edit imx client path**.
9. In the file browser, select the file `ImxClient.exe` in the One Identity Manager installation folder and click **Open**.

Setting up a working environment with the Visual Studio Code extension

In this section, you will learn how to use the One Identity Visual Studio Code extension to set up your working environment. After setup, you will be able to use the Visual Studio Code extension and its functions without any issues.

Related topics

- Connecting to the One Identity Manager database on page 17
- Creating a development folder on page 18

Connecting to the One Identity Manager database

Many of the functions available to you in the Visual Studio Code extension require a connection to the One Identity Manager database.

To connect to the One Identity Manager database

1. In Visual Studio Code, click **Explorer** in the left navigation.
2. In **Explorer**, navigate to **One Identity** | **Workspace**.
3. Next to **Database**, click **One Identity Manager: connect to database**.

   The dialog to select the database connection is opened.
4. In the dialog, perform one of the following actions:
   - to use an existing connection to the One Identity Manager database, select it in the **Select a database connection** menu.

     - OR -
   - to create a new connection to the One Identity Manager database, click **Add new connection** and enter a new connection.
5. Select the authentication method and, under **Authentication method**, enter the login data for the database.
6. Click **Log in**.

Creating a development folder

You must perform the following actions to create a development folder:

- Download the HTML application folders from the database and save them as sub-folders in the development folder.
- Download the libraries from the database and save them in the sub-folder `imx-modules` in the development folder.
- The newest libraries are downloaded to the sub-folder `imx-modules` in the development folder. Skip this step if a connection to the database has not been established (also see Connecting to the One Identity Manager database on page 17).

To set up a One Identity Manager development folder

1. On the hard drive, create a folder that you want to use as a development folder.

   NOTE: The path to this folder and the folder itself, can only contain UTF-8 characters.
2. In Visual Studio Code, click **File** | **Open folder** in the toolbar.
3. In the file browser, select the folder that you created previously to use as the development folder.
4. In Visual Studio Code, click **Explorer** in the left navigation.
5. In **Explorer**, navigate to **One Identity** | **Workspace**.
6. Next to **Development folder**, click **One Identity Manager: set up development folder**.
7. Confirm this prompt with **Continue setup**.

Using the One Identity Visual Studio Code extension

In this section, you will learn how to use the One Identity Visual Studio Code extension.

Related topics

- Compilation branches on page 19
- Updating the API client on page 20
- Creating HTML applications automatically on page 21
- Compile HTML applications and save to database on page 21
- Check translations on page 22

Compilation branches

Use compilation branches to manage different versions of your compiled project and to store the versions in the database. You can select the required compilation branch, show available compilation branches, create compilation branches and delete compilation branches.

Related topics

- Selecting a compilation branch on page 19
- Creating compilation branches on page 20
- Deleting compilation branches on page 20

Selecting a compilation branch

To select and use a compilation branch

1. In Visual Studio Code, click **Explorer** in the left navigation.
2. In **Explorer**, navigate to **One Identity** | **Configuration**.
3. Next to **Branch ID**, click **One Identity Manager: edit branch id**.

   TIP: If a connection to the One Identity Manager database has not yet been established, you must set up the connection now (see Connecting to the One Identity Manager database on page 17).

   A menu with available compilation branches is opened.
4. In the menu, click the compilation branch you would like to use.

Creating compilation branches

To create a compilation branch

1. In Visual Studio Code, click **Explorer** in the left navigation.
2. In **Explorer**, navigate to **One Identity** | **Configuration**.
3. Next to **Branch ID**, click **Add branch**.

   An input field opens.
4. In the input field, enter a name for the compilation branch and press **Enter**.

   TIP: If a connection to the One Identity Manager database has not yet been established, you must set up the connection now (see Connecting to the One Identity Manager database on page 17).

Deleting compilation branches

To delete a compilation branch

1. In Visual Studio Code, click **Explorer** in the left navigation.
2. In **Explorer**, navigate to **One Identity** | **Configuration**.
3. Next to **Branch ID**, click **One Identity Manager: edit branch id**.

   TIP: If a connection to the One Identity Manager database has not yet been established, you must set up the connection now (see Connecting to the One Identity Manager database on page 17).

   A menu with available compilation branches is opened.
4. In the menu, click the configuration branch you would like to delete.
5. In **Explorer** next to **Branch ID**, click **Delete branch**.
6. Confirm this prompt with **OK**.

Updating the API client

You can update the base libraries and, therefore, the API client at any time.

To update the API Client

1. In Visual Studio Code, click **Explorer** in the left navigation.
2. In **Explorer**, navigate to **One Identity** | **Workspace**.
3. Next to **API Client**, click **One Identity Manager: update api client**.

   TIP: If a connection to the One Identity Manager database has not yet been estab-

lished, you must set up the connection now (see Connecting to the One Identity Manager database on page 17).

Creating HTML applications automatically

To recognize and create HTML applications automatically, check the folders available in your development folder. It involves the following steps:

- The sub-folder `assets` in the working directory is connected to the sub-folder `src/assets` in the application.
- The relevant plugins are connected and integrated.

NOTE: HTML applications are also automatically set up before each compilation.

To automatically create HTML applications

1. In Visual Studio Code, click **Explorer** in the left navigation.
2. In **Explorer**, navigate to **One Identity** | **Workspace** | **Development folder**.
3. Next to **Apps**, click **reload existing apps**.

