One Identity Manager 9.0

HTML5 Development Guide

One Identity Manager HTML5 Development Guide
Updated - 01 August 2022, 14:24
Version - 9.0

Contents

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.

You can use existing HTML applications from the One Identity GitHub repository as templates for this (see Developing HTML applications with the One Identity GitHub repository on page 6).

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.

Architecture of One Identity Manager HTML applications

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

Detailed information about this topic

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
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

Name	Description	Project type	Static dependencies
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@<Versionsnummer>
   ```

   NOTE: Refer to https://github.com/OneIdentity/IdentityManager.Imx to find out which version of Angular CLI you require.

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** property of the **PortalServiceItemsInteractive** wrapper 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

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:

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

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 user interface translations using the ImxClient command line program. For more information about the ImxClient command line program, see the *One Identity Manager API Development Guide*.

TIP: Translatable text is marked in the source code with the #LDS# prefix.

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 home page, 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.

   TIP: If the folder does not exist, create it.

6. Import the ZIP file into your One Identity Manager database using the Software Loader and assign it to the **Development and Testing** machine role. For more information on importing files using the Software Loader, see the *One Identity Manager Operational Guide*.

7. Copy the ZIP file to the `bin\imxweb\custom` subfolder of your IIS installation.

   TIP: If the folder does not exist, create it.

8. Import the ZIP file into your One Identity Manager database using the Software Loader and assign it to the **Business API Server** machine role.

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).

Detailed information about this topic

-
-

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).

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