FintechOS Platform 24.3

Administration Guide

TOC

FintechOS Platform Administration Guide

The FintechOS Platform is an innovation acceleration software that enables fast, plug & play, comprehensive digital transformation of companies that offer financial services.

It is a highly scalable technology that can be run from the cloud.

We recommend using one of the enterprise cloud providers that the FintechOS Platform is compatible with.

Deployment

IMPORTANT!
Starting with release 22.1, only FintechOS cloud deployments are supported for the FintechOS Platform.

System Requirements

FintechOS Platform System Requirements

Software minimum required version	18.2.x	20.1.x (Genie)	20.2.x (Pulsar)	22.1
.NET Framework	4.6.2	4.6.2	4.7.2	4.7.2
SQL Server	SQL Server 2012 (11.x)	SQL Server 2012 (11.x)	SQL Server 2012 (11.x)	SQL Server 2019
Windows Server	Windows Server 2012 R2	Windows Server 2012 R2	Windows Server 2012 R2	Windows Server 2019

Below are details about which Windows Server roles and features are required. They were determined on a Windows Server 2012 R2. You must determine the equivalents for your particular Windows Server version.

Required Server Roles

Web Server (IIS)

Required Features

- NET Framework 3.5 Features \ .NET Framework 3.5 (includes .NET 2.0 and 3.0)
- NET Framework 4.5 Features \ .NET Framework 4.5
- NET Framework 4.5 Features \ ASP.NET 4.5
- NET Framework 4.5 Features \ WCF Services \ HTTP Activation
- NET Framework 4.5 Features \ WCF Services \ TCP Port Sharing
- Windows PowerShell \ Windows PowerShell 4.0
- Windows Process Activation Service \ Process Model 17
- Windows Process Activation Service \ Configuration APIs

Required Web Server Role (IIS) / Role Services

- Web Server \ Common HTTP Features \ Default Document
- Web Server \ Common HTTP Features \ Directory Browsing
- Web Server \ Common HTTP Features \ HTTP Errors
- Web Server \ Common HTTP Features \ Static Content
- Web Server \ Common HTTP Features \ HTTP Redirection
- Web Server \ Health and Diagnostics \ HTTP Logging
- Web Server \ Performance \ Static Content Compression
- Web Server \ Performance \ Dynamic Content Compression
- Web Server \ Security \ Request Filtering
- Web Server \ Security \ Basic Authentication

- Web Server \ Security \ URL Authorization
- Web Server \ Security \ Windows Authentication
- Web Server \ Application Development \ .NET Extensibility 4.5
- Web Server \ Application Development \ Application Initialization
- Web Server \ Application Development \ ASP.NET 4.5
- Web Server \ Application Development \ ISAPI Extensions
- Web Server \ Application Development \ ISAPI Filters
- Web Server \ Application Development \ Server Side Includes
- Web Server \ Application Development \ WebSocket Protocol
- Web Server \ Management Tools \ IIS Management Scripts and Tools

FintechOS Portal System Requirements

FintechOS Portal can run on the following browsers, on both desktop and mobile devices:

Browser	Operating System
Google Chrome	Windows 10
Mozilla ESR	Windows 10
Mozilla Firefox	Windows 10
Microsoft Edge	Windows 10
Opera	Windows 10
Safari (desktop)	macOS - latest version
Safari (mobile)	IOS - latest version
Google Chrome (mobile)	Android - latest version

IMPORTANT!
Please be aware that FTOS-Studio is fully compatible only with Google Chrome!

We recommend that you use the latest major version available for the browser.

System Requirements for WebRTC Components

NOTE

Components that are using WebRTC impose a series of limitations for our services to work correctly.

WebRTC (Web Real-Time Communication) is a free, open-source project that provides web browsers and mobile applications with real-time communication (RTC) via simple application programming interfaces (APIs). It allows audio and video communication to work inside web pages by allowing direct peer-to-peer communication, eliminating the need to install plugins or download native apps. WebRTC is being standardized through the World Wide Web Consortium (W3C) and the Internet Engineering Task Force (IETF).

Its mission is to "enable rich, high-quality RTC applications to be developed for the browser, mobile platforms, and IoT devices, and allow them all to communicate via a common set of protocols".

Taking into account the versions supported by WebRTC and our 3rd party providers, please find below the list of supported browsers on different devices.

Desktops / Laptops:

Browser	Recommended Version	Supported Version
Google Chrome	Latest	79 and greater
Mozilla Firefox	Latest	71 and greater
Microsoft Edge	Versions 80 - 81	79 and greater
Safari	13.1 and greater	13.1 and greater
Opera	Latest	66 and greater
Internet Explorer 11	Not Supported	Not Supported

NOTE

As WebRTC is in constant development, please ensure to always use the latest version.

We recommend to use Google Chrome for the best overall experience.

Mobile Devices:

Operating System	Browser	Supported Version
Android (version 6.0 and greater)	Google Chrome	Latest
Android (version 6.0 and greater)	Mozilla Firefox	Not Supported
Android (version 6.0 and greater)	Opera	Latest
iOS (version 12 and greater)	Safari	2 most recent major versions
iOS	Google Chrome	Not Supported
iOS	Microsoft Edge	Not Supported
iOS	Mozilla Firefox	Not Supported
iOS	Opera	Not Supported

NOTE
WebRTC does not support Chrome on iOS devices. On iOS you can only use the Safari engine.

Cookie Specifications

The FintechOS Platform uses the following cookies:

Server-Side Encrypted Cookies

These are security cookies used for client-server authentication.

Cookie	Description	Expiry
.EBSCORE$1	Authentication cookie.	Session
.EBSCORE$1-CSRFToken	CSRF prevention token cookie.	Session
.EBSCORE$1_PartialToken	Used for authentication.	Session
IdToken	Identity token generated by the identity provider for the current user.	Session
AccessToken	Access token generated by the identity provider for the current user for accessing the platform resources.	IDP configurable (Access Token Lifespan)
RefreshToken	Session token generated by the identity provider for the current user session.	IDP configurable (SSO Session Idle)
RefreshExpTicks	Session token time-to-live expressed in ticks.	IDP configurable (SSO Session Idle)

Client-Side Cookies

These are client-side generated cookies that store a transient state. Prior to release 24.1.1, they were saved unencrypted in the Document Object Model cookies object. Starting with release 24.1.1, they have been migrated and are now either sent via request headers or stored in local storage, enforcing secure sessions for user data.

Cookie	Description	Expiry
.EBSCORE$1-CorrelationId	Browser session specific cookie used for logging.	Session
.EBSCORE$1-culture	Current language. Preserves language settings between user sessions.	365 days
.EBSCORE$1-ShowTooltipsOnForms	Show/hide tooltips in FintechOS Portal forms.	100 days
.EBSCORE$1-palette	Color palette settings for FintechOS Portal.	100 days
.EBSCORE$1-theme	Theme settings for FintechOS Portal.	100 days

Third Party Libraries

This section lists the third party libraries used by the FintechOS Platform.

Server Side Libraries

Library	Comments
AngleSharp	HTML Parser for .NET. Used to parse entities form the HTML structure.
Aspose.PDF	PDF features. Used by various SDK methods that work with PDF files.
Autofac	Dependency injection.
AutoMapper	Converts one object type to another.
AWSSDK.S3	Used for document storage.
Azure.Core	Used for document storage.

Library	Comments
Azure.Storage.Common	Used for document storage.
BundleTransformer.Core	CSS/JS minify.
BundleTransformer.SassAndScss	CSS/JS minify.
Dapper	Object mapper for .NET. Used to map database query results to the business model.
DocX	Word manipulation. Used by the DocumentReport service.
DotLiquid	Liquid templates for .NET. Used indirectly (by another third party library).
EPPlus	Excel spreadsheets for .NET.
Esprima.NET	JavaScript syntax parser for .NET. Used to parse client and server side scripts to identify specific elements (like EbsResource).
Flurl.Http.Signed	HTTP client library for .NET. Used to make HTTP calls.
GemBox.Document	Read, write, convert, and print document files (DOCX, DOC, PDF, RTF, HTML, and ODT) from .NET. Used to convert docx to pdf.
HtmlAgilityPack	HTML parser that builds a read/write DOM and supports plain XPATH or XSLT. Used in request validation.
ICSharpCode.SharpZipLib	Zip, GZip, Tar and BZip2 library for .NET.
Jint	JavaScript engine. Used in server automation scripts.
Microsoft.ClearScript	Adds scripting features to .NET applications. Supports V8 (Windows, Linux, macOS). Used for server-side execution and debugging.
Microsoft.CodeAnalysis.CSharp	
Microsoft.CodeDom.Providers.DotNetCompilerPlatform	
Newtonsoft.Json	JSON serialization.

Library	Comments
Namotion.Reflection	Advanced reflection APIs. Used indirectly (by another third party library).
NJsonSchema	JSON schema reader, generator and validator. Used to generate REST clients from Swagger definitions.
NSwag.CodeGeneration	JSON schema reader, generator and validator. Used to generate REST clients from Swagger definitions.
NuGet.Versioning	
Owin	Open Web Interface for .NET. Used for SignalR communication.
Quartz	Job scheduling system. Used by the scheduling agent.
Scriban	Content generation based on templates. Used to generate SMS/email messages.
Serilog	Logging library. Used for logging.
SharpScss	
SharpZipLib	
SixLabors.ImageSharp	Cross-platform, 2D Graphics library for .NET. Used for Exif info manipulation/clearing.
SonarAnalyzer.CSharp	
TimeZoneConverter	
VaultSharp	Used by the FintechOS Configuration Manager.
WebGrease	Optimize static files (like JavaScript, CSS) in a Web application.
WindowsAzure.Storage	
YamlDotNet	Parser and generator of YAML files. Used for integration with other systems based on YAML files.
Yarp.ReverseProxy	Reverse proxy for debugging.

Client Side Libraries

Library	Used For
axios	HTTP client.
bootstrap	JavaScript UI framework.
bootstrap-iconpicker	Iconpicker control.
cronstrue	Cron expresions.
crossroads	Routing library.
dagre.min.js	Joint JS dependency.
dx	DevExtreme controls library.
es6-promise.auto.js	Utility.
es6-shim.js	Utility.
graphlib.min.js	Joint JS dependency.
gridstack	Dashboard widgets, geometric layout.
hasher	Browser history utility.
head.min.js	Utility.
highcharts-more	Charts.
highlight	Code highlighting.
highstock	Charts.
intlTelInput.min.js	Phone number formatting.
joint	SVG utility.
jquery	HTML traversal and manipulation.
jquery.mixitup	Animated filtering (home shortcuts).
jquery-migrate	Jquery 2.x to 3.x compatibility.
jquery-toast-plugin	Toast messages.
jquery-ui	Jquery controls library.
js-cookie	Cookie handling.
lodash	Utility.
moment	Date utility.
moment-timezone	Moment plugin.
monaco-editor	Code editor.
mousetrap.js	Keyboard shortcuts.
powerbi	Power BI client.
react	JavaScript framework.
react-dom	
react-redux	React binding for Redux.
signals	Hasher dependency.
svg-pan-zoom	Pan and zoom SVG files.

Library	Used For
tinymce	WYSIWYG editor.
userpilot	User help and feedback. No longer used, but the capability was not removed yet.
vue	JavaScript framework mainly used for Power BI rendering.

API Configuration

The configuration keys for the FintechOS Platform APIs are stored in the "Configuration Manager" on page 46 at the `<kvSecretsEngine>/<environmentName>/<apiAppName>` path. Here, you can use the following secrets to configure the APIs.

FtosOpenApi

```
{
  "CachingTimeoutMinutes": "60",
  "FtosPortalUrl": "https://www.myDomain.com/portal/api/",
  "HttpRetryCounts": "3",
  "HttpRetryInitialDelaySeconds": "1",
  "HttpTimeoutSeconds": "120",
  "IdentityProvider": "Keycloak",
  "LogRequestsAndResponses": "false"
}
```

Key Path	Key Name	Key Value
kv/<environment>/<apiAppName>/FtosOpenApi	CachingTimeoutMinutes	Sliding timeout in minutes for cached entity metadata and actions. If no endpoints have been accessed for the designated number of minutes, new requests will be sent to the platform to retrieve entity metadata and actions information. (This is also when the endpoints for new entities will be rendered in the swagger definition)
kv/<environment>/<apiAppName>/FtosOpenApi	FtosPortalUrl	Should point to `<environmentName>/<portalName>/api`.
kv/<environment>/<apiAppName>/FtosOpenApi	HttpRetryCounts	Number of retry attempts if the API request fails with a 5xx or 408 status code. Default: 3.
kv/<environment>/<apiAppName>/FtosOpenApi	HttpRetryInitialDelaySeconds	Number of seconds between retry attempts. Default: 1.
kv/<environment>/<apiAppName>/FtosOpenApi	HttpTimeoutSeconds	Timeout in seconds for API requests.

Key Path	Key Name	Key Value
kv/<environment>/<apiAppName>/FtosOpenApi	IdentityProvider	• Keycloak - Authentication via Keycloak JWT tokens. If selected, you must also configure the "KeycloakApi" on the next page secret. • Ebs - **(Deprecated)** Legacy Ebs authentication using access tokens. If selected, you must also configure the "EbsCoreApi (deprecated)" on page 27 secret.

Key Path	Key Name	Key Value
kv/<environment>/<apiAppName>/FtosOpenApi	LogRequestsAndResponses	• true - Logs API and platform (Portal) requests and responses if the minimum severity level for logged messages is set to Info (see "Observability" on page 68). • false - Default. Disables API logging. **IMPORTANT!** API logging has a significant impact on performance. It should only be used for debugging purposes.

KeycloakApi

The KeycloakApi secret must be configured if the `kv/<environment>/<apiAppName>/FtosOpenApi/IdentityProvider` key is set to Keycloak.

```
{
  "Audience": "account",
  "ClientId": "admin-portal",
  "ClientSecret": "xxxxxxxxxxxxxxxxxxxxxxxxx",
  "HttpRetryCounts": "3",
  "HttpRetryInitialDelaySeconds": "1",
  "HttpTimeoutSeconds": "30",
```

```
  "Password": "xxxxxxxxxx",
  "RealmUrl":
"https://www.myDomain.com/auth/realms/fintechOSrealm",
  "UserName": "host"
}
```

Key Path	Key Name	Key Value
kv/<environment>/<apiAppName>/KeycloakApi	Audience	The claim aud or Audience extends from the JWT specification defined under RFC-7519. It allows the consuming party to validate if a particular JWT is meant for them or not.
kv/<environment>/<apiAppName>/KeycloakApi	RealmUrl	Keycloak realm URL.
kv/<environment>/<apiAppName>/KeycloakApi	HttpTimeoutSeconds	Timeout in seconds for the Keycloak JWT token request.
kv/<environment>/<apiAppName>/KeycloakApi	HttpRetryCounts	Number of retry attempts if the Keycloak request fails with a 5xx or 408 status code. Default: 3.

Key Path	Key Name	Key Value
kv/<environment>/<apiAppName>/KeycloakApi	HttpRetryInitialDelaySeconds	Number of seconds between retry attempts. Default: 1.
kv/<environment>/<apiAppName>/KeycloakApi	UserName, Password, ClientId, ClientSecret, GrantType	Credentials for the Keycloak JWT token retrieval.

EbsCoreApi (deprecated)

> **IMPORTANT!**
> This type of authentication is deprecated as it has been replaced with JWT-based authentication via the "FintechOS Identity Provider" on page 121. This information is provided only for backward compatibility.

The EbsCoreAPI secret must be configured if the `kv/<environment>/FtosOpenApi/IdentityProvider` key is set to EbsCoreApi.

```
{
    "AdminClientId": "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx",
    "AdminUser": "username",
    "AdminPassword": "password"
}
```

Key Path	Key Name	Key Value
kv/<environment>/<apiAppName>/EbsCoreApi	AdminClientId, AdminUser, AdminPassword.	Credentials used to retrieve the API access token from the GetToken endpoint. Using this token, entity data is then dynamically loaded in order to generate the OpenAPI/Swagger definition.

System Digital Solution Packages Installation

The steps below describe how to perform both an automatic installation and a manual import of a FintechOS SysPack.

Depending on the FintechOS platform version that you want to install, make sure the correct SysPack type is applied:

1. For standard FintechOS infrastructure installation use Standard Syspacks.
2. For Professional/ Enterprise FintechOS infrastructure installation use the following SysPacks:
    a. Banking environments: Professional Banking SysPacks
    b. Insurance environments: Professional Insurance SysPacks

> **NOTE**
> SysPacks are mutually exclusive. The platform installation requires only one SysPack type.

Below are the components for each FintechOS SysPack.

Package	Description
01 FTOS DFP Common.zip	FTOS DFP Common Data Model
02 FTOS Content Templates.zip	FTOS Content Templates
02 FTOS Foundation.zip	FTOS Foundation
02 FTOS Project HyperPersonalization.zip	FTOS Hyperpersonalization Processor Data Model
02 FTOS Versioning.zip	FTOS Versioning PreReq
03 FTOS Project Campaign.zip	FTOS Campaign Management Data Model
FTOS Project Cognitive Processor Client.zip	FTOS OCR and Onfido Processors Scripts
FTOS Project Cognitive Processor Operator.zip	N/A

Package	Description
FTOS Project Data Governance Consent Management.zip	N/A
FTOS Project Data Governance Sensitive Data.zip	FTOS Data Governance Sensitive Data Management Data Model
FTOS Project Digital Review.zip	N/A
FTOS Project Esign Processor.zip	FTOS Esign Processor Scripts
FTOS Project Integration.zip	FTOS Integration Scripts

HINT
Details about each component are in their .zip packages.

For an automatic installation, follow the steps described in the **SysPacks Automatic Installation** section.

IMPORTANT!
Starting with FintechOS Platform V20.2.9, the SysPack can be imported asynchronous. When importing the packages in an Azure environment, always use async syntax.

Prerequisites

In order to install the SysPacks, you need the latest FintechOS platform version installed, with the database configured. For specific steps, see the Installation page.

NOTE
When using **FtosSysPackageDeployer** with SQL Server Integrated Authentication make sure:
1. The Windows user running the above command has read/ write rights access to the FTOS database.
2. You run the command without the SQL username/ password parameters.

When using **FtosSysPackageDeployer** with SQL Server Build In Authentication make sure:
1. The login used has read/ write access to the FTOS database.
2. You run the above command with the SQL username/ parameters.

IMPORTANT!
Starting with v24.3.2, the size of digital solution packages is restricted at import to the default of 50MB. Navigate to **Configuration Manager > Studio > app-settings > DigitalSolutionPackageMaxSizeInMB** key to change the default to another size.

Pre-Installation Checklist

The SysPack has unique constraints on some of the standard entities like: `FTOS_DFP_FlowSettings`, `FTOS_DFP_ProcessorSettings`, `FTOS_VersionSettings`, `FTOS_VersionSettingsItem`, `FTOS_EntityStatusSettings`, `FTOS_MKT_AudienceSegments`, `FTOS_MKT_Audience`.

If you have already moved data using the Configuration Data Deployment Package menu, then you probably have already configured some unique constraints.

Before running the script, make sure you:

1. Disable the constraints that you have created on your environment, allowing the system to create the new ones after the SysPacks are imported.
2. Use the new **Configuration Data Definitions** imported with the SysPacks when you export the data.

Automatic Installation Steps

1. Download the desired SySDigitalSolutionPackages compatible with you platform version from the Release Hub.
2. Unzip the installation kits.
3. Use *FtosSysPackageDeployer* to install the Syspack as follows:
    - Locate the *FtosSysPackageDeployer* in the unzipped FintechOS installation kit at the following location: **<unzipped_install_archive>\Tools\FtosSysPkgDeployer**.

- Navigate to the location where you have unzipped the SysPack and copy the *FtosSysPackageDeployer* here. Let's call this location **<pckg_deployer_dir>**.
- Open `async install_SysPackDA.bat` to edit and replace the parameters described in the `install_SysPack.bat` Parameters Explanation section, with your own values.
- Right-click `async install_SysPackDA.bat` » Run as administrator.

`install_SysPackDA.bat` Parameters Explanation

For asynchronous import run the following command:

```
FtosSysPkgDeployer.exe -i –a -s <studio_url> -u <studio_
user_name> -p <studio_user_password> -z <db_Server> -v
<db_server_login_username> -k <db_server_login_password>
-d <db_name> -r <syspack_file_path>
```

Field	Description
<studio_url>	The web URL of the Innovation Studio installation, for example http://localhost/ftos_studio.
<studio_user_name>	The username of the Innovation Studio user under which this import is executed. The user has to exist in Innovation Studio prior to this operation
<studio_user_ password>	The password for the Innovation Studio user.
<db_server>	The name of the database server where the FintechOS installation database was created.
<DB_user>	The username of the SQL Server user with administration rights on the FintechOS installation database.
<db_server_login_ username>	The login username of the SQL Server user with administration rights on the FintechOS installation database.

Field	Description
<db_server_login_ password>	The password for the above mentioned SQL user.
<db_name>	The name of the database where the FintechOS Platform is deployed.
<syspack_file_path>	The physical path to the unzipped SysPack previously downloaded.

HINT
For more information about the deployment tool, please run FtosSysPackageDeployer.exe without any arguments to see the built-in help

Post-Installation Setup

After installing the .zip packages, access the 100_AfterImportManualCopy folder and follow the below steps:

1. Add the following images to the Upload EBS folder `<portal_EBS_folder>` (the Portal with operator flow):
    a. <syspack_file_path>**100_AfterImportManualCopy** \CopyToUploadEBS\emptyOCR.jpg
    b. <syspack_file_path>**100_ AfterImportManualCopy**\CopyToUploadEBS**emptyPhoto**.png
2. Copy the following folders over the FintechOS Portal installation directory for every Portal with back-office or B2C installed.
    a. <syspack_file_path>\ **100_AfterImportManualCopy** \FTOS Project Cognitive Processor Files\dcs-sdk-version**custom**
    b. <syspack_file_path>\ **100_AfterImportManualCopy** \FTOS Project Cognitive Processor Files\dcs-sdk-version**custom-on-demand**

c. Copy any other needed js files in the corresponding js folder.

d. For Onfido, follow theiInstructions from the **InstallGuideOCRWithOnfido v1.1** file.

Cognitive Processor Custom Folders Explanation

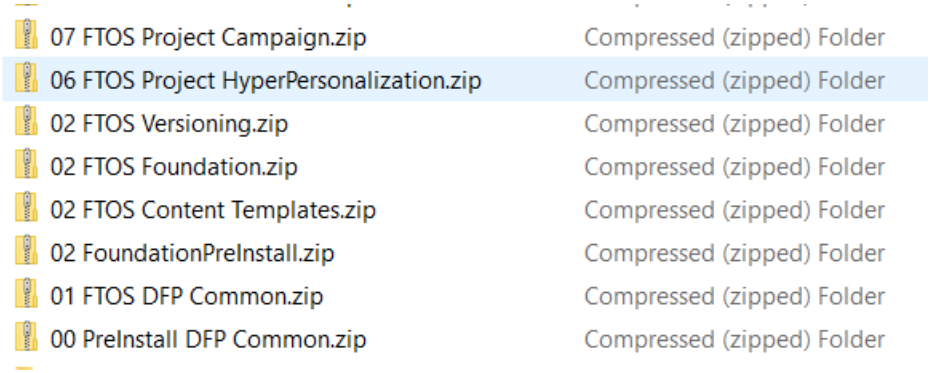
Folder	Description
custom	Contains the video custom components: • css, images, and javaScripts: `dcs-sdk.js` and `onfido.min.js`
custom-on-demand	Contains the liveness component resources.

HINT
For any other information about the steps performed and their result, check **<pckg_deployer_dir>\Logs**.