   TIP: If a connection to the One Identity Manager database has not yet been established, you must set up the connection now (see Connecting to the One Identity Manager database on page 17).

Compile HTML applications and save to database

You can compile your HTML applications and save the results automatically to the One Identity Manager database.

To compile HTML applications and save to the database

1. In Visual Studio Code, click **Explorer** in the left navigation.
2. In **Explorer**, navigate to **One Identity** | **Workspace** | **Development folder** | **Apps**.
3. Click **Compile app** next to the HTML application that you would like to compile and save to the database.

   TIP: If a connection to the One Identity Manager database has not yet been established, you must set up the connection now (see Connecting to the One Identity Manager database on page 17).

   A debug compilation is run and the changes are saved to the enabled Compilation branch.

Check translations

In your HTML application, you can check the development folder for missing translations.

To check the development folder for missing translations

1. In Visual Studio Code, click **Explorer** in the left navigation.
2. In **Explorer**, navigate to **One Identity** | **Workspace** | **Development folder**.
3. Next to **Translations**, click **Check translations**.

   A report opens. The report shows you any files containing texts that have not yet been translated or have not been translated in full.

Creating applications

You can the use One Identity Visual Studio Code extension to create your HTML applications and link them into the existing system.

Follow these steps to create your HTML application:

1. Create the new HTML application.
2. Import the source files into the database.
3. Add the HTML application to the database.

To create a new HTML application

1. Install the One Identity Visual Studio Code extension (see Installing the One Identity Visual Studio Code extension on page 17).
2. Set up the work environment with the help of the One Identity Visual studio Code extension (see Setting up a working environment with the Visual Studio Code extension on page 17),
3. Add a new CCC folder to your work environment.
4. Ensure that you are using the correct version of Angular (9.0):
   a. Open a command line prompt.
   b. In the CCC folder, run the **ng version** command.
   c. Check the version number returned.

      TIP: If nothing is returned it means that Angular is not installed locally or globally.

      To install Angular globally, run the command: **npm install -g @angular/cli**.

      To install Angular locally, run the command: **npm install @angular/cli**.
5. In the CCC folder, run the command: **ng new <your HTML application name>**

6. In the QBM/OpsWeb folder, copy the imx-plugin-config.json file and add it to the CCC/<your HTML application name> folder.
7. In the CCC/<your HTML application name> folder, edit the (freshly copied) imx-plugin-config.json file:
   - Remove all apart **common** and **shared**.
8. In the CCC/<your HTML application name> folder, edit the package.json file:
   a. In the scripts section, add "build:debug": "ng build --prod --source-map".
   b. In the dependencies section, add dependencies to imx-api, imx-qbm-components, and imx-qbm-dbts.
   c. In the dependencies | rxjs section, change the version to ^6.3.3.
9. In the CCC/<your HTML application name> folder, edit the angular.json file:
   - In the projects | <your HTML application name> | architect | build | options subsection, add "outputPath": "dist".
   - In the projects | <your HTML application name> | architect | build | options subsection, add "preserveSymlinks": true.
10. In the CCC/<your HTML application name>/src folder, edit the index.html file:
    - In the head section, add <base href=".">.
11. In the CCC/<your HTML application name>/src folder, edit the environment.<your domain name>.ts and environment.ts files:
    - Enter one entry for each client URL:

      For environment.<you domain name>.ts: clientUrl: ''

      For environment.ts: clientUrl: 'http://localhost:8182'
12. In the CCC/<your HTML application name> folder, edit the tsconfig.json file:
    - In the compilerOptions | paths subsection, add "@shared/*": ["src/imx-plugins/QBM/shared/*"].
13. Use the One Identity Visual Studio Code extension to make sure that you are not using complication branches (see Selecting a compilation branch on page 19).
14. Use the One Identity Visual Studio Code extension to compile your HTML application (see Compile HTML applications and save to database on page 21).
15. Open your compiled HTML application with the URL: **$<API Server URL>/html/<your HTML application name>/**.

    NOTE: Here you must use the same name for the HTML application as stored in the package.json file.

To import source files in to the database

1. Go to your work environment.
2. Start the Software Loader program.
3. Click **Import into database**.
4. Click **Next**.

5. On the **Connect to database** page, select a database and enter the user credentials.
6. Click **Next**.
7. Select your work environment.
8. Mark all the files in the `CCC/<your HTML application name>` folder but exclude the following folders:
    - `node_modules`
    - `dist`
    - `.git`
    - `src/assets`
    - `src/imx-modules`
9. Click **Next**.
10. Confirm the prompt with **Yes**.
11. On the **Assign machine roles** page, mark all the file and enable the **HTML Development** machine role.
12. Click **Next**.
13. On the **Select change label** page, perform the following actions:
    - To not use a change label, click **Do not assign the files to a change label**.
    - To use a change label, click **Assign files to following change label**. Then click **...** and select the desired changed labels.
14. Click **Next**.
15. On the **Transferring files** page, once the files have been successfully transferred to the database, click **Next**.
16. On the **Wizard complete** page, click **Finish**.

To add the HTML application to the database

1. Start the Designer program.
2. Navigate to **Base data** | **Security settings** | **HTML applications**.
3. Select the **Object** | **New** menu item to add a new item.
4. Enter a display name and the **CCC/<your HTML application name>** path.

Related topics

- Installing the One Identity Visual Studio Code extension on page 17
- Selecting a compilation branch on page 19
- Compile HTML applications and save to database on page 21

About us

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