Manual Import Installation Steps

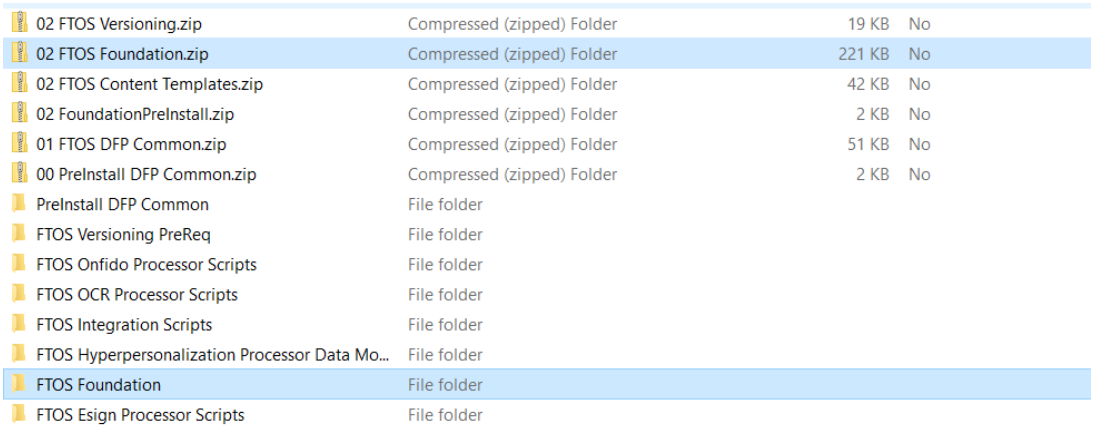
Follow the below steps if you choose to import the SysPack's individual deployment packages by hand.

1. Import the projects in SySDogitalSolutionPackages.Log into FintechOS Studio and navigate to **Configuration Management** > **Deployment**> **Digital Solution Packages**,
2. Click **Import Digital Solution Package** and select the zip packages in the order set by their names and import them one by one.



3. Run the SQL scripts found in the folders that have a part of the name of the packages.

   For example **02 FTOS Foundation. zip** and **FTOS Foundation**.



> **NOTE**
> If you need to update certain packages from the SysPacks, import the .zip files for those packages and run the sql scripts from the folder.

Generate SSL Certificate for FintechOS

This FintechOS guide explains how to generate an SSL certificate for a Fully Qualified Domain Name (FDQN) on a Windows Server machine using the win-acme tool.

1. Make sure that ports 80 and 443 on your environment allow Internet connectivity.
2. Download the tool from the official win-acme release page.
3. Run *wacs.exe* as administrator.
4. Type **M** to create a certificate with full options.

```
N: Create certificate (default settings)
M: Create certificate (full options)
R: Run renewals (0 currently due)
A: Manage renewals (1 total)
O: More options...
Q: Quit
Please choose from the menu: M
```

5. Type **2** to manually input the domain names included in the certificate.

```
Please specify how the list of domain names that will be
included in the
certificate should be determined. If you choose for one of
the "all bindings"
options, the list will automatically be updated for future
renewals to
reflect the bindings at that time.
1: Read bindings from IIS
2: Manual input
3: CSR created by another program
C: Abort
How shall we determine the domain(s) to include in the
certificate?: 2
```

6. Enter the FDQN you want to use. E.g.: vm-customer360-dev.westeurope.cloudapp.azure.com

```
Description:        A host name to get a certificate for.
This may be a
                    comma-separated list.
Host: vm-customer360-dev.westeurope.cloudapp.azure.com
Source generated using plugin Manual: vm-customer360-
dev.westeurope.cloudapp.azure.com
```

```
Friendly name '[Manual] vm-customer360-
dev.westeurope.cloudapp.azure.com'. <Enter> to accept or type
desired name: vm-customer360-
dev.westeurope.cloudapp.azure.com
```

7. Type **2** to serve the verification files from memory.

```
The ACME server will need to verify that you are the owner of
the domain
names that you are requesting the certificate for. This
happens both during
initial setup *and* for every future renewal. There are two
main methods of
doing so: answering specific http requests (http-01) or
create specific dns
records (dns-01). For wildcard domains the latter is the only
option. Various
additional plugins are available from https://github.com/win-
acme/win-acme/.
1: [http-01] Save verification files on (network) path
2: [http-01] Serve verification files from memory
3: [http-01] Upload verification files via FTP(S)
4: [http-01] Upload verification files via SSH-FTP
5: [http-01] Upload verification files via WebDav
6: [dns-01] Create verification records manually (auto-renew
not possible)
7: [dns-01] Create verification records with acme-dns
(https://github.com/joohoi/acme-dns)
8: [dns-01] Create verification records with your own script
9: [tls-alpn-01] Answer TLS verification request from win-
acme
C: Abort
How would you like prove ownership for the domain(s)?: 2
```

8. Type **2** to select the RSA key type.

```
After ownership of the domain(s) has been proven, we will
create a
Certificate Signing Request (CSR) to obtain the actual
certificate. The CSR
determines properties of the certificate like which (type of)
key to use. If
you are not sure what to pick here, RSA is the safe default.
1: Elliptic Curve key
2: RSA key
```

```
C: Abort
What kind of private key should be used for the certificate?:
2
```

9. Type **2** to store the certificate as PEM encoded files.

```
When we have the certificate, you can store in one or more
ways to make it
accessible to your applications. The Windows Certificate
Store is the default
location for IIS (unless you are managing a cluster of them).
1: IIS Central Certificate Store (.pfx per host)
2: PEM encoded files (Apache, nginx, etc.)
3: PFX archive
4: Windows Certificate Store
5: No (additional) store steps
How would you like to store the certificate?: 2
```

10. Type **2** to insert the path where you wish to store the certificates from the console.

    E.g.: C:\Users\john.doe\Documents)

```
Description:        .pem files are exported to this folder.
File path: .
Description:        Password to set for the private key .pem
file.
1: None
2: Type/paste in console
3: Search in vault
Choose from the menu: 2
```

11. Type **1** to disable password protection for the private key file.

```
Description:        Password to set for the private key .pem
file.
1: None
2: Type/paste in console
3: Search in vault
Choose from the menu: 2
```

12. Type **5** to decline any additional store steps.

```
1: IIS Central Certificate Store (.pfx per host)
2: PEM encoded files (Apache, nginx, etc.)
```

```
3: PFX archive
4: Windows Certificate Store
5: No (additional) store steps
Would you like to store it in another way too?: 5
Installation plugin IIS not available: This step cannot be
used in combination with the specified store(s)
```

Migrate User Accounts, Roles and Business Units to the FintechOS Identity Provider

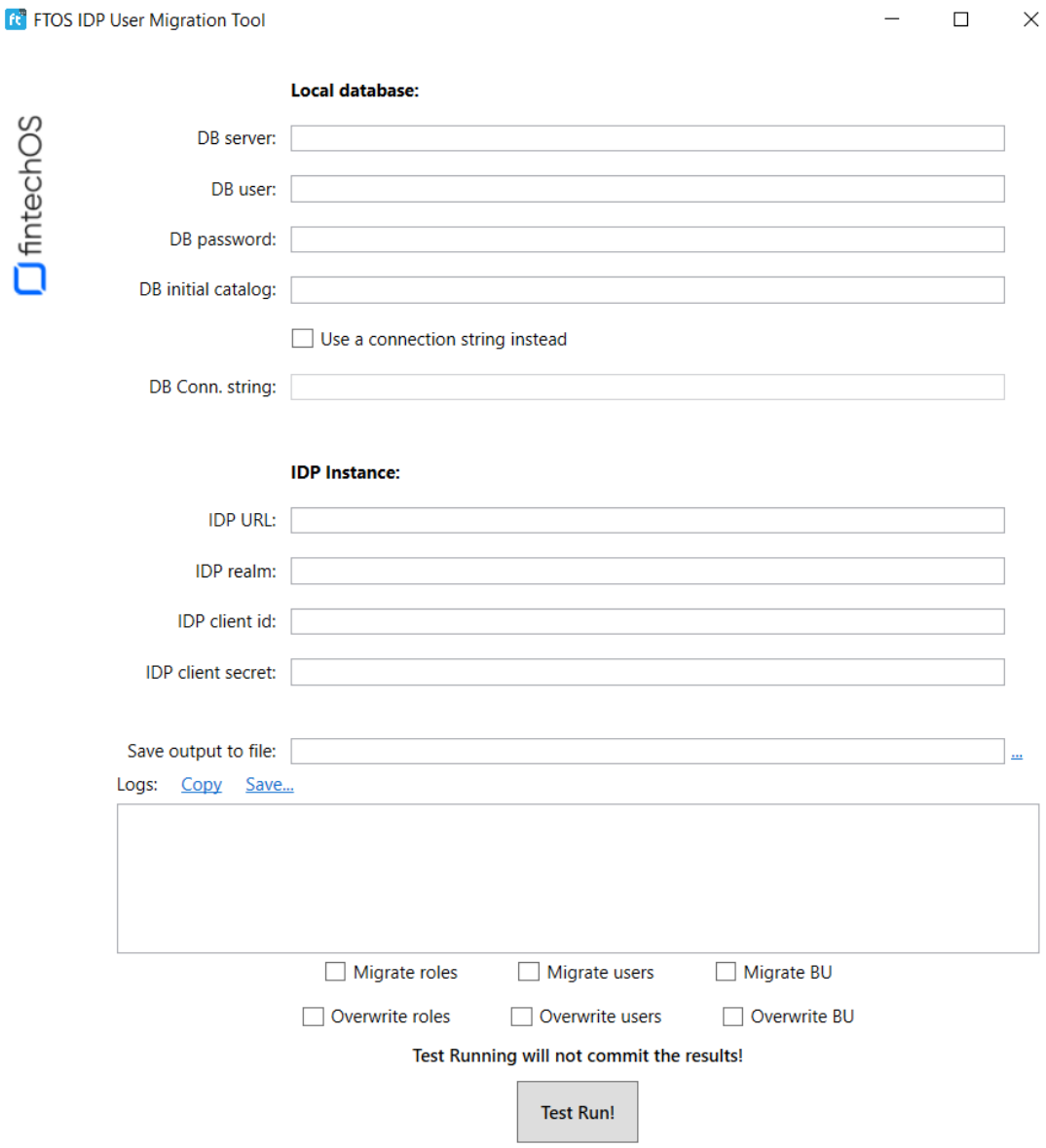
Starting with release 22.1, the FintechOS Platform uses the "FintechOS Identity Provider" on page 121 for identity and access management. In order to facilitate user accounts, roles and business units migration from legacy systems to the FintechOS Identity Provider, an automatic user migration tool has been provided.

IMPORTANT!
The FintechOS Identity Provider requires each user to have a unique email address. Users who don't have a unique email address will not be migrated.

To use the migration tool, follow the instructions below:

1. Make sure .NET 5.0 Desktop Runtime is installed on your system.
2. From the FintechOS Platform installation kit, copy the User Migration Tool archive and extract it on your system.

3. In the extracted folder, run the User Migration Tool executable (FTOS.Tools.UserMigrationToIdpUi.exe).



4. Fill in the following information:

Field	Description
DB server	Required if "Use a connection string instead" below is not checked. Database server address of your legacy FintechOS system.
DB user	Required if "Use a connection string instead" below is not checked. Database username used to connect to the legacy FintechOS database to download user accounts and user roles.
DB password	Required if "Use a connection string instead" below is not checked. Database user password for the above user account.
DB initial catalog	Required if "Use a connection string instead" below is not checked. Database name of your legacy FintechOS system.
Use a connection string instead	Uses the database connection string provided in the "DB Conn. string" below field to connect to the legacy FintechOS system database instead of the above settings.
DB Conn. string	Required if "Use a connection string instead" above is checked. Allows you to provide a database connection string which will be used to connect to the legacy FintechOS system database.
IDP URL	Required. URL of the FintechOS Identity Provider instance.
IDP realm	Required. FintechOS Identity Provider realm set up for the FintechOS Platform.
IDP client id	Required. Client ID set up in the FintechOS Identity Provider for the FintechOS Studio.
IDP client secret (v22.1.1 and later)	Required. The client secret associated with the client ID.

Field	Description
IDP admin API username (v22.1 only)	A user account set up in the FintechOS Identity Provider admin console with realm management role.
IDP admin API secret (v22.1 only)	Password for the above user account.
Save output to file (v22.1.1 and later)	Specify a .csv file where information regarding the users/roles/business units that have been migrated (together with the migration status) will be saved. If not specified, the output will be directed to the text area below.
Command line & results (v22.1 only)	You can use the automatically generated command to run users/roles/business units migration from the command line.
Save to log file (v22.1 only)	Allows you to define a text file where the migration results will be logged.

Field	Description
Migrate roles	Specify whether or not the tool should migrate user roles. **IMPORTANT!** In order the properly migrate the users, the roles must be migrated first. Either check both `Migrate roles` and `Migrate users` (the tool will first migrate the roles, and then the users) or run the tool first with the `Migrate roles` option checked, then run it again with the `Migrate users` option checked. If the tool tries to migrate the users before the roles are migrated, it will fail.
Migrate users	Specify whether or not the tool should migrate user accounts. **IMPORTANT!** User accounts without a valid email address will not be migrated. If another user account with an identical email address already exists in the FintechOS Identity Provider, the user will not be migrated (the legacy database authentication does not enforce unique email addresses for user accounts).
Migrate BU	Specify whether or not the tool should migrate business units.

Field	Description
Skip external users (v22.1 only)	Does not migrate external user accounts (users with a valid entry in the MembershipProviderUserId attribute).
Overwrite roles	Specify whether or not the tool should overwrite matching user roles that already exist in the FintechOS Identity Provider.
Overwrite users	Specify whether or not the tool should overwrite matching user accounts that already exist in the FintechOS Identity Provider.
Overwrite BU	Specify whether or not the tool should overwrite matching business units that already exist in the FintechOS Identity Provider.

5. (v22.1.1 and later) Click **Test Run!**. This will not commit the results. This step performs a dry migration (the tool simulates the actual migration), but the results are not saved.
6. (v22.1.1 and later) After the test run is finished, check the output .csv file and look for roles/users/business units that have the ERROR status. This indicates that the corresponding users/roles/business units will have issues during the real migration. Check the Reason column to determine why a user/role/business unit is in an error state and fix the issue.
7. (v22.1.1 and later) If, during the previous step, you had to fix any errors, you can close the tool and run it again to make sure there are no more issues.
8. Click **Run Migration!** This action commits the results and the users/roles/business units are migrated to the FintechOS Identity Provider.

Check the Migration Output

The test migration cannot identify all the possible errors that might occur during the actual migration. Even if, based on the test migration output, all the users/roles/business units should be migrated successfully, make sure to also check

the actual migration output file and verify the statuses of the users/roles/business units that have been migrated. If additional errors (that couldn't be identified in the test run) are present, try to fix them based on the information in the Reason column.

> **NOTE**
> The migration is idempotent and can be run multiple times.

Troubleshooting

During the test run, the following errors may occur:

- A role that needs to be migrated is found, but the `Overwrite roles` option is not checked. In this case, during the actual migration, the role will be skipped.
- There are users with duplicate emails or usernames in the legacy database. The legacy database and applications support multiple users with identical email addresses or usernames, but the FintechOS Identity Provider doesn't! To fix this, make sure that there are no duplicate email addresses or usernames in the database. If the errors are not fixed, during the actual migration, only the first such encountered user account will be migrated successfully.
- The user does not have an email address. This is a required field in the FintechOS Identity Provider, so users that do not have a valid email address will not be migrated. Make sure that all the users have a valid email address prior to running the actual migration.
- The user does not have a username. This is a required field in the FintechOS Identity Provider, so users that do not have a username will not be migrated. Make sure that all the users have unique usernames assigned prior to running the actual migration.
- A matching username has been found in the FintechOS Identity Provider, but the email address is different.

- A matching email address has been found in the FintechOS Identity Provider, but the username is different.
- Two matches are found in the FintechOS Identity Provider for the user to be migrated: one that matches the username and one that matches the email address.
- A matching user is found in the FintechOS Identity Provider, but the `Overwrite users` option is not checked. During the actual migration, this user will be skipped.
- There is an issue connecting to the database or the FintechOS Identity Provider. This will be notified in the text area of the tool. The users/roles/business units migration will not proceed.
- There is an issue connecting to the database or the FintechOS Identity Provider. This will be notified in the text area of the tool. The users/roles/business units migration will not proceed.
- None of the `Migrate roles`, `Migrate users`, or `Migrate BU` options are checked. The migration will stop, as there is no actual migration to be done.
- The test run has finished (the Run Migration button is visible), then one of the migration options changes. The migration tool will reset and the Test Run button will be displayed again. This is to make sure that the test run results are accurate.

Configuration Manager

The FintechOS Platform uses the HashiCorp Vault secrets management system to store system configurations in a secure and controlled environment. This protects sensitive data, such as system parameters, environment variables, services credentials, or API keys and simplifies user access and environment management.

> **NOTE**
> When changing system parameters from FintechOS Studio, the corresponding Vault secrets will be updated accordingly.

The Vault Agent can be installed either as an Azure web app for cloud deployments or as a Windows service for on-premise deployments.

Manage Vault Secrets

Directory Structure

The directory structure of a vault path is described below:

`<kvSecretsEngine>/<environmentName>/<applicationName>/<node>`

Directory	Designation	Description
<kvSecretsEngine>	Secrets Engine	FintechOS uses Vault's KV version 2 secrets engine to store system configurations as key-value pairs.
<environmentName>	Environment	You can define different sets of configurations specific to various environments such as development, testing, or production. This allows you to change the configurations for your system in one go by switching the environment directory.

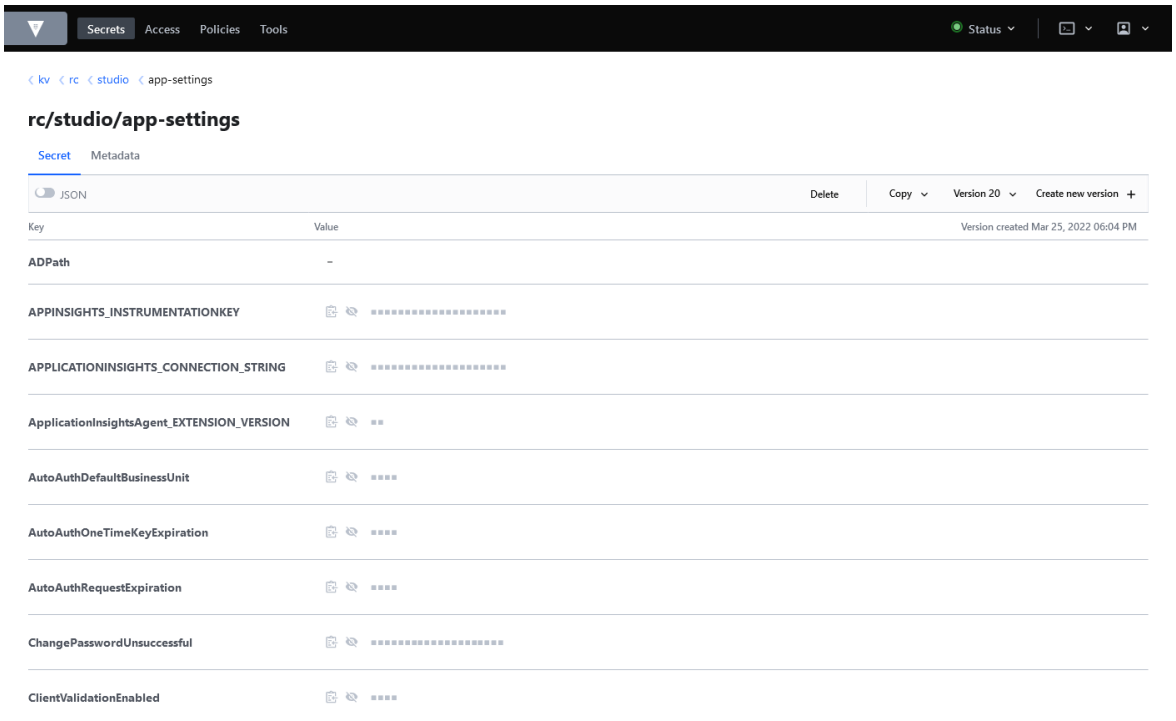
Directory	Designation	Description
<applicationName>	Application	Directory indicating the type of FintechOS system component such as an FintechOS Studio instance, a FintechOS Portal instance, or a FintechOS Identity Provider.
<node>	Node	Nodes allow you to group your secrets for easy classification and access management.

For example:

`kv/production/fintech-os-portal/app-settings`

Secrets

Within each node, you can define multiple secrets in the form of key-value pairs.



Vault Connection

To configure the connection between the Vault Agent and an FintechOS Studio or Portal instance, open its *web.config* file in a text editor and, in the `app-settings` node, edit the following keys:

```
<app-settings>
    <add key="vault__uri" value="https://myVaultWebApp" />
    <add key="vault__token" value="myVaultAuthToken" />
    <add key="vault__workspace__application_
environment" value="test" />
    <add key="vault__workspace__application_name" value="fintech-
os-portal" />
</app-settings>
```

Key	Value
vault__uri	Address of the Vault Agent web app or Windows service.
vault__token	Authentication token created by Vault for the operator used to access the system configurations.
vault__workspace__ application__name	Type of FintechOS ecosystem component. See "Application" on the previous page.
vault__workspace__ application__environment	Type of environment. See "Environment" on page 46.

Enable web.config Override

IMPORTANT!
The Vault secrets management system is the default method for storing system configurations. The web.config override is only intended for development and testing purposes, not for production use.

To control your FintechOS Studio or FintechOS Portal application settings from the web.config file instead of Vault, open the *web.config* file in a text editor and, in the `app-settings` node, add or enable the following key:

```
<app-settings>
    ...
    <add key= "feature-development-mode" value="1" />
    ...
</app-settings>
```

Key	Value
feature-development-mode	• 1 - enables web.config override. If an application setting is defined both as a Vault secret and web.config key, the web.config key will take precedence. • 0 - disables web.config override

IMPORTANT!
The `feature-development-mode` key should never be enabled in production, as it has multiple purposes targeted for developers (i.e. extra logging).

Import system parameters to Vault

Prerequisites

In the secrets engine, make sure the `EbsSqlServer` secret exists at the following location: `<kvSecretsEngine>/<environmentName>/<applicationName>/<connection-strings>`.

In order to import the system parameters from the database to Vault, use the SysParamToVault executable tool that can be found at the following path: [solution kit folder]\Tools\SysParamToVault\SysParamToVault.exe .

1. Open Windows PowerShell with the admin role.
2. Navigate to the SysParamToVault folder from the solution kit.
3. Run the SysParamToVault.exe file with the following parameters:

```
.\SysParamToVault.exe --e https://vault-proto.azurewebsites.net --t
s.sdq0RQaa7uLpTGUQG44e433y --n portal --r dev --d 1 --g
mykvSecretEngine
```

Parameter short name	Parameter long name	Description
g	vault-engine	The name of the secret engine.
e	vault-endpoint	Vaul endpoint.
t	vault-token	Vault token.
n	application-name	Application name.
r	application-environment	Application environment.
d	disable-ui-confirmations	Optional parameter. The default value is 0 (false). If set to true (1), the user is must confirm the database name configured in Vault in order for the importing process to continue.

The system parameters are now copied to the secret engine at the following location :`<kvSecretsEngine/<environmentName>/<applicationName>/<system-parameters>`.

OIDC Integration with IDP

The FintechOS platform comes with the capability to set up an OpenID Connect protocol integration between the Configuration Manager (Vault) and the FintechOS Identity Provider. This feature helps with accessing and configuring different components of the platform, in particular the Configuration Manager.

The OIDC integration comes with:

- policy based access
- FintechOS role based access to the following security roles:

- **Developer**: can **read** connection strings, can **update** app settings, b2c, FtosOpenApi.
- **Release Manager**: has the same privileges as Developer security role, plus **update** app-features and app-configurations.

Login to Configuration Manager with OIDC Enforced

1. In your browser, go to the Vault instance of your environment.
2. From the **Method** dropdown, pick **OIDC**.
3. In the **Role** field, type in Developer or ReleaseManager. Click **Sign in with OIDC Provider**. A popup opens.
4. Add your Username and Password. Click Sign In. The Configuration Manager opens and you will be able to see and access certain areas and secrets/configurations based on the role you have.

Sending Emails with SMTP

For sending emails with SMTP via the JobServer, some configurations need to be made in the Configuration Manager. You would also need to use the ftos.messaging.send ServerSDK function.

1. The following settings need to be done in the Configuration Manager:
2. Navigate to **jobserver-platformservices > PlatformServices**
3. Click the view button next to **Settings**. The settings are displayed.
4. Scroll-down and, if missing, create a new version of the file and add the following configuration:

```
{
```

```
    "execParams": "provider=smtp;providerSetting=SmtpEmail",
    "Assembly":
"FintechOS.Jobs.MessageDelivery.ScheduledServices",
    "Class":
"FintechOS.Jobs.MessageDelivery.ScheduledServices.SendMessag
esService",
    "Method": "",
    "Name": "FTOS.SendMessagesServiceSmtp",
    "SendNotification": null,
    "Type": "class"
}
```

5. Save and click the view button next to **Triggers**. The settings are displayed.

6. Scroll-down and, if missing, create a new version of the file and add the following configuration:

```
{
    "Async": false,
    "Calendar": null,
    "EndTime": "03.11.2080 11:02",
    "Expression": "0/30 * * * * ?",
    "Name": "FTOS.SendMessagesServiceSmtp",
    "PoolTime": null,
    "RepeatCount": "-1",
    "RescheduleAfterRun": false,
    "Services": [
        "FTOS.SendMessagesServiceSmtp"
      ],
     "StartTime": "02.11.2020 11:00"
    }
```

DevOps

DevOps is a set of processes that unifies development (Dev) and processes (Ops) to complement software development. Unlike traditional software development methodologies, DevOps enables companies to create and improve products and go to market at a faster pace.

This section covers the following topics:

Configure the File Upload Location

The file upload location provides storage for applications that require users to upload or download files (documents, images, etc.). The FintechOS Platform supports the Azure Blob and Amazon S3 Buckets storage providers for storing the uploaded or generated user files.

Use Azure Blob Storage for File Uploads

1. By default, a private container named *uploadebs* should be set up in your Azure storage account. If you don't have one or if you wish to use a different container, create a new container.

2. In the "Configuration Manager" on page 46, edit the following key:

Key Path	Key Name	Key Value

kv/<environment>/<application>/app-settings	FtosAzureBlobStorageOptions	<pre>{
		"ConnectionString": "DefaultEndpointsProtocol=https;AccountName=myAccountName;AccountKey=myAccountKey==;EndpointSuffix=core.windows.net",
		"RootContainerName": "uploadebs"
		}</pre>

where:

- *connectionString* is the connection string the platform is using to connect to an Azure Blob container;
- *rootContainer* is the root container name where the user files will be stored (by default *uploadebs*).

> **HINT**
>
> The connection string can be obtained in Azure from the Access Keys section of the storage account.

Use Amazon S3 Buckets for File Storage

In the "Configuration Manager" on page 46, edit the following key:

Key Path	Key Name	Key Value

kv/<environment>/<application>/app-settings	AmazonS3StorageOptions	<pre>{
		"AccessKeyId":"<access_key>",
		"SecretAccessKey":"<secret_key>",
		"RootBucketName":"<bucket_name>",
		"Region":"<aws_region>"
		}</pre>

where:

- *AccessKeyId* and *SecretAccessKey* are used by FTOS to sign the requests made to AWS. For more information, see Access Keys (Access Key ID and Secret Access Key).
- *RootBucketName* is the root bucket name where the user files will be stored.
- *Region* is the AWS region. For a complete list of available regions, see the Amazon documentation, section *Regions, Availability Zones, and Local Zones*. The region attribute must have one of the values from the column "Region". E.g.: `<aws region="eu-central-1"></aws>`

Configure the Storage for Custom Files Folders

Custom files are used to customize portal profiles, allowing you to change the appearance of theFintechOS Portal instances. They contain resources such as stylesheets, JavaScript files, or images that determine the look-and-feel of the FintechOS Portal.

These resources are grouped into two main folders: **custom** and **custom-on-demand** (which is used mainly for the liveness component resources). They are hosted using Azure Blob storage containers.

Configure the Storage for the Custom Folder

1. By default, a private container named *custom* should be set up in your Azure storage account. If you don't have one or if you wish to use a different container, create a new container.
2. In the "Configuration Manager" on page 46, edit the following key:

Key Path	Key Name	Key Value

kv/<environment>/<application>/app-settings	CustomFilesStorageSettings	<pre>{
		  "ConnectionString": "DefaultEndpointsProtocol=https;AccountName=myAccountName;AccountKey=myAccountKey==;EndpointSuffix=core.windows.net",
		  "RootContainerName": "custom"
		}</pre>

where:

- *connectionString* is the connection string the platform is using to connect to an Azure Blob container;
- *rootContainer* is the root container name where the custom files will be stored (by default *custom*).

> **HINT**
>
> The connection string can be obtained in Azure from the Access Keys section of the storage account.

Edit the key in both FintechOS Studio and Portal "Configuration Manager" on page 46 environments. If the key is not found the platform will fallback to a local file storage rooted in <applicationRoot>\custom folder.

Configure the Storage for the Custom-on-Demand Folder

1. By default, a private container named *custom-on-demand* should be set up in your Azure storage account. If you don't have one or if you wish to use a different container, create a new container.
2. In the "Configuration Manager" on page 46, edit the following key:

Key Path	Key Name	Key Value

kv/<environment>/<application>/app-settings	CustomOnDemandFilesStorageSettings	<pre>{
		"ConnectionString": "DefaultEndpointsProtocol=https;AccountName=myAccountName;AccountKey=myAccountKey==;EndpointSuffix=core.windows.net",
		"RootContainerName": "custom-on-demand"
		}</pre>

where:

- *connectionString* is the connection string the platform is using to connect to an Azure Blob container;
- *rootContainer* is the root container name where the custom files will be stored (by default *custom-on-demand*).

> **HINT**
>
> The connection string can be obtained in Azure from the Access Keys section of the storage account.

Edit the key in both FintechOS Studio and Portal "Configuration Manager" on page 46 environments. If the key is not found the platform will fallback to a local file storage rooted in <applicationRoot>\custom-on-demand folder.

Importing and Exporting Packages

In FintechOS Platform, users with elevated privileges (admin users) can export metadata from an environment and import it into another environment, using digital solution packages (recommended).

In FintechOS Platform, you have the following options for importing and exporting packages:

- In FintechOS Studio, from the Development menu > Digital Solution Packages. For more information, see the FintechOS Studio, section Digital Solution Packages.
- From the command line by using the **FtosSysPackageDeployer** tool.

The **FtosSysPackageDeployer** tool is available in the release subdirectory, *Tools**FtosSysPackageDeployer*. It allows you to do the following from the command prompt:

- list the customization sets found in a FTOS server.
- import / export in / from server a customization set from / in a local file.

> **NOTE** In order to use the tool, make sure to run the command prompt as admin.

For information on how to use the **FtosSysPackageDeployer** tool , see the built-in help by running the command prompt as admin and executing `FtosSysPackageDeployer.exe` without arguments. The tool is using the POST CUSTOMIZATION SET method of the FintechOS Platform API.

Use the FtosSysPackage Deployer for Digital Solution Packages

You can use the FtosSysPackage Deployer to import Digital Solution Packages that contain SQL objects, execute scripts before or after import, or map custom and custom-on-demand files to a portal profile:

1. Create a .config file with the following JSON :

```
{
  "digital-asset-custom-files-portal-mapping": [
    {
      "digital-asset-name": "DA",
      "portals": ["PP1", "PP2"],
      "map-to-all-portals-without-profile": true
    }
  ],
  "digital-solution-package": {
    "sql-scripts": {
      "before-import-folder": "FolderName",
      "after-import-folder": "AnotherFolderName"
    },
    "idp-themes": {
      "import-themes": true,
      "abort-import-on-exception": true
    }
  }
}
```

Descriptions:

`digital-asset-custom-files-portal-mapping : Array<Object>`

Object properties:

- `digital-asset-name: String` - The name of the digital asset that contains the custom files.
- `portals: Array<String>` - The portal profile to which the files should be deployed.

- `map-to-all-portals-without-profile: Boolean` - Set to true if the custom files should also be deployed on portal instances that do not have a profile. Otherwise, set to false.

`digital-solution-package : Object`

`sql-scripts : Object`

- `before-import-folder: String` - The name of the folder from the directory where the before scripts are located.
- `after-import-folder: String` - The name of the folder from the directory where the after scripts are located.

`idp-themes : Object`

- `import-themes: Boolean` - Controls whether or not the import should also process the IDP Themes found in the package (if any). Set to true if the IDP themes related to Portal Profiles should be pushed to IDP. Otherwise set to false.
- `abort-import-on-exception: Boolean` - Set to true if the import should roll back if an error occurs when the IDP themes are pushed to IDP. Otherwise set to false, and in case of an error the import continues and the error is logged.

2. Follow the guidelines below before adding the .config file:
    - the .config file must have the same name as the deploy package
    - the .config file must be in the same folder as the Digital Solution Package, but not inside the .zip file. This also applies to the SQL scripts folder.
3. Use *FtosSysPackageDeployer* to install the digital solution package as follows:
    - Locate the *FtosSysPackageDeployer* in the unzipped FintechOS installation kit at the following location: **<unzipped_install_archive>\Tools\FtosSysPkgDeployer**.
    - Create a new .bat file and edit and replace the parameters below with your own values.

- Add the config file with the custom file mapping.
- Right-click the .bat file and run it as administrator.

> **NOTE**
> If you have previously installed the "System Digital Solution Packages Installation" on page 28, use the `async install_SysPack.bat` file for editing and replacing the parameters.

.bat Parameters Explanation

For asynchronous import run the following command:

```
FtosSysPkgDeployer.exe-i –a -s <studio_url> -u <studio_
user_name> -p <studio_user_password> -z <db_Server> -v
<db_server_login_username> -k <db_server_login_password>
-d <db_name> -r <syspack_file_path>
```

Field	Description
<studio_url>	The web URL of the FintechOS Studio installation, for example http://localhost/ftos_studio.
<studio_user_name>	The username of the FintechOS Studio user under which this import is executed. The user has to exist in FintechOS Studio prior to this operation
<studio_user_ password>	The password for the FintechOS Studio user.
<db_server>	The name of the database server where the FintechOS installation database was created.
<DB_user>	The username of the SQL Server user with administration rights on the FintechOS installation database.
<db_server_login_ username>	The login username of the SQL Server user with administration rights on the FintechOS installation database.

Field	Description
<db_server_login_password>	The password for the above mentioned SQL user.
<db_name>	The name of the database where the FintechOS Platform is deployed.
<syspack_file_path>	The physical path to the unzipped Digital Solution Package previously downloaded.

HINT
For more information about the deployment tool, please run FtosSysPackageDeployer.exe without any arguments to see the built-in help.

Execute maintenance scripts

You can automatically execute maintenance scripts after package import by using the switch `execute-after-import-maintenance-script` with the `aim` alias. The switch takes true or false values.

When true or not specified, after importing a Digital Solution Package, the FtosSysPkgDeployer.exe executes maintenance scripts. When false, maintenance scripts are not executed. The switch's default value is true.

Examples:

```
FtosSysPkgDeployer.exe -i -s <value> -u <value> -p <value> --
execute-after-import-maintenance-script false
FtosSysPkgDeployer.exe -i -s <value> -u <value> -p <value> --aim
false
```

File-Type Upload Control

In the FintechOS Platform, you can control what types of files users can upload into the system.

This feature prevents users from uploading wrong file types, thus saving time from investigating errors and having to resubmit the files.

> **NOTE** The file-type upload control feature has been added to the previous existing validations: file extension validation, content size validation etc. For a content to be uploaded all validations must pass.

Enable the file-type upload control

By default, the file-type upload control is disabled. To enable it, add the following secret in Vault:

Key Path	Key Name	Key Value
kv/<environment>/<application>/app-settings	feature.upload.filetype-check	true

File-Type Upload Processing

If the file-type upload control is enabled, when uploading files using client scripts (using the ebs.upload function) or server automation scripts (using the ftos.files.upload function), the system verifies the uploaded content against the file extension. The system will try to match the uploaded content (the bytes) with the provided file extension based on a list of files signatures.

Files signatures are available for the following file types: pdf, docx, xlsx, pptx, odt, ods, jpg/jpeg, doc, xls, ppt, rtf, xml, png, gif, bmp, mp4, csv, mkv.

No match, the file is uploaded

If the matching process does not find any match between the file content and the available file signatures then the upload is allowed.

The user uploads an Autocad file.

Match, but the signature's extension is not what the file says it is

if the matching process finds a match between the file extension and the available file signature, the system further checks the file internal type (that's is, MIME type) which

serves as an integrity check. If there is a mismatch between the two, that means that the internal type of the file does not correspond to what the file extension says it is and the file upload is not allowed. An error will be returned.

The user tries to upload a PNG file (the content has a PNG signature) that has a ".jpg" extension

Executable files

By design, if the matching process identifies that the uploaded content has an EXE or DLL signature then the upload is not allowed. An error will be returned.

FintechOS Platform API a Standalone Web App

FintechOS Platform gives you the ability to set the FintechOS Platform API as a standalone web app, which means that the API it will work in exclusive API server mode. This is particularly useful when you want to get data from FintechOS Platform using API calls and use it within your own web apps. The following controllers are available via the API standalone app: WCF Services, API and Authorization.

> **NOTE** The Portal functionality is disabled, that means that the API web app will not be available to end users.

To configure the API as a standalone app, enable the API server by adding the following secret , as provided below:

Key Path	Key Name	Key Value
kv/<environment>/<application>/app-settings	feature-api-server	1

(Deprecated) Add configuration in web.config files:

```
<configuration>
        <app-settings>
            ...
            <add key="feature-api-server" value="1" />
            </app-settings>
 ...
 </configuration>
```

Activating Localization Debug Mode

[[[Undefined variable General.PlatformName]]] supports the localization of static and dynamic elements, as well as the localization of customized messages and metadata.

Before localizing in a new language, you need to prepare your environment to easily identify the resources to be localized. To do so, open Vault secrets change the value of the following:

From:

Key Path	Key Name	Key Value
kv/<environment>/<application>/app-settings	ebs:debugLocalization	0;

To:

Key Path	Key Name	Key Value
kv/<environment>/<application>/app-settings	ebs:debugLocalization	1;

The table below lists the localization keys:

Key	Description
ebs:uiLocalization	Toggles dynamic UI localization (HTML templates, after generate JavaScript). The immediate result visible in the user interface is the dynamic generation of HTML templates in the optimum format for localization.

Key	Description
ebs:dataLocalization	Toggles metadata (data) localization and automatically creates database support for each language.
ebs:debugLocalization	Toggles the debug mode.
ebs:localizationSchedulerTimeSpan	Interval in milliseconds when checking for external resource updates in the database.

Once the localization debug mode is activated, the following markers are displayed for both the user interface and the metadata localized values:

✓ - To indicate the localized values.

 - To indicate that values are localizable, but no localization has been provided.

 - To indicate that the fields allow localization of the data inside the field.

(Deprecated) Configuring keys in web.config files

In web.config files, change the value for the below keys:

From:

```
<add key="ebs:debugLocalization" value="0" />
```

To:

```
<add key="ebs:debugLocalization" value="1" />
```

Configure Notifications for Operations

In the FintechOS Platform it is possible to receive notifications for operations such as scheduled jobs.

For v24.3.2 and above, you must firstly configure the channel provider. In **Configuration Manager**, navigate to **Server Scripts > App settings** and configure the **EmailTo** and **EmailCC** fields. Add the email addresses. Restart the Service Scripts app service from Azure.

Make sure you tick the **Enabled** box, to enable the schedule job.

For versions 24.3.2 and below, the options such as **Send Notification On Error** or **Send Notification On Success** are available in Studio. For this to properly function, the email settings need to be configured in the **mail.config** file, as follows:

- Navigate to the **JobServer** folder. Please note that the **JobServer** folder is in the folder with the name of your project.
- Open the **mail.config** file with a text editor (such as Notepad++). It needs to contain the following:

```
<mailSettings>
  <server>test@fintechos.com</server>
  <port>587</port>
  <auth>true</auth>
  <user>test@fintechos.com</user>
  <password>insertPasswordHere</password>
  <from>test@fintechos.com</from>
  <to>test@fintechos.com</to>
  <cc>test@fintechos.com</cc>
  <bcc></bcc>
  <replyTo>test@fintechos.com</replyTo>
</mailSettings>
```

- Fill in the parameters, as follows:
    - `<server>FintechServer</server>` - the server from which FintechOS operates;

- `<port>587</port>` - the port corresponding to the server;
- `<auth>true</auth>` - set the value to **true** if the authentication requires username and password;
- `<user>test@fintechos.com</user>` - the username corresponding to the server;
- `<password>insertPasswordHere</password>` - the password of the provided user;
- `<from>test@fintechos.com</from>` - the email address which appears in the **From** field;
- `<to>test@fintechos.com</to>` - the email address to send the notification to;
- `<cc>test@fintechos.com</cc>` - optional; an additional address to send the notification to.

- Save and close the **mail.config** file.

Observability

Observability allows the FintechOS Platform to generate log events and push those logs to various destinations such as system consoles, files, log servers, or Application Performance Management (APM) services.

Send log messages to the system console

Configuration in the "Configuration Manager" on page 46:

Key Path	Key Name	Key Value
kv/<environment>/<application>/app-settings	console-logging	enabled=0; logLevel=Debug;

Parameter	Description
enabled	• 0 - disable console logging • 1 - enable console logging
logLevel	Minimum severity level for the logged messages. Available options are: Verbose, Debug, Info, Warning, Error, and Fatal.

(Deprecated) Configuration using web.config keys:

```
<app-settings>
    ...
    <add key="console-logging" value="enabled=0;
logLevel=Debug;" />
    ...
</app-settings>
```

Send log messages to local file storage

Configuration in the "Configuration Manager" on page 46:

Key Path	Key Name	Key Value
kv/<environment>/<application>/app-settings	file-logging	enabled=1; logLevel=Debug; flushInterval=1s; fileName=trace_roll_.log; retainedFileCount=99; rollSizeBytes=31457280;

Parameter	Description
enabled	• 0 - disable file logging • 1 - enable file logging
logLevel	Configures the minimum severity level for the logged messages. Available options are: Verbose, Debug, Info, Warning, Error, and Fatal.
flushInterval	If provided, a full disk flush will be performed periodically at the specified interval in seconds. E.g: 1s, 5s, 20s

Parameter	Description
fileName	Name of the log file. The file will be stored in the folder where the application starts.
retainedFileCount	Maximum number of log files that will be retained, including the current log file. For unlimited retention, set it to null. The default value is 31.
rollSizeBytes	If defined, a new log file is created when the log file size reaches the designated number of bytes. New log files will have an index counter appended in the format _NNN, with the initial log file given no index counter.

(Deprecated) Configuration using web.config keys:

```
<app-settings>
    ...
    <add key="file-logging" value="enabled=1;
logLevel=Debug; flushInterval=1s; fileName=trace_roll_.log;
retainedFileCount=99; rollSizeBytes=31457280;" />
    ...
</app-settings>
```

Send log messages to a Seq structured log server

Configuration in the "Configuration Manager" on page 46:

Key Path	Key Name	Key Value
kv/<environment>/<application>/app-settings	seq-logging	enabled=0; apiKey={seq instrumentation key}; logLevel=Debug; flushInterval=1s; serverUrl=http://localhost:5341; maxEventCount=100000;

Parameter	Description
enabled	• 0 - disable Seq logging • 1 - enable Seq logging
apiKey	Seq API key that authenticates the client to the Seq server.

Parameter	Description
logLevel	Minimum severity level for the logged messages. Available options are: Verbose, Debug, Info, Warning, Error, and Fatal.
flushInterval	Time to wait (in seconds) between checking for event batches.
serverUrl	Base URL of the Seq server that log events will be sent to.
maxEventCount	Maximum number of events that will be held in-memory while waiting to ship them to Seq. Beyond this limit, events will be dropped. Default: 100000.

(Deprecated) Configuration using Web.config keys:

```
<app-settings>
    ...
    <add key="seq-logging" value="enabled=0; apiKey={seq
instrumentation key}; logLevel=Debug; flushInterval=1s;
serverUrl=http://localhost:5341; maxEventCount=100000;" />
    ...
</app-settings>
```

Send log messages to an Azure Application Insights service

You can use the Azure Application Insights application performance management service (subscription required) to collect logging information. If you want to centralize your logs, you can configure multiple machines on the same cluster to send their logging information to the same Application Insights subscription.

> **NOTE**
> Logs saved to the local file storage are visible instantly. Messages sent to Azure Application Insights might be visible after a short delay ranging from seconds to minutes.

Configuration in the "Configuration Manager" on page 46:

Key Path	Key Name	Key Value
kv/<environment>/<application>/app-settings	azure-appinsights-logging	enabled=0; apiKey={app insights instrumentation key}; logLevel=Verbose; flushInterval=1m;

Parameter	Description
enabled	• 0 - disable Azure Application Insights logging • 1 - enable Azure Application Insights logging
apiKey	Instrumentation key value that it is used by default by all Microsoft.ApplicationInsights.TelemetryClient instances created in the logger.
logLevel	Minimum severity level for the logged messages. Available options are: Verbose, Debug, Info, Warning, Error, and Fatal. **IMPORTANT!** Specifying a log level greater than Warning, may flood Azure Application Insights with irrelevant information. It is not recommended to enable Azure Application Insights logging on development environments.
flushInterval	Maximum telemetry batching interval. Once the interval expires, Microsoft.ApplicationInsights.WindowsServer.TelemetryChannel serializes the accumulated telemetry items for transmission.

(Deprecated) Configuration using web.config keys:

```
<app-settings>
    ...
    <add key="azure-appinsights-logging" value="enabled=0;
apiKey={app insights instrumentation key}; logLevel=Verbose;
flushInterval=1m;" />
    ...
</app-settings>
```

Configure Azure Application Insights telemetry

Configuration in the "Configuration Manager" on page 46:

Key Path	Key Name	Key Value
kv/<environment>/<application>/app-settings	azure-appinsights-telemetry-logging	• 0 - disable telemetry • 1 - enable telemetry

NOTE
Deactivating telemetry does not affect logging.

(Deprecated) Configuration using web.config keys:

```
<app-settings>
    ...
    <add key="azure-appinsights-telemetry-logging" value="0"
/>
    ...
</app-settings>
```

Logging context

In addition to the actual log event, developers can track the context in which a log event occurred (machine name, portal profile, user context regional settings, thread ID, etc.), as well as define their own custom log context properties via server-side scripting. For more information, see the Innovation Studio User Guide.

Prevent Sequencer Infinite Loops

The FintechOS Studio sequencers allow you to define complex sequences of codes for uses such as invoice numbers. Among the customizations you can apply to your sequencers, is the Filter JS script which allows you to skip specific sequence numbers. Depending on your Filter JS code, this may lead to situations where the sequencer

enters an infinite loop. To prevent this, you can limit the sequence numbers your sequencers can skip until an infinite loop error is thrown. To do so, in the "Configuration Manager" on page 46, set up the following key:

Key Path	Key Name	Description
kv/<environment>/<Portal Instance>/app-settings	SequencerFilterInfiniteLoop	Numeric value indicating the maximum number of sequence numbers that can be skipped until an infinite loop error is thrown.

(Deprecated) Configuration using web.config keys

```
<app-settings>
    ...
    <add key="SequencerFilterInfiniteLoop" value="20"/>
    ...
</app-settings>
```

Configure the Async Engine

Async Engine is an asynchronous processing engine that allows you to address scenarios that require time-consuming logic or a high volume of requests through asynchronous processing. Some use cases include:

- **Large scale data processing** - Scenarios where large volumes of data need to be processed, and synchronous processing would result in unacceptably long response times. Examples include processing bulk uploads, generating complex reports, or handling data-intensive calculations. The Async Engine can queue incoming requests, process them asynchronously, and provide the results to the caller when processing is complete.
- **Time consuming operations** - Some operations may require extended processing time, making synchronous communication impractical. For instance, calling external APIs

with high latency or performing computationally intensive tasks can benefit from asynchronous processing. The Async Engine allows these operations to be offloaded, ensuring that the caller is not blocked waiting for a response.

- **Rate limiting and throttling** - APIs and services often impose rate limits or throttling constraints on their consumers. The Async Engine can help manage these constraints by queueing requests and processing them at a pace that adheres to the rate limits imposed by the APIs or services. This ensures that the system operates efficiently while respecting external constraints.
- **Resilient processing with redelivery and DLQ** - In scenarios where errors or failures can occur during processing, the Async Engine offers resiliency through redelivery mechanisms and the use of a Dead Letter Queue (DLQ). This allows the system to retry processing in case of failures and to store messages that could not be processed in the DLQ for further analysis, reprocessing, or deletion.
- **Scalable workload distribution** - As workloads increase, the Async Engine enables horizontal scaling by distributing the processing across multiple instances or workers. This ensures that the system can handle the increasing demand while maintaining performance and responsiveness.
- **Service decoupling** - The Async Engine promotes a decoupled architecture, allowing individual components to evolve independently, enhancing maintainability and reducing the impact of changes to the system. This allows better separation of concerns and easier management of the application.

Asynchronous Flow

In synchronous processing, each step of a flow is executed sequentially, and the flow must wait for each step to complete before proceeding to the next one. This means that the entire flow is blocked until all steps finish.

The asynchronous implementation introduces message queues that collect incoming step inputs. Each step can operate independently as it consumes inputs from its queue, allowing for parallel processing and efficient resource utilization. Additionally, the message queues provide a buffer that can handle bursts of incoming requests, ensuring smooth processing and improved resilience in case of spikes in load.

Async Engine Configuration

To set up the Async Engine on your environment, you need to configure its storage account, queues, flows, tables, redelivery logic, outbound HTTP connections, and inbound thread pool.

1 Configure the Azure Storage Account

To configure the storage account that hosts the Async Engine, in the "Configuration Manager" on page 46, at the `kv/<environment name>/async-engine` path, edit the following key:

Key	Value
`azure.storage.config`	`{ "accountName": "myAzureStorageAccount", "accountKey": "myAzureStorageAccountAuthKey", "suffix": "core.windows.net" }`

- `accountName` - Azure storage account name
- `accountKey` - Azure storage account key
- `suffix` - The termination of the storage account URL. Defaults to core.windows.net for the Azure Cloud, but may differ for countries with their own cloud infrastructure (e.g. China).

2 Configure the Queues

To describe the configuration for each queue involved in the async processing, in the "Configuration Manager" on page 46, at the `kv/<environment name>/async-engine` path, edit the following key:

Key	Value

queue.config	<pre>[
	  {
	  "name": "testasyncqueue1",
	  "maxMessages": 10,
	  "timeToLive": 1,
	  "delay": 1000
	  },
	  {
	  "name": "testasyncqueue2",
	  "maxMessages": 10,
	  "timeToLive": 1,
	  "greedy": true
	  },
	  {
	  "name": "testasyncqueue3",
	  "maxMessages": 4,
	  "timeToLive": 1,
	  "delay": 2000,
	  "greedy": false
	  }
	]</pre>

Parameter	Description	Data Type	Default Value
name	Name of the queue.	string	
maxMessages	Maximum number of messages that can be retrieved from the queue in one poll. The allowed range is 1 to 32 messages.	integer	1
delay	Number of milliseconds between polls.	long	500
greedy	When set to true, if a poll processes any messages, the next poll will run immediately, ignoring the delay setting.	boolean	false
timeToLive	Number of hours a message stays alive in the queue. Set to -1 for no expiration. Must be -1 or any positive number.	integer	1

3 Configure the Flows

To set up your flows, in the "Configuration Manager" on page 46, at the `kv/<environment name>/async-engine` path, edit the following key:

Key	Value

flow.config	[
	{
	"flowName": "test",
	"queueMappings": [
	{
	"sourceQueueName": "testasyncqueue1",
	"endpointName": "ServicePipesTest_Endpoint1",
	"targetQueueName": "testasyncqueue2",
	"endpointUrl": "https://google.com",
	"maxThreads": 10,
	"initial": true
	},
	{
	"sourceQueueName": "testasyncqueue2",
	"endpointName": "ServicePipesTest_Endpoint2",
	"targetQueueName": "testasyncqueue3",
	"maxThreads": 10
	},
	{
	"sourceQueueName": "testasyncqueue3",
	"endpointName": "ServicePipesTest_Endpoint3",
	"maxThreads": 10
	}
	]
	}
	]

- `flowName` - The name of the asynchronous business flow. You can define multiple flows, each with its own queue mapping.
- `queueMappings` - Settings for each step in the flow:
    - `initial` - Flag that indicates the first step in the flow.
    - `sourceQueueName` - The name of the queue that stores the step's inputs.
    - `endpointName` - Name of the endpoint called to process the inputs. If the endpoint is outside the FintechOS Platform, use the `endpointUrl` parameter instead.
    - `endpointUrl` - The URL of the endpoint called to process the inputs. If this is a FintechOS Platform endpoint, use the `endpointName` parameter instead. If you set up both an `endpointName` and an `endpointUrl`, the step will use only the `endpointName`.
    - `targetQueueName` - The name of the queue where the responses from the endpoint are sent (the `sourceQueueName` for the next step in the flow, as the step's outputs become inputs for the next step). The last step of the flow doesn't have a `targetQueueName`.
    - `maxThreads` - The number of processing threads that are used to process messages read from the step's queue. This depends on the `maxMessages` property from the `queue.config` key. For example, if the poll reads a batch of 32 messages, you should set this to at least 32 to ensure parallel processing. If processing takes too long, when the next batch of 32 messages is read, some of the processing threads may still be busy with messages from the previous batch. This means that new messages have to wait for the threads to finish their processing. Depending on your performance requirements, you may typically need to set up a higher number of threads than messages.

4 Configure the Storage Tables

The Async Engine uses Azure Table Storage to track the status of each asynchronous action, as well as the messages that encountered processing errors or exceeded the allowed number of redelivery attempts (the Dead Letter Queue - DLQ).

To configure the tables, in the "Configuration Manager" on page 46, at the `kv/<environment name>/async-engine` path, edit the following key:

Key	Value
`table.config`	`{ "statusTable": "AsyncStatusTable", "dlqTable": "AsyncDlqTable" }`

- **statusTable** - Name of the table where the processing status is stored.
- **dlqTable** - Name of the Dead Letter Queue table, where messages are stored in case of an error.

5 Configure the Redelivery Logic

If a message delivery fails, you can set up a number of redelivery attempts before sending that message to the Dead Letter Queue. To do so, in the "Configuration Manager" on page 46, at the `kv/<environment name>/async-engine` path, edit the following key:

Key	Value

redelivery.config	<pre>{
	  "default": {
	  "count": 2,
	  "delay": 2000
	  },
	  "additional": [
	  {
	  "httpStatus": 502,
	  "count": 2,
	  "delay": 1000
	  }
	  ]
	}</pre>

- `default` - Default redelivery settings.
  - `count` - Number of redelivery attempts.
  - `delay` - Delay between redelivery attempts in milliseconds.
- `additional` - Additional redelivery attempts settings for specific HTTP response status codes.
  - `httpStatus` - HTTP response status code.
  - `count` - Number of redelivery attempts.
  - `delay` - Delay between redelivery attempts in milliseconds.

If an Async Engine exception occurs during internal processing, the default redelivery configuration is used.

6 Configure the Outgoing HTTP Connection Pool

To optimize the usage of network resources and improve the overall performance and responsiveness, the Async Engine employs HTTP connection pooling. Instead of opening and closing a connection on each call, multiple endpoint connections are kept alive in a connection pool and reused in subsequent requests.

To configure the outgoing HTTP connection pool, in the "Configuration Manager" on page 46, at the `kv/<environment name>/async-engine` path, edit the following key:

Key	Value

`http.connection.config`	<pre>{
	"requestTimeout": 5000,
	"connectionTimeout": 20000,
	"socketTimeout": 230000,
	"totalConnections": 2000,
	"routeConnections": 200,
	"connectionTimeToLive": 30000
	}</pre>

Parameter	Description	Data Type	Default Value
routeConnections	The maximum number of connections per route (endpoint).	integer	20
totalConnections	The maximum number of connections in the pool.	integer	200
connectionTimeToLive	The number of milliseconds to keep a connection alive. By default, the connection is permanently kept alive.	long	
requestTimeout	The timeout in milliseconds to wait for a connection from the pool to be allocated to the request. A timeout value of zero is interpreted as an infinite timeout. A negative value is interpreted as undefined (system default). If a finite timeout is set and reached before a connection becomes available, an error is thrown and the message is sent to the DLQ. This waiting period is highly impacted by the volume of requests.	integer	-1

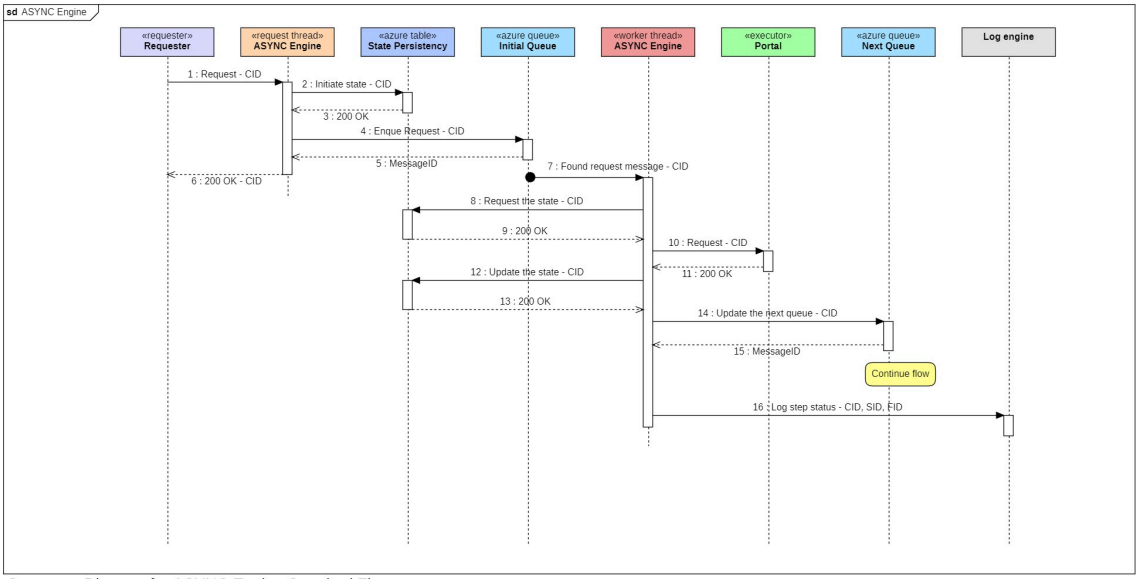
Parameter	Description	Data Type	Default Value
connectionTimeout	The timeout in milliseconds to attempt to establish communication with the endpoint. A timeout value of zero is interpreted as an infinite timeout. A negative value is interpreted as undefined (system default). If a finite timeout is set and reached before the communication with the endpoint is established, an error is thrown and the message is sent to the DLQ. This waiting period is highly impacted by the volume of requests.	integer	-1
socketTimeout	The timeout in milliseconds for waiting for data (a maximum period of inactivity between two consecutive data packets). A timeout value of zero is interpreted as an infinite timeout. A negative value is interpreted as undefined (system default). If a finite timeout is set and the endpoint is unresponsive for the set time, an error is thrown and the message is sent to the DLQ. This waiting period is highly impacted by the volume of requests.	integer	-1

IMPORTANT!

`routeConnections` and `totalConnections` should be set relative to the `maxThreads` property of the flow.config key. Each processing thread may require its own connection from the pool. How fast the responses are received also has an impact on the values that should be set. You should test and set up the number of connections accordingly.

7 Configure the Threads Pool for Incoming Requests

To optimize the usage of system resources and improve performance, a pool of worker threads (with a corresponding work queue) is kept for all the requests arriving to the Async Engine that need to be routed to the initial queues of the various flows.



Sequence Diagram for ASYNC Engine Standard Flow

To configure the request threads pool, in the "Configuration Manager" on page 46, at the `kv/<environment name>/async-engine` path, edit the following key:

Key	Value
`threadpool.config`	`{ "poolSize": 5, "maxPoolSize": "5", "maxQueueSize": "10" }`

Parameter	Description	Default Value
poolSize	Default number of request threads to be kept alive in the pool.	10
maxPoolSize	Maximum number of concurrent request threads.	20
maxQueueSize	Maximum number of pending requests stored in the requests queue. Use -1 for an unbounded queue.	1000

8 Configure Async Engine Authentication and Authorization

To set up the Async Engine's authentication and authorization settings, in the "Configuration Manager" on page 46, at the `kv/<environment name>/async-engine` path, edit the following keys:

Key	Value	Description

openid.config	<pre>{
	"realm": "fintechOSRealm",
	"auth-server-url": "https://myServer.azurewebsites.net/auth",
	"ssl-required": "external",
	"resource": "admin-async-dev",
	"principal-attribute": "preferred_username",
	"credentials": {
	"secret": "mySecret"
	},
	"use-resource-role-mappings": "false",
	"autodetect-bearer-only": "true"
	}</pre>
		• realm - The FintechOS realm configured in the "FintechOS Identity Provider" on page 121.
		• auth-server-url - The "FintechOS Identity Provider" on page 121 discovery endpoint.
		• ssl-required - The default value is *external* meaning that HTTPS is required by default for external requests. In production environments, this should be set to *all*.
		• resource - Name of the Async Engine resource as defined in the FintechOS realm

Key	Value	Description
		configured in the "FintechOS Identity Provider" on page 121. • principal-attribute - OpenID Connect ID Token attribute used to populate the UserPrincipal name. Possible values are: *sub*, *preferred_username*, *email*, *name*, *nickname*, *given_name*, and *family_name*. Default: *preferred_username*. • secret - Secret key set up in the "FintechOS Identity Provider" on page 121 for the Async Engine. • use-resource-role-mappings - When *true*, the

Key	Value	Description
		adapter retrieves the user's application level role mappings from the token. When *false*, it looks at the realm level role mappings. This should be set up in accordance with your OpenID configuration. Default: *false*. • autodetect-bearer-only - Set it to *true*. **Do not change.**

Key	Value	Description

rbac.config	<pre>{
	"apiMappings":[
	{
	"url":"/api",
	"roles":[
	"async-user"
	]
	}
	],
	"applicationMappings":[
	{
	"url":"/actuator",
	"roles":[
	"async-admin"
	]
	}
	]
	}</pre>
		Property used to configure role based access (RBAC) in the Async Engine.
		• apiMappings - RBAC for Async Engine API endpoints
		◦ url - Relative path to the API endpoints of the async flows. **Do not change.**
		◦ roles - Platform user roles that have access to the async flows endpoints.
		• applicationMappings - RBAC for application management/configuration URL.

Key	Value	Description
		◦ url - Relative path to the management URL. **Do not change.** ◦ roles - Platform user roles that have access to the management URLs. Here power-users or administrator roles should be configured.
openapi.url	https://myServer.azurewebsites.net/ftosapi/automation-processors/actions/	URL address of the OpenAPI component.
openapi.username	{username}	User name used by Async Engine when generating an access token to call the platform.
openapi.password	{password}	Password used by Async Engine when generating an access token to call the platform.

Configuration Example

```
{
  "azure.storage.config": {
    "accountName": "myAccount",
    "accountKey": "myAccountKey",
    "suffix": "core.windows.net"
  },
  "flow.config": [
    {
      "flowName": "test",
      "queueMappings": [
        {
          "sourceQueueName": "testasyncqueue1",
          "endpointName": "ServicePipesTest_Endpoint1",
          "targetQueueName": "testasyncqueue2",
          "endpointUrl": "https: //google.com",
          "maxThreads": 10,
          "initial": true
        },
        {
          "sourceQueueName": "testasyncqueue2",
          "endpointName": "ServicePipesTest_Endpoint2",
          "targetQueueName": "testasyncqueue3",
          "maxThreads": 10
        },
        {
          "sourceQueueName": "testasyncqueue3",
          "endpointName": "ServicePipesTest_Endpoint3",
          "maxThreads": 10
        }
      ]
    }
  ],
  "http.connection.config": {
    "requestTimeout": 5000,
    "connectionTimeout": 20000,
    "socketTimeout": 230000,
    "totalConnections": 2000,
    "routeConnections": 200,
    "connectionTimeToLive": 30000
  },
  "openapi.password": "myPassword",
  "openapi.url": "https://myEnvironment/ftosapi/automation-
processors/actions/",
```

```
  "openapi.username": "host",
  "openid.config": {
    "use-resource-role-mappings": false,
    "autodetect-bearer-only": true,
    "ssl-required": "external",
    "auth-server-url": "https: //myEnvironment/auth",
    "principal-attribute": "preferred_username",
    "resource": "admin-servicepipes-mybank",
    "credentials": {
      "secret": "mySecret"
    },
    "realm": "fintechOSrealm"
  },
  "queue.config": [
    {
      "name": "testasyncqueue1",
      "maxMessages": 10,
      "timeToLive": 1
    },
    {
      "name": "testasyncqueue2",
      "maxMessages": 10,
      "timeToLive": 1
    },
    {
      "name": "testasyncqueue3",
      "maxMessages": 4,
      "timeToLive": 1
    }
  ],
  "rbac.config": {
    "apiMappings": [
      {
        "url": "/api",
        "roles": [
          "async-engine"
        ]
      }
    ],
    "applicationMappings": [
      {
        "url": "/actuator",
        "roles": [
          "async-engine"
        ]
      }
    ]
```

```
  },
  "redelivery.config": {
    "default": {
      "count": 2,
      "delay": 2000
    },
    "additional": [
      {
        "httpStatus": 502,
        "count": 2,
        "delay": 1000
      }
    ]
  },
  "table.config": {
    "statusTable": "AsyncStatusTable",
    "dlqTable": "AsyncDlqTable"
  },
  "threadpool.config": {
    "poolSize": 5,
    "maxPoolSize": 5,
    "maxQueueSize": 10
  }
}
```

Integrations

This section explains how to integrate the FintechOS Platform with third party services, such as payment processors, electronic signature providers, or digital profile reviews:

FintechOS Service Pipes

Service Pipes are the integration layer of the FintechOS Platform. It uses Apache Camel as routing and mediation engine to integrate the FintechOS Platform with external systems. Apache Camel is an integration framework that allows easy implementation of routing and mediation logic using a variety of domain-specific languages (DSLs) .

The FintechOS Service Pipes are built as a docker image and deployed in an Azure AppService.

App Service Configuration

To configure the Service Pipes on your Azure environment, log in to the **Azure Portal** and navigate to your Service Pipes app service blade. In the Configuration section, set up the following settings:

Setting	Description
app.loglevel.application (optional)	Minimum severity level for the logged messages for the Service Pipes app. Available values are: DEBUG, ERROR, FATAL, INFO, OFF, TRACE, WARN. Default: INFO

Setting	Description
app.loglevel.root (optional)	Minimum severity level for the logged messages for all the packages. For the Service Pipes app specifically, the app.loglevel.application setting will take precedence over app.loglevel.root. Available values are: DEBUG, ERROR, FATAL, INFO, OFF, TRACE, WARN. Default: INFO
app.loglevel.security (optional)	Minimum severity level for security messages (authentication or authorization). Available values are: DEBUG, ERROR, FATAL, INFO, OFF, TRACE, WARN. Default: INFO
spring.profiles.active (optional)	Leave empty to use the "FintechOS Identity Provider" on page 121 for authentication. This is the default behavior. The legacy authentication using platform credentials is no longer supported.
app.context.path (optional)	The servlet path used for the Service Pipes app service, which is going to be appended to the app service URL. By default, the app will be available at the */services* servlet path, for example: https://app-myApp.azurewebsites.net/services/
app.vault.url	URL of the "Configuration Manager" on page 46 app service or Windows service.
app.vault.token	Access token for the "Configuration Manager" on page 46.
app.vault.secrets.engine (optional)	Secrets engine used by the "Configuration Manager" on page 46. Default: kv
app.vault.environment	The path where the services pipes folder is found in Vault (e.g. dev). The properties from this secrets path are required for the application startup and runtime.
app.vault.application	The folder where the application properties are found. It defaults to service-pipes.
app.vault.refresh.rate	The rate at which Vault properties are refreshed. It defaults to 3600000 (1h).

Configuration Manager Settings

The settings used to associate the Service Pipes with a FintechOS Portal instance are stored in the "Configuration Manager" on page 46. Currently, a single Service Pipes instance can be associated with a single FintechOS Portal instance. The corresponding

secrets are stored in the Configuration Manager at the `kv/{environment name}/service-pipes` path.

Key	Value	Description

openid.config	<pre>{
	"realm": "fintechOSRealm",
	"auth-server-url": "https://myServer.azurewebsites.net/auth",
	"ssl-required": "external",
	"resource": "admin-servicepipes-dev",
	"principal-attribute": "preferred_username",
	"credentials": {
	"secret": "TkATkOrKeTuubnZqo7ecEKGAgq6UXEvW"
	},
	"use-resource-role-mappings": "false",
	"autodetect-bearer-only": "true"
	}</pre>
		• realm - The FintechOS realm configured in the "FintechOS Identity Provider" on page 121.
		• auth-server-url - The "FintechOS Identity Provider" on page 121 discovery endpoint.
		• ssl-required - The default value is *external* meaning that HTTPS is required by default for external requests. In production environments, this should be set to *all*.
		• resource - Name

Key	Value	Description
		of the Service Pipes resource as defined in the FintechOS realm configured in the "FintechOS Identity Provider" on page 121. • principal-attribute - OpenID Connect ID Token attribute used to populate the UserPrincipal name. Possible values are: *sub*, *preferred_username*, *email*, *name*, *nickname*, *given_name*, and *family_name*. Default: *preferred_username*. • secret - Secret key set up in the

Key	Value	Description
		"FintechOS Identity Provider" on page 121 for the Service Pipes. • use-resource-role-mappings - When *true*, the adapter retrieves the user's application level role mappings from the token. When *false*, it looks at the realm level role mappings. This should be set up in accordance with your OpenID configuration. Default: *false*. • autodetect-bearer-only - Set it to *true*. Do not change.

Key	Value	Description

rbac.config	<pre>{
	"apiMappings":[
	{
	"url":"/inbound",
	"roles":[
	"service-pipes-user"
	]
	}
	],
	"applicationMappings":[
	{
	"url":"/actuator",
	"roles":[
	"service-pipes-admin"
	]
	}
	]
	}</pre>
		Property used for configuring role based access (RBAC) in Service Pipes.
		• apiMappings - RBAC for API endpoints
		◦ url - Relative path to the API endpoints. **Do not change.**
		◦ roles - Platform user roles that have access to the API endpoints.
		• applicationMappings - RBAC for application management/configuration URL.
		◦ url - Relative

Key	Value	Description
		path to the management URL. **Do not change.** ◦ roles - Platform user roles that have access to the management URLs. Here power-users or administrator roles should be configured.

Key	Value	Description

threadpool.config	<pre>{
	"poolSize": 5,
	"maxPoolSize": "5",
	"maxQueueSize": "10"
	}</pre>
		To optimize the usage of system resources and improve performance, a pool of worker threads (with a corresponding work queue) is kept for all the requests that need to be routed by the Service Pipes.
		• poolSize - Default number of worker threads to be kept alive in the pool. Default: 10.
		• maxPoolSize - Maximum number of worker threads that can be active at one time. Default: 20.
		• maxQueueSize - Maximum number of pending requests waiting in the requests queue.

Key	Value	Description
		Use -1 for an unbounded queue. Default: 1000.
portal.url	https://myServer.azurewebsites.net/portal/api/openApiV2/CallAction	URL of the FintechOS Portal instance associated with the Service Pipes.
openapi.enabled	true	Flag used to determine if requests are forwarded to OpenAPI instead of Portal.
openapi.url	https://myServer.azurewebsites.net/ftosapi/automation-processors/actions/	URL address of the OpenAPI component.
portal.username	{username}	Used by Service Pipes when generating an access token to call the platform.
portal.password	{password}	Used by Service Pipes when generating an access token to call the platform.

Key	Value	Description
openapi.config	Provides Service Pipes with the paths to the endpoints, entities, and digital journeys APIs. `{ "serverUrl": "https://<openapi-app-name>.azurewebsites.net/ftosapi", "requestType": { "endpoint": "/automation-processors/actions/", "entity": "/evolutive-data-model/entities/", "digitalJourney": "/digital-journeys/" } }`	• serverUrl - Root path to the platform's APIs. • endpoint - Relative path to the platform's custom endpoints APIs. • entity - Relative path to the entities APIs. • digitalJourney - Relative path to the digital journey APIs.

Key	Value	Description

throttle.config	<pre>{
	  "enabled": true,
	  "cacheRequestsMillis": ,
	  "pool": {
	  "maxConnections": ,
	  "connectionsPerRoute": ,
	
	"timeoutConnectionRequest": ,
	  "socketTimeout":
	  },
	  "refreshCacheCron": "",
	  "clients": [
	  {
	  "name": "",
	  "allowedRequests": ,
	  "allowedTimeFrameMs": ,
	  "timeToLive":
	  }
	  ]
	}</pre>
		This property is needed only if Service Pipes application is started with the "throttiling" profile.

User Roles

When using the "FintechOS Identity Provider" on page 121 for identity and access management, make sure you assign the following user roles accordingly:

- service-pipes-admin - Role needed for users responsible for monitoring the Service Pipes. This will provide them access to the Service Pipes monitoring tool available at the `<Service Pipes URL>/actuator/hawtio` path. E.g.: https://myServer.azurewebsites.net/services/actuator/hawtio
- service-pipes-user - Role needed for users who need to authenticate to the Service Pipes, such as user accounts that will run digital journeys that make calls to the Service Pipes server.

For advanced configurations, roles can be further customized using the "Configuration Manager" on page 46.

Connect to Azure Notification Hubs

The server SDK sendMobileNotifications function allows you to send notifications to subscribed user devices via the Azure Notifications Hub push engine. To connect FintechOS Platform with the Azure Notifications Hub, follow the steps below:

1. Configure your notifications hub on the Microsoft Azure cloud computing service. For details, see the Azure Notification Hubs documentation.

   > **NOTE**
   >
   > You have to create one notification hub per mobile app, per environment.

2. In Vault, add secrets for the hub name and endpoint settings of each client application, based on the model below:

Key Path	Key Name
kv/<environment>/<application>/app-settings	azure-mobile-notifications-myApp1-hubname
kv/<environment>/<application>/app-settings	azure-mobile-notifications-myApp1-endpoint
kv/<environment>/<application>/app-settings	azure-mobile-notifications-myApp2-hubname
kv/<environment>/<application>/app-settings	azure-mobile-notifications-myApp2-endpoint

Key Name	Key Value
azure-mobile-notifications-myApp1-hubname	xxxhubname1
azure-mobile-notifications-myApp1-endpoint	Endpoint=sb://xxxnamespace1.servicebus.windows.net/;SharedAccessKeyName=DefaultFullSharedAccessSignature1;SharedAccessKey=xxxxxxxxx1

Key Name	Key Value
azure-mobile-notifications-myApp2-hubname	xxxhubname2
azure-mobile-notifications-myApp2-endpoint	Endpoint=sb://xxxnamespace2.servicebus.windows.net/;SharedAccessKeyName=DefaultFullSharedAccessSignature2;SharedAccessKey=xxxxxxxxx2

In the example above:

- We set up two client applications that will receive notifications: **myApp1** and **myApp2**.
- The notifications are sent using the **xxxhubname1** and **xxxhubname2** Azure notifications hubs respectively.

- The endpoints for the two hubs are **sb://xxxnamespace1.servicebus.windows.net/** and **sb://xxxnamespace2.servicebus.windows.net/**.
- The shared access key names are **DefaultFullSharedAccessSignature1** and **DefaultFullSharedAccessSignature2**.
- The shared access keys are **xxxxxxxxx1** and **xxxxxxxxx2**.

(Deprecated) Add configuration in web.config:

```
<app-settings>
    ...
    <add key="azure-mobile-notifications-myApp1-
hubname" value="xxxhubname1">
    <add key="azure-mobile-notifications-myApp1-
endpoint" value=
"Endpoint=sb://xxxnamespace1.servicebus.windows.net/;SharedA
ccessKeyName=DefaultFullSharedAccessSignature1;SharedAccessK
ey=xxxxxxxxx1">
    ...
    <add key="azure-mobile-notifications-myApp2-
hubname" value="xxxhubname2">
    <add key="azure-mobile-notifications-myApp2-
endpoint" value=
"Endpoint=sb://xxxnamespace2.servicebus.windows.net/;SharedA
ccessKeyName=DefaultFullSharedAccessSignature2;SharedAccessK
ey=xxxxxxxxx2">
</app-settings>
```

Push Notifications Log

Sent notifications are saved in the FTOS_DPA_MessageQueue table. The table contains an entry for each notification sent to each user. This message queue can be viewed at the http://localhost:57123/Main#/entity/FTOS_DPA_MessageQueue/list link:

Attribute	Description
ToAddress	Mobile app name.
UserId	ID of the user that received the notification.

Attribute	Description
Subject	Message queue subject set in the sendMobileNotifications function call that initiated the notification push.
Body	Message received by the recipient.
ChannelProvider	Provider with the same name as the Mobile App Name.
CommunicationChannel	Hardcoded to **AzureNotificationHub**.
ChannelProviderParams	Recipient filter used when sending the notification (example: "role: developer").
MessageStatus	Hardcoded to **Sent**. If notifications were not successfully received, check the logging in the Azure Portal.

FAQs

What happens if the messages are added to the FintechOS message queue, but are not sent by the Azure notification hub?

The messages remain in the Sent status. The main purpose of the message queue logging is to track notification attempts. For troubleshooting, the main place to check statuses is the Azure portal.

Is there a limit to the number of notifications sent at once?

No. The Azure Notification Hubs support a maximum of 20 tags on a notification command, but FintechOS Platform automatically splits the notification into batches if this number is exceeded.

If my role contains 5 users, but only 3 are registered to the notification hub, how many push notifications are going to be sent?

In the FintechOS Platform message queue, there are going to be 5 entries (one for each user). On the Azure notification hub, after sending the request from FintechOS Platform, the outcome will be: 3 Success, 0 Failed.

Can I have only one notification hub for 2 client apps ?

It is recommended to have 1 hub per app. In the Azure portal > Notification Hub setup, you can set only one key, which is normally associated to only one app.

There are exceptions such as Android-FCM where the API key corresponds to the Firebase project. In this case, you can:

- Add multiple apps on the same Firebase project, resulting in a single API key.
- Create a project for each App, resulting in separate API keys/hubs.

CertSign Integration for electronic signature

Certsign is a digital certification for digital signatures. It will provide the user with the capability to use the Esign processor in the Studio and Portal. This makes possible to sign contracts and other documents by a customer. The existing integration provides two types of signature:

- Remote signature (with authorization code sent through sms)
- Automatic signature (with an existing certificate)
- Automatic signature with qualified electronic sign.

After the installation of the ESign provider package, you should add the following configuration in Vault, or in JobServer serviceSettings.config:

In Vault

Key Path	Key Name
kv/<environment>/<FintechOS Portal instance>/app-settings	FTOSServicesESignProvider2Endpoint
kv/<environment>/<FintechOS Portal instance>/app-settings	FTOSServicesESignProvider2AppId
kv/<environment>/<FintechOS Portal instance>/app-settings	ESignProvider2CertName

- For the FTOSServicesESignProvider2Endpoint key, add as value the URL to the environment.
- For the FTOSServicesESignProvider2AppId key, add as value the subscription key.
- For the ESignProvider2CertName key, add as value the mapping for the certificate provided by FTOS.

In JobServer serviceSettings.config:

```
<add
key
=
"FTOSServicesESignProvider2Endpoint"
 value="https://aztestapi01.azure-api.net/certSign"/> <!-- This is
the test env url -->
<add key="FTOSServicesESignProvider2AppId" value=""/><!-- the
subscription key -->
<add key="ESignProvider2CertName" value="certSignTest"/> <!-- the
mapping for the certificate provided by FTOS-->
```

If you have to configure the automatic signature also, add the following secrets in Vault:

Key Path	Key Name
kv/<environment>/<FintechOS Portal instance>/app-settings	ESign2AutomaticNumber_ {ProfileName}
kv/<environment>/<FintechOS Portal instance>/app-settings	ESign2AutomaticName_{ProfileName}

- For ESign2AutomaticNumber_{ProfileName}, add as value the serial number provided for the specific profile
- For ESign2AutomaticName_{ProfileName}, add as value the issuer information for the profile

(Deprecated) Add keys in web.config

```
<add key="ESign2AutomaticNumber_
{ProfileName}" value=""/> <!--this will contain the
serial number provided for the specific profile-->
<add key="ESign2AutomaticName_
{ProfileName}" value="cn=certSIGN CA Class 2
G2,ou=certSIGN CA Class 2 G2,o=certSIGN,c=RO"/> <!--
this will contain the issuer information for the
profile-->
```

IMPORTANT!
The token {ProfileName} must be replaced with a profile name that will be used when requesting the signature process.

Configure the CData Sync Service

The CData Sync service is required for the FintechOS Platform Data Pipes data replication feature. CData Sync must be installed on the same machine as the FintechOS Platform. The service is shared between FintechOS Platform instances. If you have multiple platform instances running on the same machine, install the CData Sync service only once.

System Requirements

- Windows Vista/Windows Server 2008 or higher.
- .NET Framework 4.5 or higher.
- 500 MB RAM required. 1+ GB recommended.
- Adequate free disk space for job logging.

Installation

1. Copy the CData Sync installation kit provided by FintechOS Platform to your local machine.
2. Open Windows PowerShell as administrator and navigate to the installation kit folder.
3. Run the following command in Windows PowerShell:

```
.\FtosCDataSyncInstaller.ps1 -p_MainCommand Install -p_
InstallDir <installation path>
```

4. This will start the installer. At the command line prompt, type **GO!** and press **Enter**.

The CData Sync server will be installed in the specified directory. The default credentials are:

- username: admin
- password: admin

For more details about managing the CData Sync server, see the CData official documentation.

Upgrade

To upgrade the CData Sync server, follow the same "Installation" above instructions, but replace the Windows PowerShell command with:

```
.\FtosCDataSyncInstaller.ps1 -p_MainCommand Upgrade -p_InstallDir
<installation path>
```

Uninstall

To uninstall the CData Sync server, follow the same "Installation" on the previous page instructions, but replace the Windows PowerShell command with:

```
.\FtosCDataSyncInstaller.ps1 -p_MainCommand Uninstall -p_InstallDir
<installation path>
```

Configure the Payment Processor Service Provider

If you wish to enable online payments in your digital journeys, you need to partner with a payment processor and configure the link to their service in the FintechOS Studio *web.config* file.

1 Define a new type of section in the web.config file for the payment processor

Opent the FintechOS Studio *web.config* file in a text editor and add a new entry inside the **<configSections>** node:

```
<section
name
="ftosPaymentProcessor"

type="EBS.Core.Utils.Services.Config.PaymentProcessorConfigSection,
EBS.Core.Utils"/>
```

2 Add the connection settings for your payment processor

Opent the FintechOS Studio *web.config* file in a text editor and add a new entry inside the **<configuration>** node (after <configSections>):

```
<ftosPaymentProcessor type="Netopia">
    <definition>
        <settings alias="conf1">
            <setting
name="environment" value="http://sandboxsecure.mobilpay.ro"/>
            <setting name="publicCertificate" value="C:\\PATH_TO_
CERT\\sandbox.cer"/>
            <setting name="privateKey" value="C:\\PATH_TO_
CERT\\sandbox.key"/>
            <setting name="signature" value="XXX"/>
            <setting
name="redirectUrl" value="http://localhost/test_redirect.html"/>
            <setting
name="confirmUrl" value="http://localhost/test_confirm.html"/>
        </settings>
    </definition>
</ftosPaymentProcessor>
```

NOTE

- Currently, only the *Netopia* payment processor type is supported, which will link to a mobilPay service provided by Netopia Payments. Additional payment processors may be available in the future.
- The *alias* will identify the payment processor service when initiating payments using the getPaymentToken function.

Configure the FTOSApiSMS Service

The FTOSApiSMS service allows the FintechOS Platform to send SMS messages. Follow the instructions below to enable the service.

To configure the service, add the following key in "Configuration Manager" on page 46

Key Path	Key Name	Key Value
kv/<environment>/<application>/app-settings	ftosApiSmsProvider	"from":"SantaClaus""serviceUrl":"https://test.azure-api.net/dcs/message3/sendSms","subscriptionKey":"ccc000000f6841ceb36e7db6b981ddfa"

Parameter	Description
from	A text representing the sender of the message.
serviceUrl	The URL to FintechOS SMS gateway.
subscriptionKey	Subscription key for the service.

Configure the OneyTrust Digital Review service

The OneyTrust Digital Review service analyzes the information in a user's profile (email, telephone number, address, etc.) and calculates a reliability score for that information. This allows companies to detect potentially problematic profiles and act accordingly (for instance, by deciding to direct them to a manual review process instead of accepting them automatically).

To set up a connection to the OneyTrust service, add the following keys in Vault:

Key Path	Key Name	Key Value
kv/<environment>/<application>/app-settings	FTOSServicesOneyTrustEndpoint	exposed endpoint url
kv/<environment>/<application>/app-settings	FTOSServicesOneyTrustAppId	the subscription key

Once the connection to the OneyTrust service is set up, you can use the createReview and getReview Server SDK functions to review user profiles.

(Deprecated) Configuration using **web.config** files:

```
    <add key="FTOSServicesOneyTrustEndpoint" value="exposed
endpoint url"/>
    <add key="FTOSServicesOneyTrustAppId" value="the
subscription key"/>
```

Security

FintechOS Platform was built on 4 pillars of security: data encryption, authentication, authorization and logging. With a keen focus on security critical aspects, such as: access rights, segregation of duties, data ownership, it also provides you with comprehensive audit trail of what happened at any given time and who performed the action.

For all cloud deployment types, you own your data and identities. You are responsible for protecting the security of your data and identities, on-premises resources, and the cloud components you control (which varies by service type). We recommend you to implement security best practices provided by your cloud provider.

This section covers the following topics:

Data Encryption and Security

One of the keys to data protection is accounting for the possible states in which your data may occur, and what controls are available for that state:

- **Data in transit**. When data is being transferred between components, locations or programs, such as over the network, across a service bus, or during an input/output process, it is thought of as being in-transit.
- **Data at rest**. This includes all information storage objects, containers, and types that exist statically on physical media, be it magnetic or optical disk.

Data in transit is encrypted using the industry standard TLS min. 1.2 encryption algorithm.

Data at rest is encrypted using the AES-256 encryption algorithm.

To establish identity and trust between [[[Undefined variable General.PlatformName]]] web-based platform and the web browser, the connection is secured via SSL certificates.

The SSL-secured communication between FintechOS Platform and the client is done using the symmetric encryption keys that are established during the authentication process.

The data model and all scripts defined within FintechOS Platform can be exposed through REST APIs to enable integration with 3rd party systems / solutions. FintechOS Platform APIs are secured through OAuth 2.0 and follow the OWASP security standards.

You can encrypt the data at rest using security best practices provided by the infrastructure provider of choice where you install and deploy FintechOS Platform (Microsoft Azure, AWS, IBM Cloud, other).

XSS Prevention

To prevent Cross-Site Scripting (XSS) and keep users safe, all user input data is sanitized by default, except for the following attributes: JavaScript, HTML and XML.

The XSS prevention secures your web apps by escaping user input of type JavaScript, HTML and XM. It censors the data received by the web pages in a way which disallows the following characters: ”<”, “</”, “>”, “<” and “>” (e.g., <text, </text, <text or >text) from being rendered.

> **IMPORTANT!** When importing deployment packages or adding new metadata in Platform versions which have XSS prevention enabled, you have to eliminate the following tags from metadata and packages: ”<”, “</”, “>”, “<” and “>”; otherwise, you will get an error message and you will not be able to import them.

Authentication

Authentication is the process of verifying the identity of a user based on a set of credentials.

FintechOS Platform provides the following authentication mechanisms:

FintechOS Identity Provider

The FintechOS Identity Provider is an OpenID compliant identity and access management solution based on the Keycloak authentication server. All FintechOS Platform components, such as FintechOS Studio, FintechOS Portal, or FintechOS API, are represented in the FintechOS Identity Provider as different clients of the same FintechOS realm.

The FintechOS Identity Provider supports two primary scenarios, depending on your infrastructure requirements:

- Acting as an identity provider
- Acting as an identity broker

Use case	Identity Provider	Identity Broker
Create/Update Users	User accounts, including service accounts, must be created, updated, activated, or deactivated in FintechOS Studio. Changes are automatically synchronized with FintechOS IDP. User details such as phone number, user type, and name are also managed in Studio and synced accordingly.	Users are created and managed in an external identity provider, such as Microsoft 365. When a user logs into Studio, FintechOS IDP receives a token from the external IDP. If the user doesn't exist in FintechOS IDP, it is created automatically, including mapped roles and business units. Studio then creates or updates the user based on the token.

Use case	Identity Provider	Identity Broker
Security Roles	All security roles, including business roles, are defined in Studio and automatically propagated to the FintechOS IDP. Role assignments for users are handled in Studio and synced with FintechOS IDP. Roles for service accounts are assigned directly in FintechOS IDP.	Roles are defined and managed in the external IDP (e.g., Microsoft 365). These roles are included in the token claims sent to FintechOS IDP, then mapped and passed to Studio, where they are assigned or updated accordingly. Roles for service accounts are not applicable in this scenario, as service accounts and their roles are managed externally or handled differently depending on the IDP setup.
Business Units	Creating, updating and assigning business units to user accounts is done in Studio.	Business unit mapping is based on token claims. If a business unit in the token doesn't exist in Studio, it is created automatically under the root using the name provided in FintechOS IDP mappings.
Login Flows and Password Policies	Login flows and password change rules are managed directly within the FintechOS IDP.	Authentication, login flows, and password management are handled entirely by the external identity provider (e.g., Microsoft 365), not by FintechOS IDP or Studio.

Identity Brokering Settings

The FintechOS Identity Provider supports identity brokering, allowing users to log in to FintechOS applications and services using any external identity provider that supports the OpenID Connect standard. You can find examples for common external identity providers configurations below:

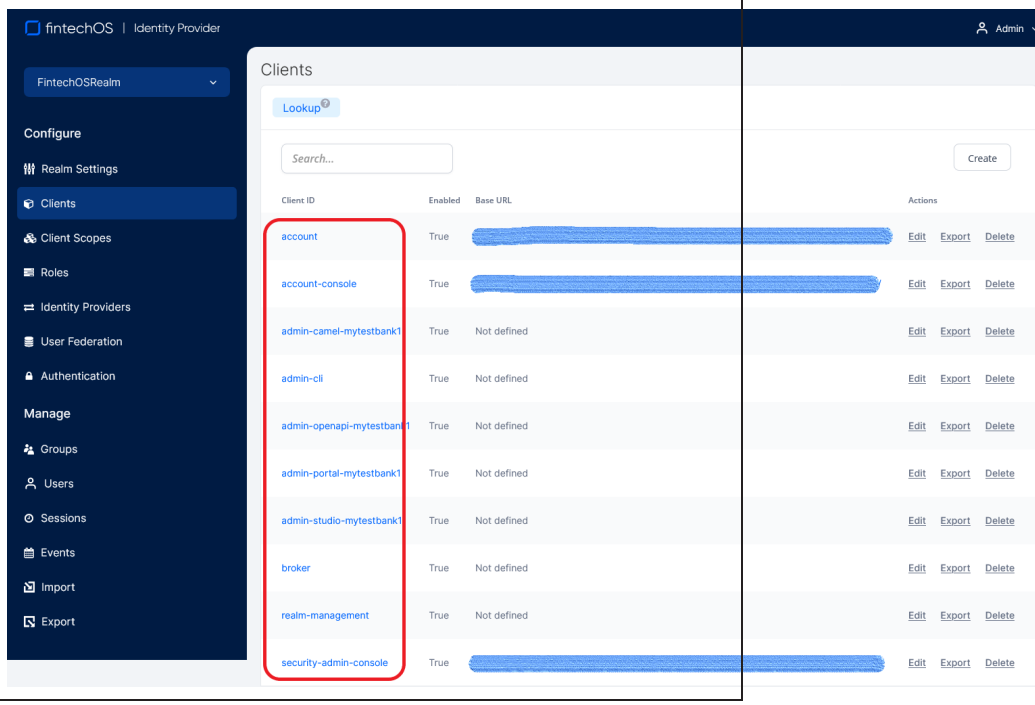
- "Using Azure AD as External Identity Provider" on page 134
- "Using Google as External Identity Provider" on page 140
- "Using Okta as External Identity Provider" on page 147
- "Using AWS Cognito as External Identity Provider" on page 153

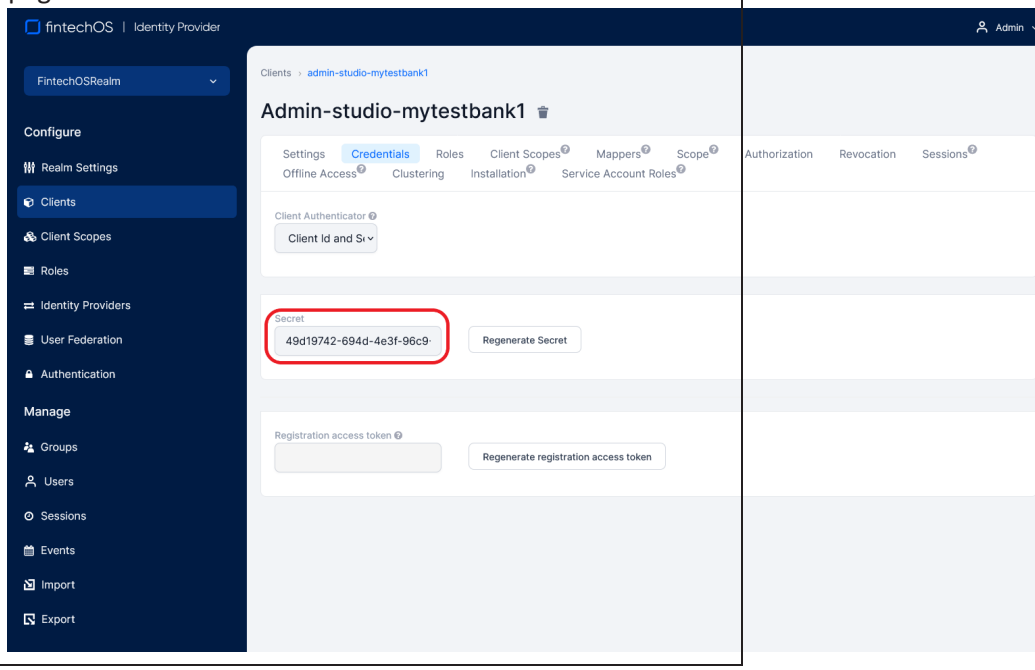
Identity Provider Settings

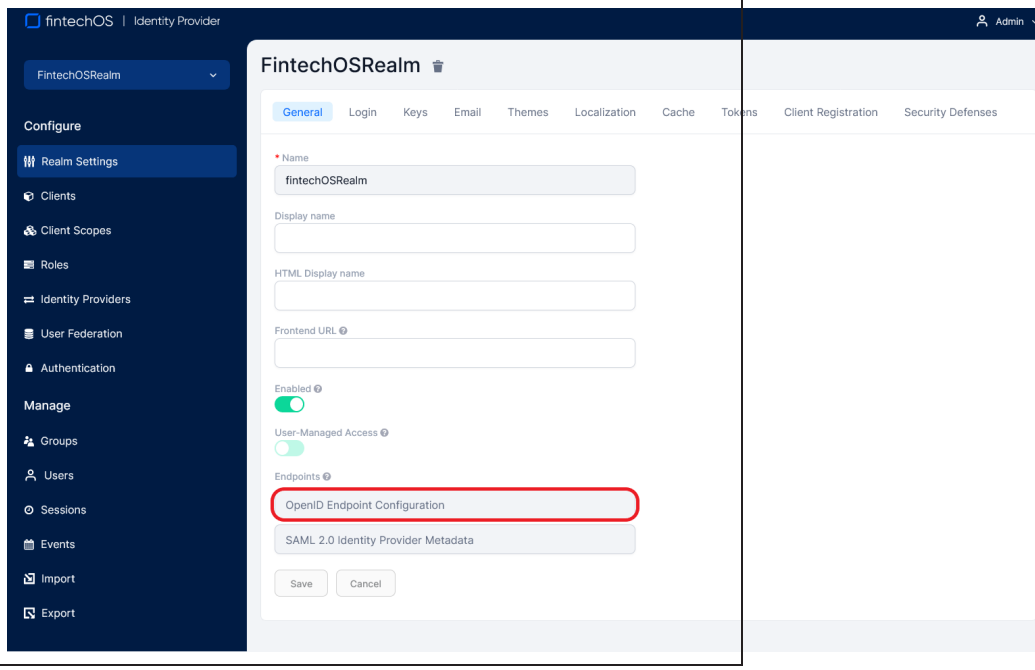
In the FintechOS Cloud Configuration Manager, set up the following secrets:

Key Path	Key Name
kv/<environment>/<application>/app-settings	EBSDefaultAuthentication
kv/<environment>/<application>/app-settings	core-setting-external-auth-provider-key-url
kv/<environment>/<application>/app-settings	core-setting-external-auth-provider-issuer
kv/<environment>/<application>/app-settings	openid-client-id
kv/<environment>/<application>/app-settings	openid-client-secret
kv/<environment>/<application>/app-settings	openid-discovery-endpoint
kv/<environment>/<application>/app-settings	openid-callback-url

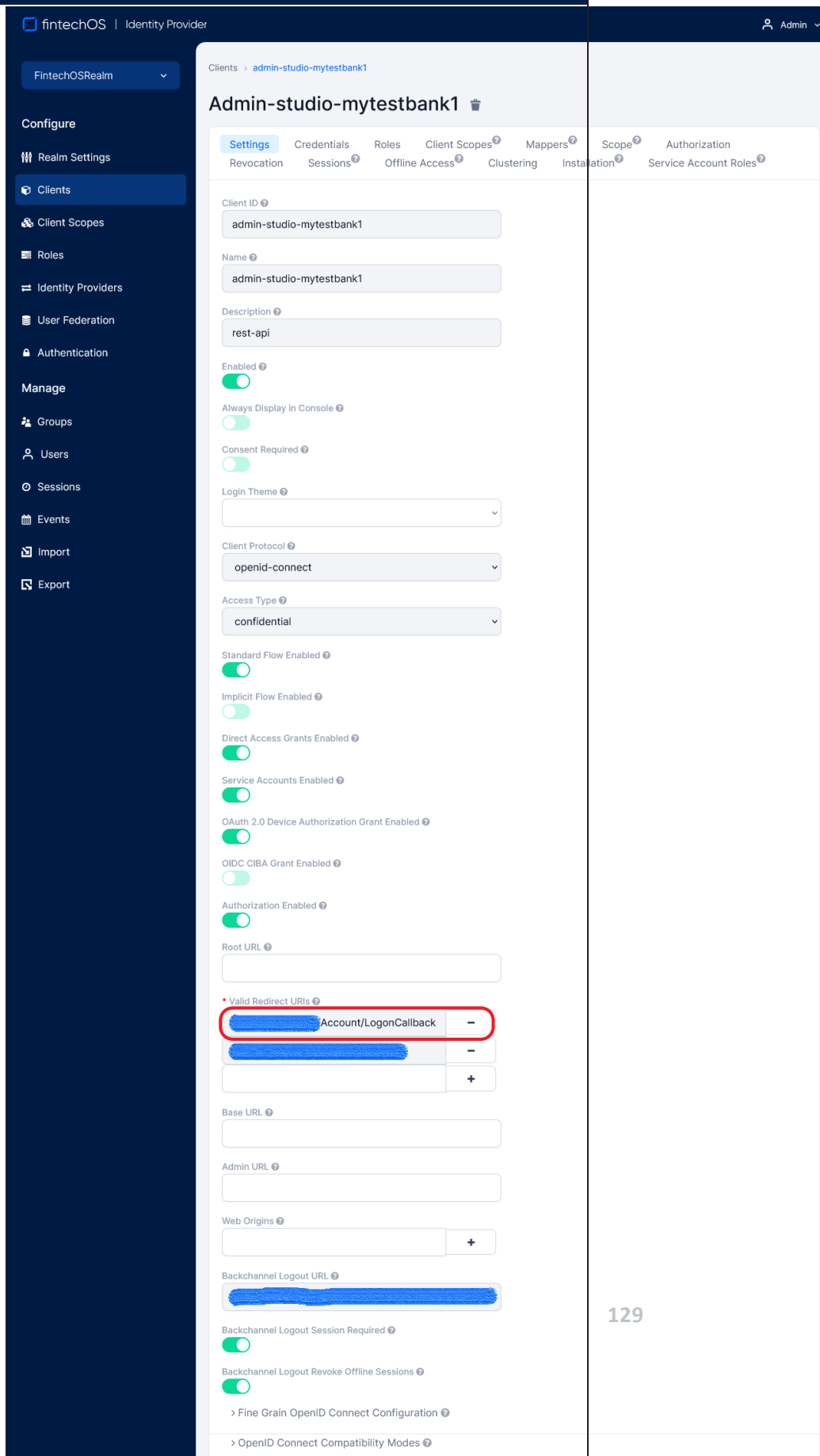
Key Name	Key Description
EBSDefaultAuthentication	Specifies the identity provider: • FTOSOIDC - FintechOS Identity Provider • EBS - Legacy FintechOS Platform authentication (deprecated) **IMPORTANT!** In a non-standard scenario where a regular portal using FintechOS Identity Provider is linked to a B2C portal using legacy authentication, it is recommended to have each portal reside on a separate domain. The requests will pass successfully only if coming from FTOS IDP towards B2C (the other way around, errors may occur). • AD - "Microsoft Active Directory Authentication" on page 160 (deprecated). • AzureAD - "Azure Active Directory Authentication" on page 166 (deprecated). • Okta - "Authentication with Okta" on page 172 (deprecated) • ADFS - "Authentication with Active Directory Federation Services" on page 180 (deprecated). • AWSCognito - "Authentication with AWS Cognito" on page 196 (deprecated)
core-setting-external-auth-provider-key-url	Link to the FintechOS Identity Provider public keys used to validate the digital signatures of the access tokens. E.g.: https://myHPFI.myDomain.com/auth/realms/fintechOSRealm/protocol/openid-connect/certs

Key Name	Key Description
core-setting-external-auth-provider-issuer	FintechOS Identity Provider instance identifier as provided in the issue field of the authentication token. This value is case sensitive. E.g.: https://myHPFI.myDomain.com/auth/realms/fintechOSRealm
openid-client-id	Your FintechOS Platform component's corresponding Client ID as defined in the FintechOS Identity Provider. E.g.: admin-portal, myInnovationStudio. In the FintechOS Identity Provider admin console, you can find the list of Client IDs in the Clients section of your FintechOS realm. 

Key Name	Key Description
openid-client-secret	Your FintechOS Platform component's corresponding client secret generated by the FintechOS Identity Provider. In the FintechOS Identity Provider admin console, you can find the client secret in the Credentials tab of your client's page. 

Key Name	Key Description
openid-discovery-endpoint	FintechOS Identity Provider endpoint. E.g.: https://myHPFI.myDomain.com/auth/realms/fintechOSRealm/.well-known/openid-configuration In the FintechOS Identity Provider admin console, you can find the discovery endpoint address in the Realm Settings section. 

Key Name	Key Description
openid-callback-url	URL where the user agent is redirected after a successful login. The default value is `{<Studio/Portal base URL>/Account/LogonCallback`. A matching entry must be configured in the FintechOS Identity Provider as a valid redirect URI for the client.

Key Name	Key Description
	

Users Log in the Portal or Studio

When accessing the FintechOS Platform Portal or FintechOS Studio, users who have an active OpenID session are logged in automatically. Otherwise, they are displayed the FintechOS Identity Provider single sign-on login page and will use the OpenID account credentials to log in to the FintechOS Platform Portal or FintechOS Studio.

Activate User Login Logs

To view user logins on a certain environment, you need to use the Save Events option in the Identity Provider. The data retrieved is user ID, IP address, session ID, errors.

1. Login to the **FintechOS Identity Provider** for your environment.
2. Go to **Realm settings > Events > User event settings**.
3. Toggle **Save events** to On. The logging is activated and in IDP database, the Event_Entity table is populated with information about the login sessions. An entry is created at each successful user login.
4. Use a query in the database to retrieve the list, for example: `SELECT id, client_id, details_json, error, ip_address, realm_id, session_id, event_time, type, user_id FROMevent_entity WHERE type = 'LOGIN';`.

To retrieve data about the current logged in user in Studio, use the ebs.getCurrentUserSessionsHistory Client SDK method.

FintechOS Platform User Account Automatic Synchronization

When a user logs in to FintechOS Platform Portal or FintechOS Studiousing the FintechOS Identity Provider single sign-on, the following information stored in the corresponding FintechOS Platform user account is updated automatically based on the FintechOS Identity Provider account settings:

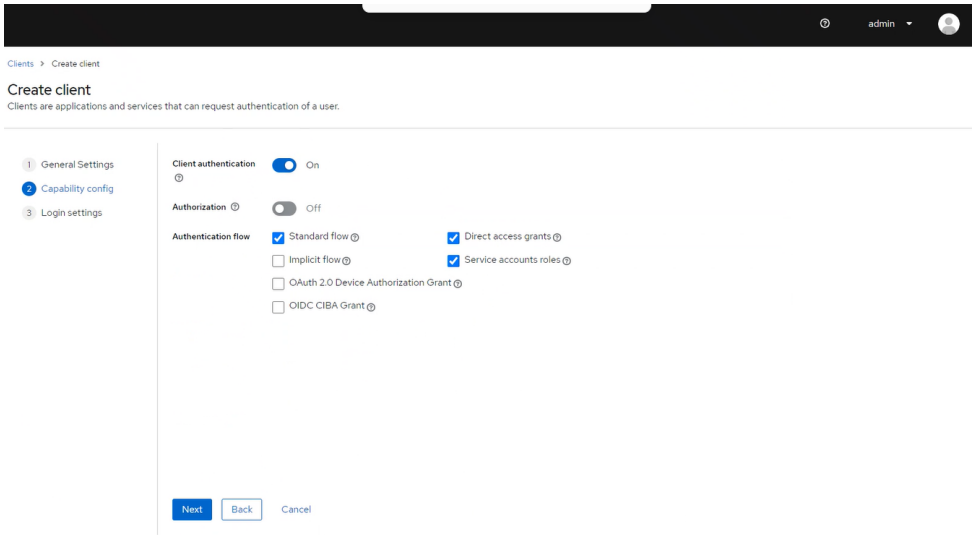
- First Name
- Last Name
- Email

- Security roles
- External user ID (uniquely identifies the user by an external identity provider)

Service Accounts

Service Accounts are a type of privileged accounts that gives the user elevated rights within the platform. Such accounts are set up in the FintechOS Identity Provider and are visible in Studio as well. They are client accounts and are not subject to regular password policies. This decreases the level of support needed to operate the platform and minimizes business disruption. Follow the steps below to create a service account:

1. Log in to the FintechOS Identity Provider admin console.
2. Select your FintechOS Platform realm.
3. Select the **Clients** blade, select **Create Client**.
4. Type in a name for the service account.
5. In the **Capability Config** section, switch **Client authentication** on, and select **Service accounts roles**, click **Save**.



6. To assign a role, go to **Service accounts roles** and select **Assign role**.
7. Choose a role from the list.
8. To make the account visible in Studio, in the Postman collection, add the new service account username to `auth.clientid`.
9. In the Postman collection, get the token for the user, and then use GetEntities Metadata. The user is now visible in **Studio > Security > System Users**.

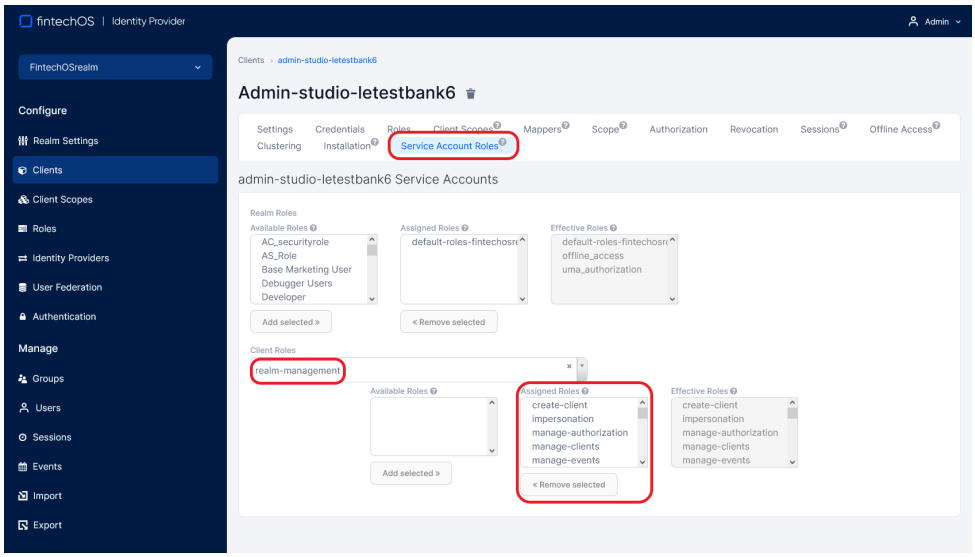
To unassign a role from a Service Account, use the options in the System Users menu entry.

Add Roles to Service Accounts

FintechOS Studio clients require full privileges for membership management (CRUD operations with users, password change/reset, etc.). Roles for service accounts can be added from the FintechOS IDP or Studio. The following steps show how to add roles from the FintechOS IDP. Check the Manage Users page for how to add roles from Studio.

1. Log in to the FintechOS Identity Provider admin console.
2. Select your FintechOS Platform realm.
3. Select the **Clients** blade.
4. Open the Studio client.
5. Go to the **Service Account Roles** tab.

6. In the Client Roles drop-down, select **realm-management** and assign all the available roles.



Deactivate Inactive Accounts

Using a scheduled job, you can configure inactive accounts to be automatically deactivated after a certain period of time. Inactive accounts are accounts with no successful logouts or session expiry for a period of time that you previously defined.

> **NOTE**
> Brokered users or service accounts or any technical identities used by platform components cannot be deactivated using this feature.

After the account is deactivated, the user receives an email informing them of this action. They can contact their administrator to have the account reactivated if needed. The administrator can activate the account by navigating in Studio to **Security > System Users > Inactive Users**, open the user details and tick the **IsAuthorized** box.

> **IMPORTANT!**
> To deactivate user accounts manually, go to Studio and follow the steps described on the System Users page.

Configure the Scheduled Job

Follow the steps below to configure the scheduled job for deactivating users that haven't logged on or been active for a predefined period of time:

1. Login to the Configuration Manager for your environment. Navigate to **Services** and **Triggers**.
2. Under **Services**, update the `userInactivityPeriod` parameter and add your own value. The default is 60 days.
3. Under **Triggers**, find the `FTOS.DisableInactiveUsersService` and update the `EndTime` parameter for running the job and the `Expression` for when the job should be ran.
4. Restart the Job Server app services.

Users receive an email informing them their accounts were deactivated and that they should contact their administrator for reactivation.

> **IMPORTANT!**
> If the capability is configured to monitor auth activity on multiple realm clients, then inactivity on any of them will lead to deactivation on all.

Using Azure AD as External Identity Provider

If your organization is using Azure Active Directory (AD) for identity and access management, you can configure the FintechOS Identity Provider to act as an identity broker, allowing users to log in to FintechOS applications and services using their existing Azure AD credentials.

1 Register the FintechOS Identity Provider as an Azure App

1. Log in to your **Azure Portal** and navigate to the **Azure Active Directory** blade.
2. Select **App Registrations**.
3. Click **+New registration**.
4. Enter a name for your app and choose who can access it.
5. Add a **Redirect URI** for the FintechOS Identity Provider. The Redirect URI is based on the Identity Provider alias you will set up for the app in the FintechOS Identity Provider and has the following structure:
   `https://{HPFI base URL}/auth/realms/{realm name}/broker/{alias}/endpoint`
   E.g.:
   `https://myHpfi.myDomain/auth/realms/FintechOSRealm/broker/AzureAD/endpoint`
6. **Save** your changes.
7. In the newly created app, select **Certificates and Secrets**.
8. In the **Client secrets** section, click **+New client secret** to generate a secret string for the FintechOS Identity Provider identification.

(Optional) Configure Access for Azure AD Users

By default, user assignment is not required, allowing any user to access the app. You should restrict access to the app only to specific assigned users or groups. To do so:

1. In the newly created app, select **Manage Application in Local Directory**.
2. Select **Properties**, and toggle **User assignment required** to Yes.
3. Select **Users and Groups** and assign the desired app users or groups.

Grant Consent to Access APIs

Apps are authorized to call APIs when they are granted permissions as part of the consent process. In order to authorize the app:

1. In the newly created app, select **API Permissions**.
2. In the **Grant admin consent for** enter the Azure Directory (tenant) ID.

2 Set up the Azure AD App as Identity Provider in the FintechOS Identity Provider

1. Log in to the FintechOS Identity Provider admin console and select your FintechOS realm.
2. Open the **Identity Providers** section.
3. From the **Add provider...** drop down, select **OpenID Connect v1.0**.

4. Fill in the following configuration settings for the Azure AD app.

Setting	Value
Alias	Identity provider alias you set for the Azure AD app (see " 1 Register the FintechOS Identity Provider as an Azure App" on page 135).
Display Name	User friendly name for the Azure AD app.
Enabled	ON
Trust Email	ON
First Login Flow	Leave the default value or select a custom login flow.
Sync Mode	Force
Authorization URL	Use your Azure Tenant ID (found in the App Registration details section of your Azure AD app).
Token URL	Use your Azure Tenant ID (found in the App Registration details section of your Azure AD app).
Client Authentication	Client secret sent as post.
Client ID	Use your App Registration Client ID (found in the App Registration details section of your Azure AD app).
Client Secret	Use the client secret set up previously (see " 1 Register the FintechOS Identity Provider as an Azure App" on page 135).
Default Scopes	Scopes to be sent when asking for authorization. Default: openid email.

5. Click **Save**.

3 Map Azure AD Security Groups to FintechOS Security Roles

When users log in, information about their security roles must be retrieved from Azure AD. For this purpose, you must set up an automatic mapping between Azure AD security groups and FintechOS Identity Provider security roles.

Set Up the ID Tokens Sent by Azure AD to Include Security Groups Information

You can include security groups information in the ID tokens sent by Azure AD as an optional claim. To do so, in the Azure app you created earlier (see " 1 Register the FintechOS Identity Provider as an Azure App" on page 135), modify the app registration manifest based on the following model:

```
"optionalClaims": {
    "idToken": [
        {
            "name": "groups",
            "source": null,
            "essential": false,
            "additionalProperties": []
        }
    ],
    "accessToken": [
        {
            "name": "groups",
            "source": null,
            "essential": false,
            "additionalProperties": []
        }
    ],
    "saml2Token": [
        {
            "name": "groups",
            "source": null,
            "essential": false,
            "additionalProperties": []
        }
    ]
}
```

Define Mappings between Azure AD Security Groups and FintechOS Security Roles

1. Log in to the FintechOS Identity Provider admin console and select your FintechOS realm.
2. Open the **Identity Providers** section.

3. Select the Azure AD app you created earlier (see " 2 Set up the Azure AD App as Identity Provider in the FintechOS Identity Provider" on page 136).
4. Open the **Mappers** tab.
5. For each security role, do the following:
    a. Click **Create**.
    b. In the **Add Identity Provider Mapper** window, fill in the following fields:

Setting	Value
Name	Enter a descriptive name for the mapper.
Sync Mode Override	force
Mapper Type	Claim to Role
Claim	groups
Claim Value	GUID of the security group set up in Azure AD.
Role	Select the corresponding FintechOS security role.

    c. Click **Save**.

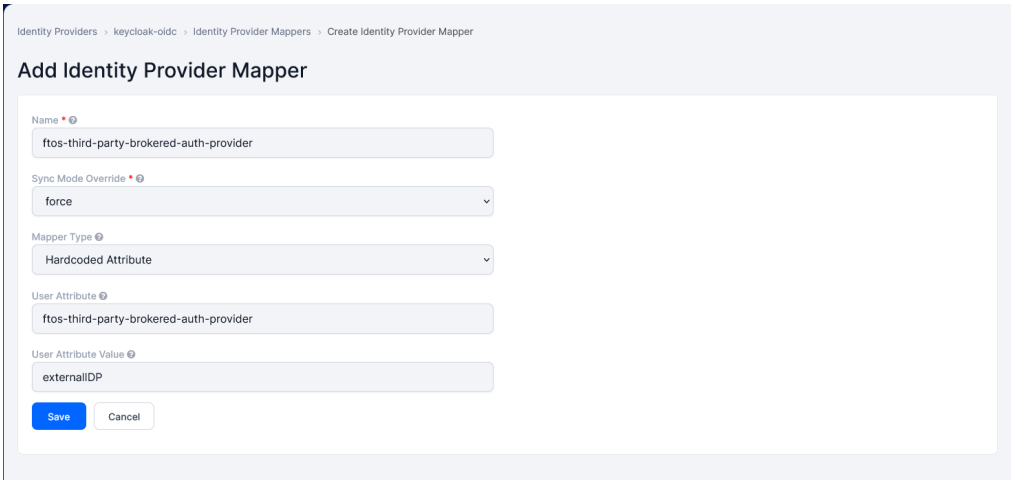
4 Disable User Account Editing in Innovation Studio

Users who authenticate in FintechOS Platform via an external identity provider cannot have their user account information edited in FintechOS Studio as modifications cannot be propagated back to the external identity provider.

In order to protect the user name, first name, last name, display name, email, and phone number fields, as well as the password reset button in the FintechOS Studio interface, a hardcoded *ftos-third-party-brokered-auth-provider* attribute mapping must be provided by the FintechOS Identity Provider for such user accounts:

1. Log in to the FintechOS Identity Provider admin console and select your FintechOS realm.
2. Open the **Identity Providers** section.
3. Open your external identity provider and select the **Mappers** tab.

4. Click **Create** to create a new mapper.
5. Fill in the following fields:
    - **Name** - Provide a name for your mapper
    - **Sync Mode Override** - force
    - **Mapper Type** - Hardcoded attribute
    - **User attribute** - ftos-third-party-brokered-auth-provider
    - **User attribute value** - Any non-null value will work, but it is recommended to use a value that is meaningful for your external identity provider, such as AzureAD or Okta.



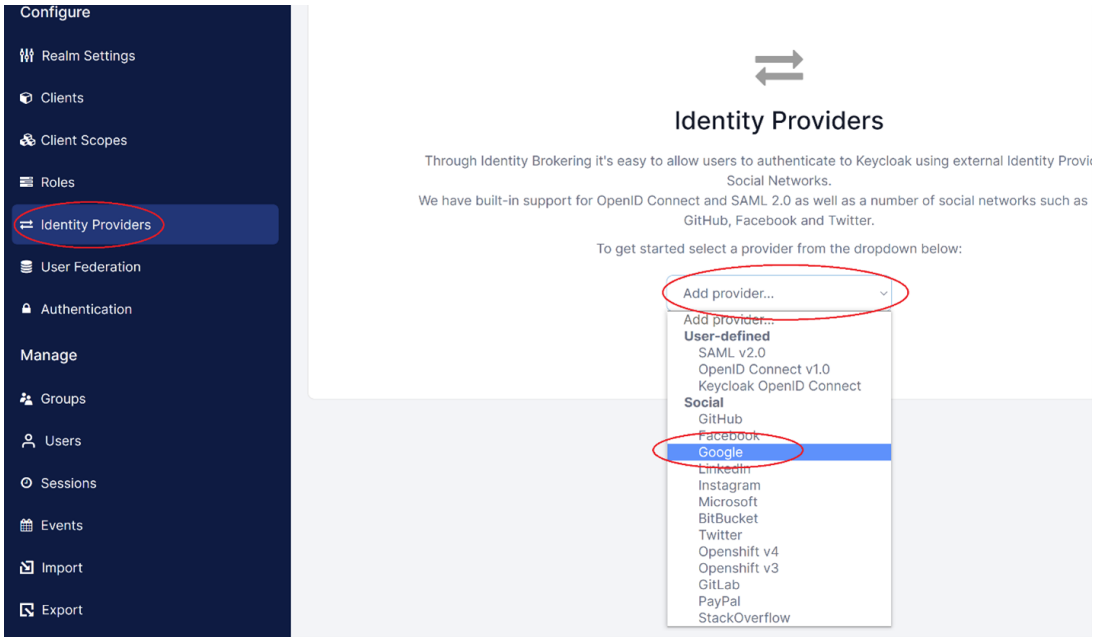
6. Click **Save**.

Using Google as External Identity Provider

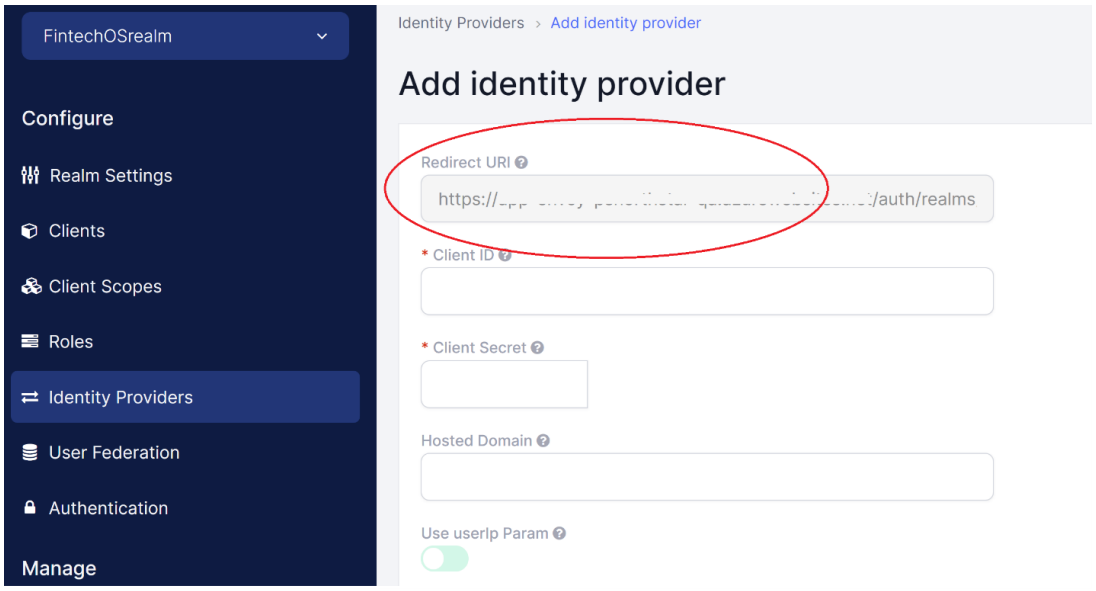
If your organization is using Google for identity and access management, you can configure the FintechOS Identity Provider to act as an identity broker, allowing users to log in to FintechOS applications and services using their existing Google credentials.

1 Identify the Redirect URI in FintechOS

1. Login to FintechOS Identity Provider.
2. Go to **Identity Providers** > **Add provider** > select **Google** from the drop-down.



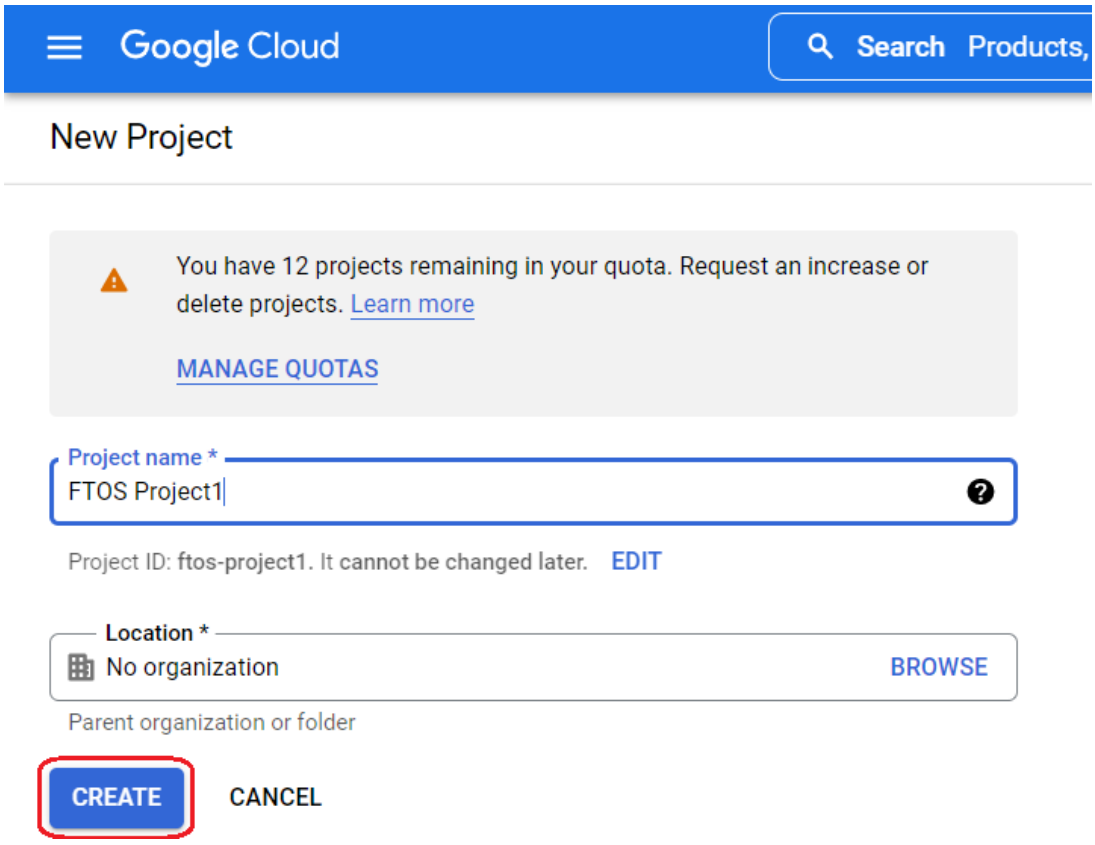
3. Identify the **Redirect URI** value and make a note of it. You will need it later.



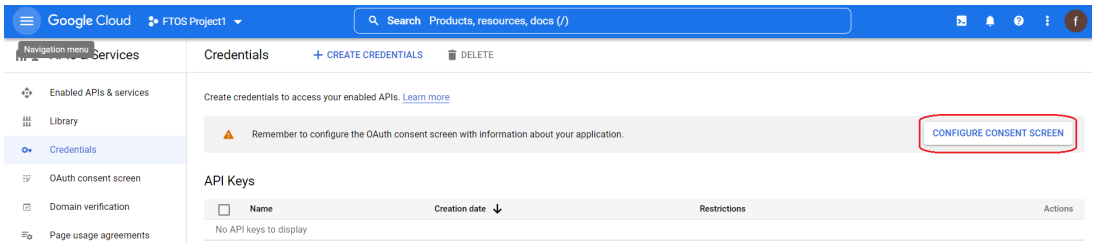
2 Create a Google App for the FintechOS Identity Provider

> **NOTE** The external Identity provider is not owned or maintained by FintechOS. Configuration and maintenance of the external Identity provider is fully performed and owned by the customer. The following steps are provided by FintechOS as a guideline to understand a minimum configuration required on client side, in order to integrate the external provider in FintechOS.

1. Go to this page and sign in with your Google account to get to the Google Developers Console.
2. Select **CREATE PROJECT** and provide a **Project Name**, then select **CREATE**.



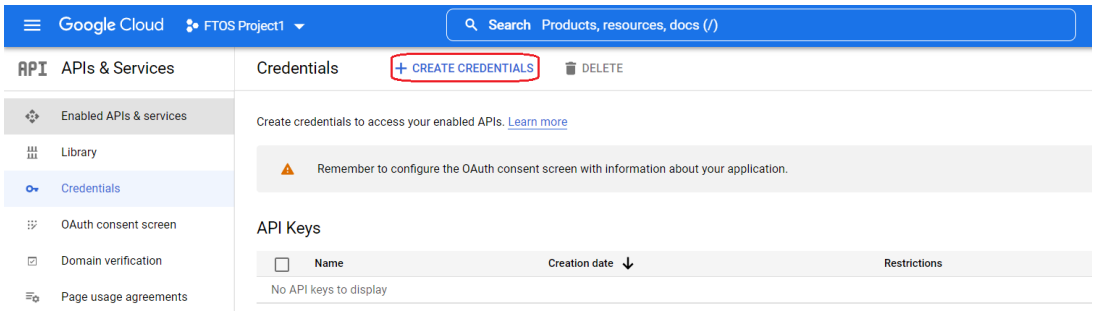
3. Go to the **Credentials** tab and select **CONFIGURE SELECT SCREEN**. This step is mandatory for the integration and it configures what the users see when they are redirected to Google for signing in.



4. In the **OAuth consent screen** page, check **External**, then select **CREATE**.

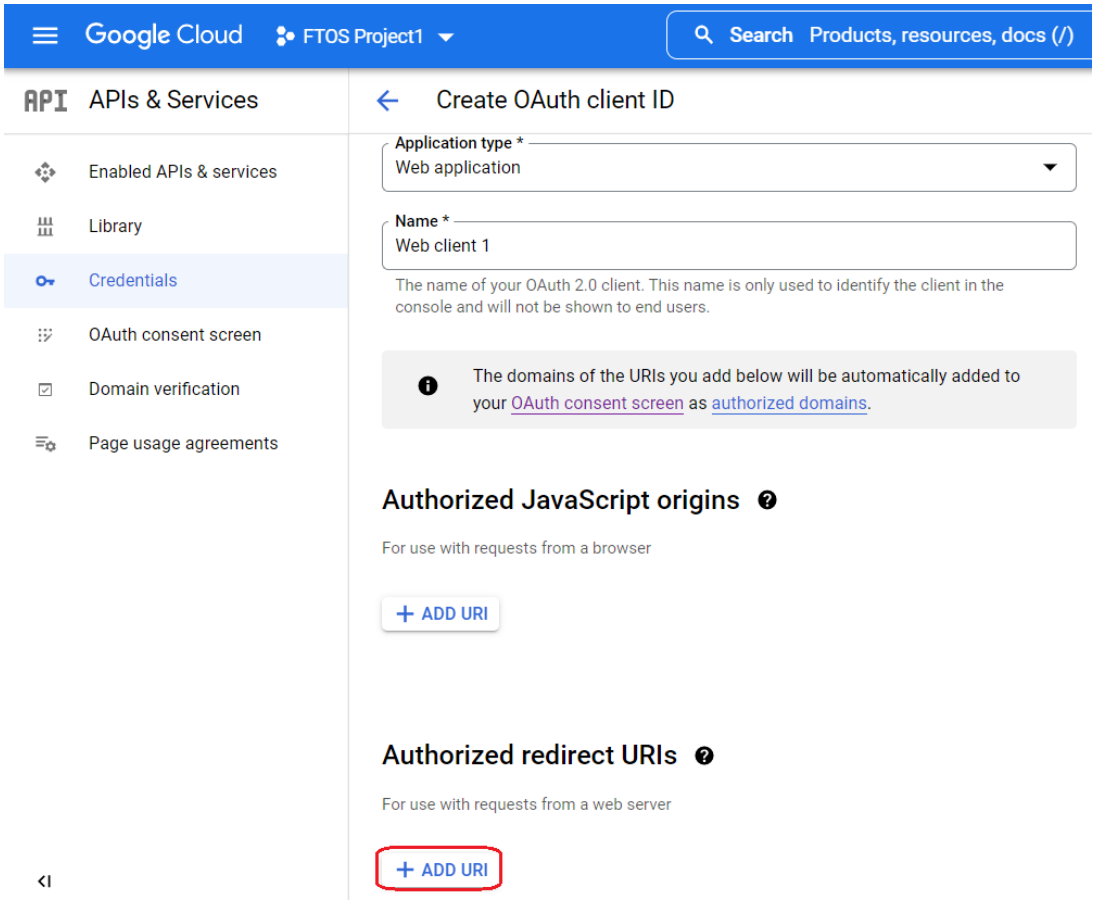
5. Next, in the **Edit app registration** page, fill-in the mandatory fields (**App name**, **User support email**, and **Developer contact information**). Then, select **SAVE AND CONTINUE**.

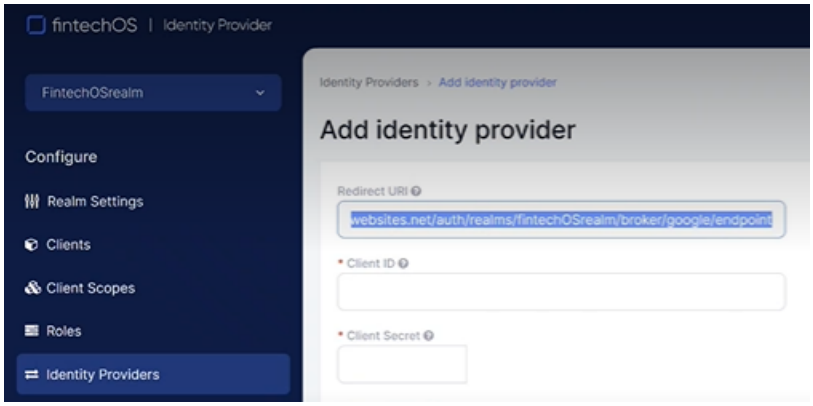
6. Go to the **Credentials** tab again and select **+CREATE CREDENTIALS** > **OAuth client ID**.



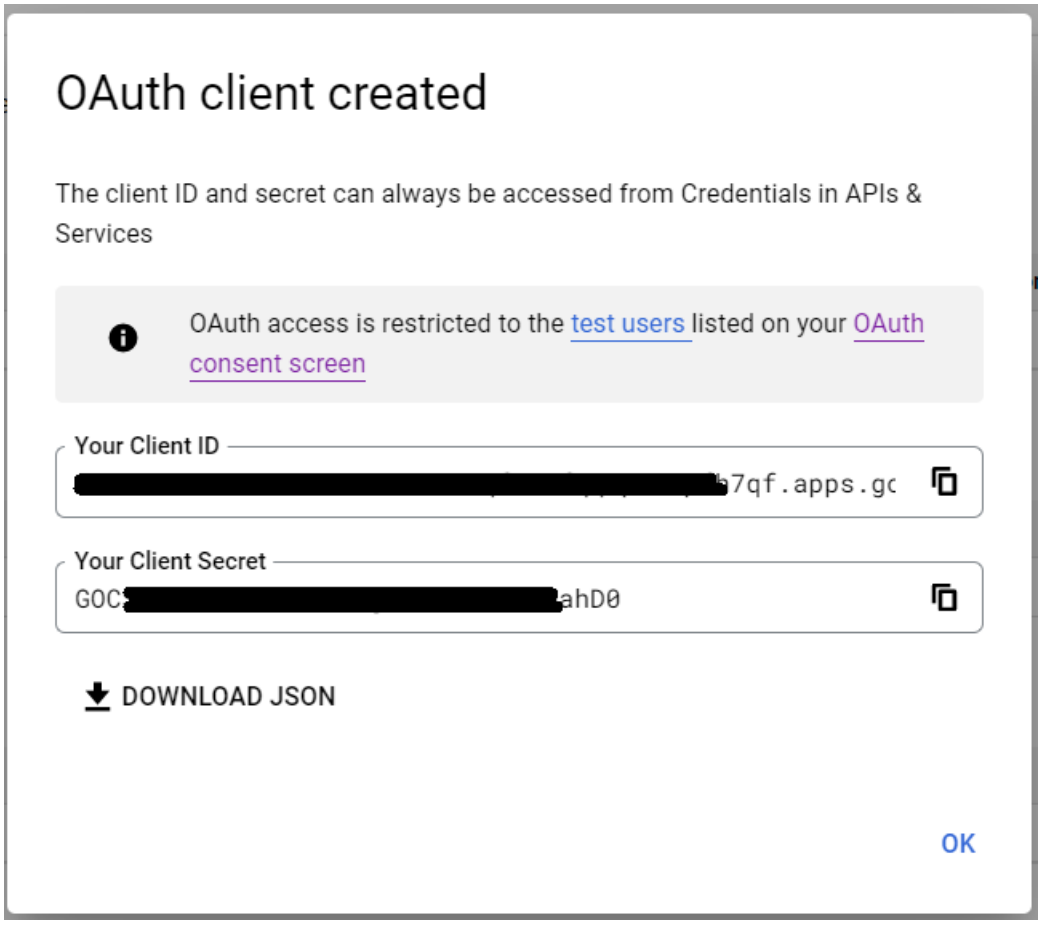
7. In the **Create OAuth client ID** page, select **Web application** from the drop-down, and select **+ADD URI** in the **Authorized redirect URIs** section. This is the value found in the FintechOS Identity Provider admin console, section **Identity Providers** (Redirect URI field), and it has the following structure: `https://{HPFI base URL}/auth/realms/{realm name}/broker/{alias}/endpoint`.

> **IMPORTANT!** The Redirect URI value should match the FintechOS platform IDP. If this field is not correctly configured, the authentication process will return an error.
>
> Multiple Redirect URIs can be configured during this step in order to integrate the same external provider with multiple applications, platforms, or portals.


Google Cloud
FTOS Project1
Search Products, resources, docs (/)
APIs & Services
Enabled APIs & services
Library
Credentials
OAuth consent screen
Domain verification
Page usage agreements
Create OAuth client ID
Application type *
Web application
Name *
Web client 1
The name of your OAuth 2.0 client. This name is only used to identify the client in the console and will not be shown to end users.
The domains of the URIs you add below will be automatically added to your OAuth consent screen as authorized domains.
Authorized JavaScript origins
For use with requests from a browser
ADD URI
Authorized redirect URIs
For use with requests from a web server
ADD URI


fintechOS | Identity Provider
FintechOSrealm
Configure
Realm Settings
Clients
Client Scopes
Roles
Identity Providers
Identity Providers > Add identity provider
Add identity provider
Redirect URI
websites.net/auth/realms/fintechOSrealm/broker/google/endpoint
* Client ID
* Client Secret

8. Select **Create** and make a note of the **Client ID** and **Client Secret**, you will need them while setting up the FintechOS Identity Provider.



3 Set up Google as Identity Provider in the FintechOS Identity Provider

1. Log in to the FintechOS Identity Provider admin console and select your FintechOS realm.
2. Open the **Identity Providers** section.
3. From the **Add provider...** drop down, select **Google**.

4. Fill in the following configuration settings for the Google server.

Setting	Value
Client ID	Use your Google client ID (from the Google console, Credentials tab).
Client Secret	Use your Google client secret (from the Google console, Credentials tab).
Enabled	ON
First Login Flow	first broker login
Sync Mode	import
Hosted Domain	(*) When * is entered, any hosted account can be used

5. Click **Save**.

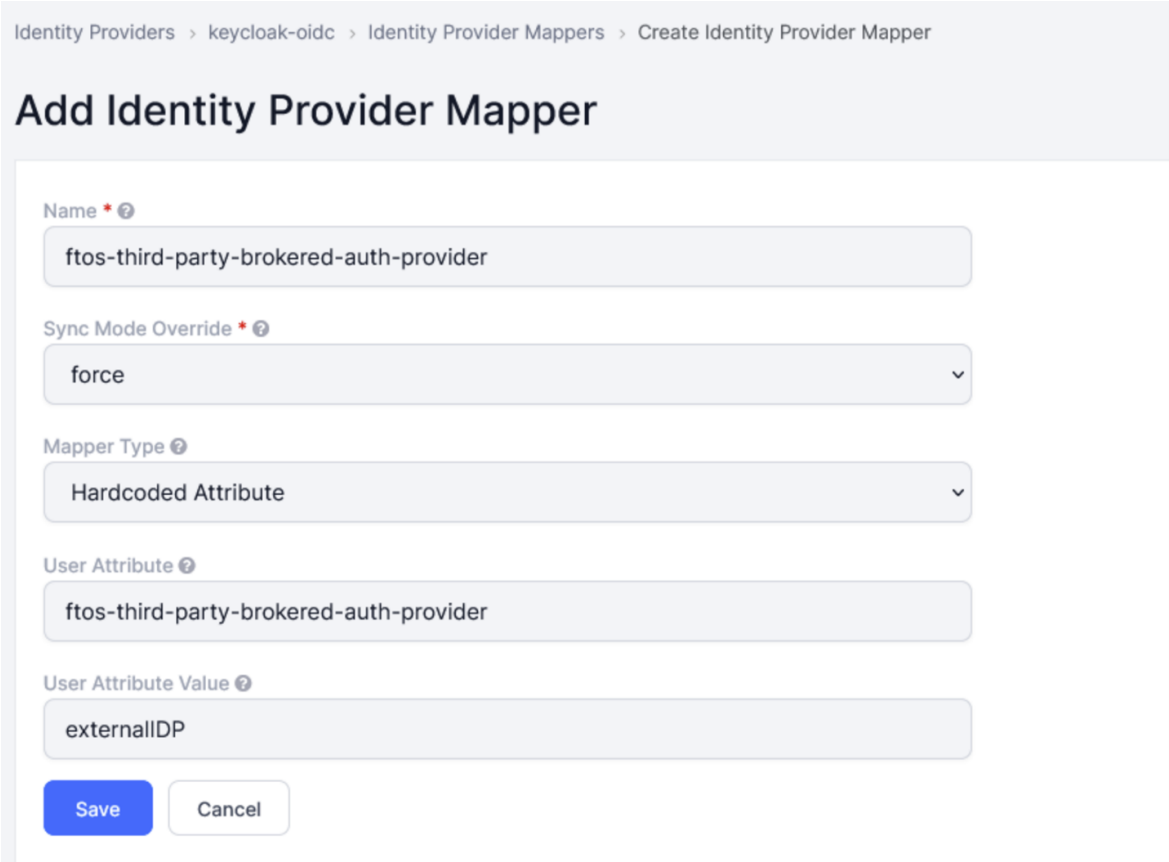
4 Configure the Attribute Mapper

Users who authenticate in FintechOS Platform via an external identity provider cannot have their user account information edited in FintechOS Studio, as modifications cannot be propagated back to the external identity provider.

In order to identify external users in FintechOS Studio, hardcoded *ftos-third-party-brokered-auth-provider* attribute mapping must be provided by the FintechOS Identity Provider for such user accounts:

1. Log in to the FintechOS Identity Provider admin console and select your FintechOS realm.
2. Open the **Identity Providers** section.
3. Open your external identity provider and select the **Mappers** tab.
4. Click **Create** to create a new mapper and fill in the following fields:
    - **Name** - Provide a name for your mapper
    - **Sync Mode Override** - force

- **Mapper Type** - Hardcoded attribute
- **User attribute** - ftos-third-party-brokered-auth-provider
- **User attribute value** - Any non-null value will work, but it is recommended to use a value that is meaningful for your external identity provider, such as AzureAD or Okta.



5. Click **Save**.

Using Okta as External Identity Provider

If your organization is using Okta for identity and access management, you can configure the FintechOS Identity Provider to act as an identity broker, allowing users to log in to FintechOS applications and services using their existing Okta credentials.

1 Create an Okta App Integration for the FintechOS Identity Provider

Sign in to your Okta admin console with your administrator account and create an app integration for the FintechOS Identity Provider. The **Sign-in redirect URI** must match the identity provider alias you will set up for Okta in the FintechOS Identity Provider and has the following structure:

`https://{HPFI base URL}/auth/realms/{realm name}/broker/{alias}/endpoint`
E.g.:
`https://myHpfi.myDomain/auth/realms/FintechOSRealm/broker/Okta/endpoint`

Make a note of the following Okta settings which you will have to provide in the FintechOS Identity Provider for integration:

- **Okta client ID** (from the Okta app, General tab)
- **Okta client secret** (from the Okta app, General tab)
- **Well-known configuration** - This is an endpoint that returns the OpenID Connect metadata related to the Okta authorization server and has the following URL template: `https://{oktaDomain}.okta.com/oauth2/{oktaServer}/.well-known/openid-configuration`

2 Set up the Okta server as Identity Provider in the FintechOS Identity Provider

1. Log in to the FintechOS Identity Provider admin console and select your FintechOS realm.
2. Open the **Identity Providers** section.

3. From the **Add provider...** drop down, select **OpenID Connect v1.0**.

4. Fill in the following configuration settings for the Okta server.

Setting	Value
Alias	Identity provider alias you set for the Okta server (see " 1 Create an Okta App Integration for the FintechOS Identity Provider" on the previous page).
Display Name	User friendly name for the Okta server.
Enabled	ON
First Login Flow	first broker login
Sync Mode	force
Authorization URL	`https://{oktaDomain}.okta.com/oauth2/{oktaServer}/authorize`
Token URL	`https://{oktaDomain}.okta.com/oauth2/{oktaServer}/token`
Logout URL	`https://{oktaDomain}.okta.com/oauth2/{oktaServer}/logout`
User Info URL	`https://{oktaDomain}.okta.com/oauth2/{oktaServer}/userinfo`
Client Authentication	Client secret sent as post.
Client ID	Use your Okta client ID (from the Okta app, General tab).
Client Secret	Use your Okta client secret (from the Okta app, General tab).
Issuer	`https://{oktaDomain}.okta.com/oauth2/{oktaServer}`
Default Scopes	Scopes to be sent when asking for authorization. Default: openid email.
Prompt	unspecified

Setting	Value
Validate Signatures	ON
Use JWKS URL	ON
JWKS URL	`https://{oktaDomain}.okta.com/oauth2/{oktaServer}/keys`

5. Click **Save**.

3 Map Okta User Groups to FintechOS Security Roles

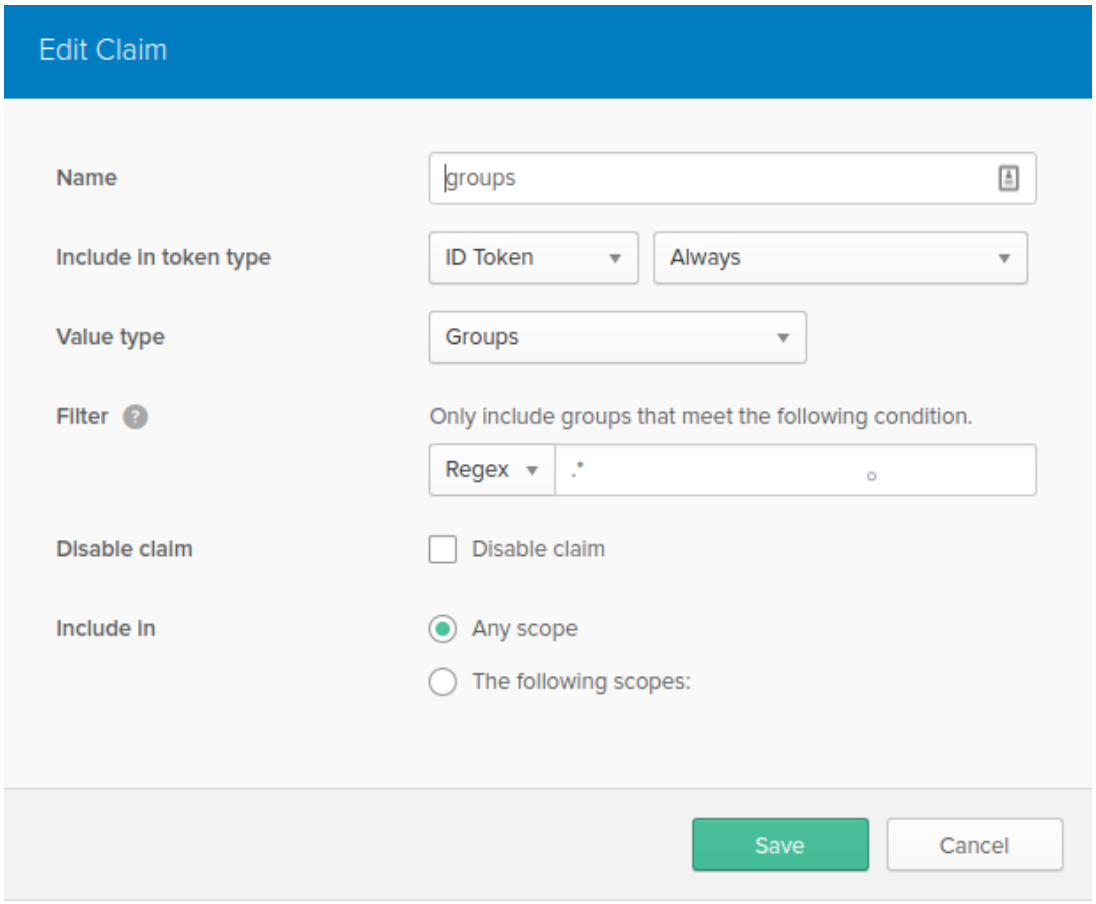
When users log in, information about their security roles must be retrieved from the Okta server. For this purpose, you must set up an automatic mapping between Okta user groups and FintechOS Identity Provider security roles.

Set Up the ID Tokens Sent by Okta to Include Security Groups Information

You can include user groups information in the ID tokens sent by Okta as an optional claim. To do so, in the in the Okta portal:

1. Hover over the **API** menu item and select **Authorization Servers**.
2. Select your Okta authorization server.
3. Open the **Claims** tab and click **Add Claim**. Fill in the following fields:

Field	Value
Name	groups
Include In token type	ID Token | Always
Value Type	Groups
Filter	Regex | .*
Disable claim	Uncheck
Include In	Any scope



4. Click **Save**.

Define Mappings between Okta Groups and FintechOS Security Roles

1. Log in to the FintechOS Identity Provider admin console and select your FintechOS realm.
2. Open the **Identity Providers** section.
3. Select the Okta server you added earlier (see " 2 Set up the Okta server as Identity Provider in the FintechOS Identity Provider" on page 148).
4. Open the **Mappers** tab.

5. For each security role, do the following:

    a. Click **Create**.

    b. In the **Add Identity Provider Mapper** window, fill in the following fields:

Setting	Value
Name	Enter a descriptive name for the mapper.
Sync Mode Override	legacy
Mapper Type	Claim to Role
Claim	groups
Claim Value	Name of the Okta group set up on the Okta server.
Role	Select the corresponding FintechOS security role.

    c. Click **Save**.

4 Disable User Account Editing in Innovation Studio

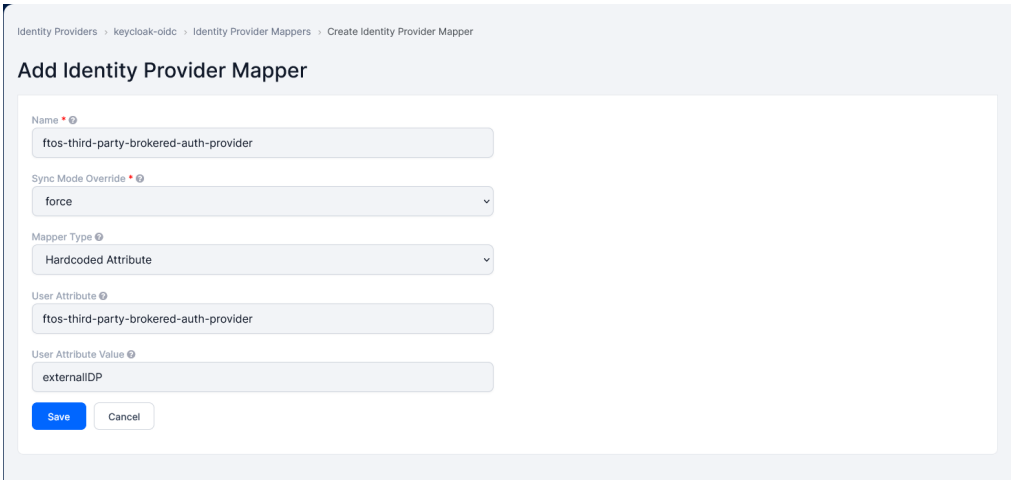
Users who authenticate in FintechOS Platform via an external identity provider cannot have their user account information edited in FintechOS Studio as modifications cannot be propagated back to the external identity provider.

In order to protect the user name, first name, last name, display name, email, and phone number fields, as well as the password reset button in the FintechOS Studio interface, a hardcoded *ftos-third-party-brokered-auth-provider* attribute mapping must be provided by the FintechOS Identity Provider for such user accounts:

1. Log in to the FintechOS Identity Provider admin console and select your FintechOS realm.
2. Open the **Identity Providers** section.
3. Open your external identity provider and select the **Mappers** tab.
4. Click **Create** to create a new mapper.

5. Fill in the following fields:

    - **Name** - Provide a name for your mapper
    - **Sync Mode Override** - force
    - **Mapper Type** - Hardcoded attribute
    - **User attribute** - ftos-third-party-brokered-auth-provider
    - **User attribute value** - Any non-null value will work, but it is recommended to use a value that is meaningful for your external identity provider, such as AzureAD or Okta.



6. Click **Save**.

Using AWS Cognito as External Identity Provider

If your organization is using AWS Cognito for identity and access management, you can configure the FintechOS Identity Provider to act as an identity broker, allowing users to log in to FintechOS applications and services using their existing AWS Cognito credentials.

1 Create an AWS Cognito App for the FintechOS Identity Provider

Sign in to your AWS Cognito console with your administrator account and create an app for the FintechOS Identity Provider. The **Callback URL** must match the identity provider alias you will set up for AWS Cognito in the FintechOS Identity Provider and

has the following structure:

`https://{HPFI base URL}/auth/realms/{realm name}/broker/{alias}/endpoint`
E.g.:
`https://myHpfi.myDomain/auth/realms/FintechOSRealm/broker/awsCognito/endpoint`

Make a note of the following AWS Cognito app settings which you will have to provide in the FintechOS Identity Provider for integration:

- **AWS Cognito client ID**
- **AWS Cognito client secret**
- **AWS Cognito domain**
- **Pool ARN** - The region and pool ID will be extracted from the pool ARN to determine the discovery endpoint. This is an endpoint that returns the OpenID Connect metadata related to the AWS Cognito authorization server and has the following URL template: `https://cognito-idp.{region}.amazonaws.com/{poolId}/.well-known/openid-configuration`

2 Set up the AWS Cognito server as Identity Provider in the FintechOS Identity Provider

1. Log in to the FintechOS Identity Provider admin console and select your FintechOS realm.
2. Open the **Identity Providers** section.
3. From the **Add provider...** drop down, select **OpenID Connect v1.0**.
4. Fill in the following configuration settings for the AWS Cognito server.

Setting	Value
Alias	Identity provider alias you set for the AWS Cognito server (see " 1 Create an AWS Cognito App for the FintechOS Identity Provider" on the previous page).

Setting	Value
Display Name	User friendly name for the AWS Cognito server.
Enabled	ON
First Login Flow	first broker login
Sync Mode	force
Authorization URL	`https://{Amazon Cognito domain}/oauth2/authorize` E.g: `https://myDomainPrefix.auth.eu-west-1.amazoncognito.com/oauth2/authorize`
Token URL	`https://{Amazon Cognito domain}/oauth2/token` E.g: `https://myDomainPrefix.auth.eu-west-1.amazoncognito.com/oauth2/token`
User Info URL	`https://{Amazon Cognito domain}/oauth2/userinfo` E.g: `https://myDomainPrefix.auth.eu-west-1.amazoncognito.com/oauth2/userinfo`
Client Authentication	Client secret sent as post.
Client ID	Use your AWS Cognito client ID.
Client Secret	Use your AWS Cognito client secret.
Issuer	`https://cognito-idp.{region}.amazonaws.com/{poolId}`
Default Scopes	Scopes to be sent when asking for authorization. Default: aws.cognito.signin.user.admin email openid phone profile.
Prompt	unspecified
Validate Signatures	ON
Use JWKS URL	ON

Setting	Value
JWKS URL	`https://cognito-idp.{region}.amazonaws.com/{poolId}/.well-known/jwks.json`

5. Click **Save**.

3 Map AWS Cognito User Groups to FintechOS Security Roles

When users log in, information about their security roles must be retrieved from the AWS Cognito server. For this purpose, you must set up an automatic mapping between AWS Cognito user groups and FintechOS Identity Provider security roles.

1. Log in to the FintechOS Identity Provider admin console and select your FintechOS realm.
2. Open the **Identity Providers** section.
3. Select the AWS Cognito server you added earlier (see " 2 Set up the AWS Cognito server as Identity Provider in the FintechOS Identity Provider" on page 154).
4. Open the **Mappers** tab.
5. For each security role, do the following:
    a. Click **Create**.
    b. In the **Add Identity Provider Mapper** window, fill in the following fields:

Setting	Value
Name	Enter a descriptive name for the mapper.
Sync Mode Override	force
Mapper Type	Claim to Role
Claim	cognito:groups
Claim Value	Name of the AWS Cognito group set up on the AWS Cognito server.
Role	Select the corresponding FintechOS security role.

c. Click **Save**.

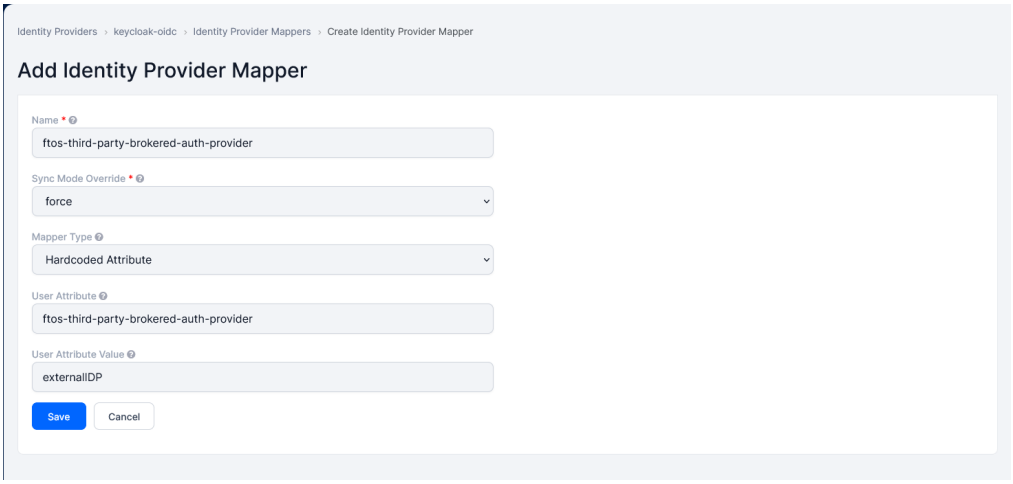
4 Disable User Account Editing in Innovation Studio

Users who authenticate in FintechOS Platform via an external identity provider cannot have their user account information edited in FintechOS Studio as modifications cannot be propagated back to the external identity provider.

In order to protect the user name, first name, last name, display name, email, and phone number fields, as well as the password reset button in the FintechOS Studio interface, a hardcoded *ftos-third-party-brokered-auth-provider* attribute mapping must be provided by the FintechOS Identity Provider for such user accounts:

1. Log in to the FintechOS Identity Provider admin console and select your FintechOS realm.
2. Open the **Identity Providers** section.
3. Open your external identity provider and select the **Mappers** tab.
4. Click **Create** to create a new mapper.
5. Fill in the following fields:
    - **Name** - Provide a name for your mapper
    - **Sync Mode Override** - force
    - **Mapper Type** - Hardcoded attribute
    - **User attribute** - ftos-third-party-brokered-auth-provider
    - **User attribute value** - Any non-null value will work, but it is recommended to use a value that is meaningful for your external identity provider, such as

AzureAD or Okta.

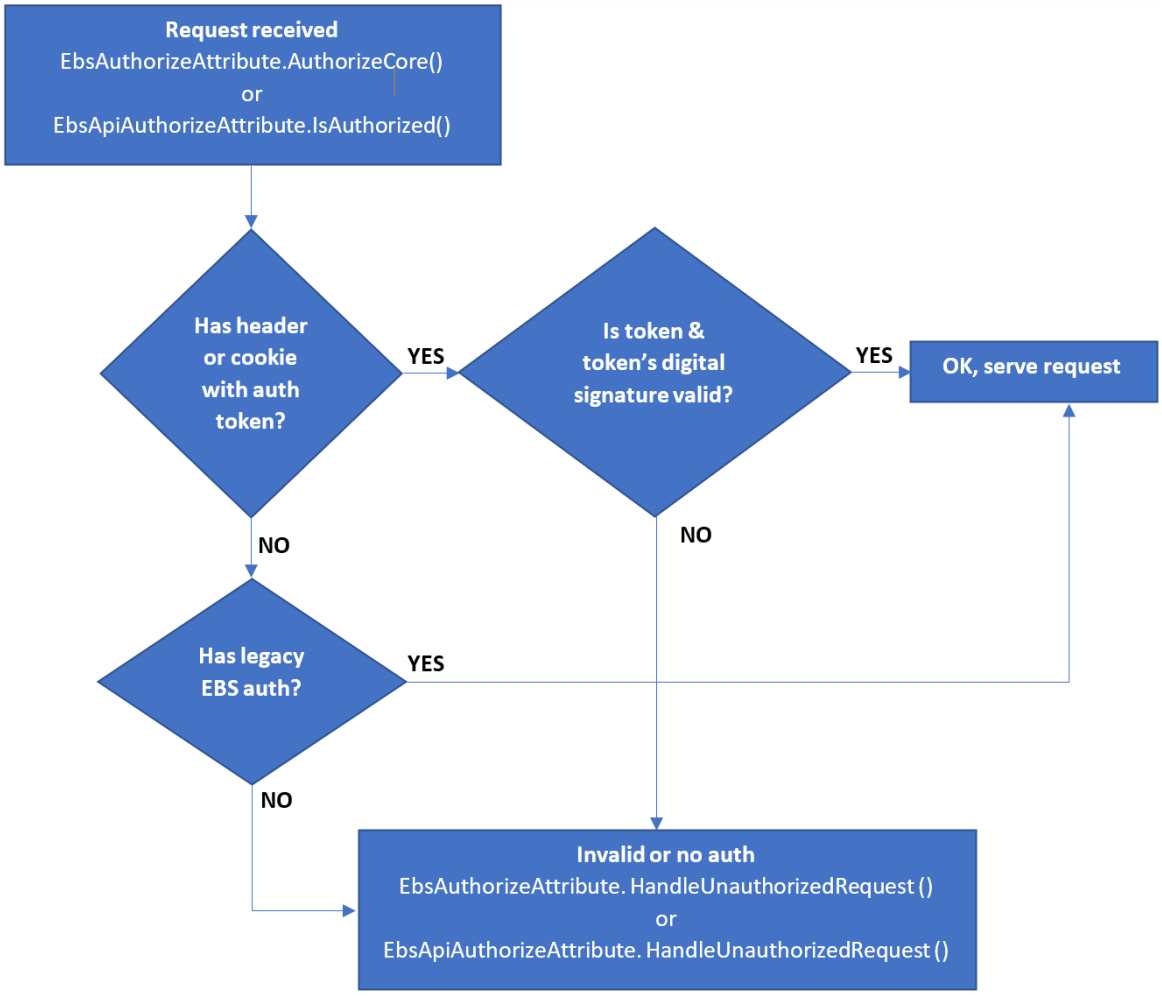


6. Click **Save**.

Authentication SDK

Authentication flow

Any URL called from either the platform or portal passes through a reverse proxy that checks if prior authentication was made (the check is made against a cookie). If authentication exists, the login is successful, and the URL is accessed. However, if no previous session is found, the proxy automatically redirects to IDP and the user is presented with the login screen. Once manually logged in, IDP redirects back to the URL that was initially called, passing through the proxy again. This time, the proxy adds an authentication header, and the user is sent to the destination URL. Here, the built-in SDK kicks in and verifies if the token signature is valid and was not altered in transit.



Tokens

Identity token – Contains information displayed in the UI (e.g. a name).

Access token – Contains functional info (claims, roles e.g. email, telephone).

Refresh token – Does not contain any information, instead is used for SSO purposes to renew the access token and extend validity.

For more information about authentication and tokens, see the FintechOS API Guide.

Scenarios

Expired access token – The user is not redirected to the login page, instead, a new token is provided.

Expired refresh token – The session is lost, and the user is redirected to the log-in page.

Dependencies

Token validity – Cannot be altered in any way and the value must be on par with the configuration in IDP.

Refresh token validity – Cannot be altered in any way and the value must be on par with the configuration in IDP

Redirect URI – Must be set up in the FintechOS platform IDP client configuration. If this field is not correctly configured, the authentication process will return an error.

Deprecated Identity Providers

> **IMPORTANT!**
> Starting with release 22.1, the FintechOS Platform uses the FintechOS Identity Provider as the default authentication layer for the FintechOS applications and services. Alternate identity providers are supported only for backward compatibility. For more information, see "FintechOS Identity Provider" on page 121.

Microsoft Active Directory Authentication

If your organization is using Microsoft Active Directory (AD) as central user repository, you can configure FintechOS Platform to give users the possibility to log in FintechOS Platform using their existing AD credentials.

FintechOS Platform supports interoperability with AD using two configurations: AD standard configuration and AD configuration using a configuration file in which you map the business units and the security roles from FintechOS Platform with the ones in AD.

To avoid unnecessary traffic across domains and return results promptly with maximum speed, you can limit the scope of Active Directory queries. For more information, see Limiting scope of the query on Active Directory.

This section covers the following topics:

AD Standard Login Configuration

In order to change the default FintechOS Platform authentication with the Microsoft Active Directory authentication, add/edit the following secret:

Key Path	Key Name	Key Value
kv/<environment>/<application>/app-settings	EBSDefaultAuthentication	AD

You are still able to log in using the administrator host credentials (using the password fromFintechOS Platform authentication).

NOTE

- When adding system users in FintechOS Platform who will be using AD credentials for logging in, in the **UserName** field, you should provide the username in the following format: **[Domain]\[Username]**. When logging in FintechOS Platform, users should provide the username in the format previously mentioned.
- Every AD has different security roles, so make sure that the Application Pool Identity of the FintechOS Platform WebApp has the privileges to search into the directory entry nodes, otherwise, when trying to log in FintechOS Platform using AD credentials, privileges related errors might occur.

(Deprecated) Add key in the **web.config** file:

Go to the web.config file of your WebApp (Portal/Designer) and add/edit the following setting:

```
<app-settings>
...
<add key="EBSDefaultAuthentication" value="AD"/>
...
</app-settings>
```

Automatically Adding Users from AD

You can automatically create / update users from Microsoft AD in FintechOS Platform using a configuration file.

> **NOTE** Automatically creating users from AD will remove the existing business units and security roles from FintechOS Platform and add the ones from AD as provided in the configuration file. If you want to keep the system user as is, you should make additional settings. For information on the additional settings, see Preserve System User's Business Unit and Security Roles.

> **IMPORTANT!** In FintechOS versions prior 18.2.8, ,getting from the Active Directory (AD) the groups to whom a user belongs to did not work smoothly; therefore, there might be situations in which wrong security roles were applied to users. With version 18.2.8, the existing configurations for mapping AD groups-roles (specified in the **~\ADUserConfiguration.xml** file) might not work as it worked in previous versions of FintechOS.

To automatically create/update users in FintechOS Platform using a configuration file, follow these steps:

1. Add the following secret in **Vault**:

Key Path	Key Name	Key Value
kv/<environment>/<application>/app-settings	EBSADAuthAutoCreateUsers	true

(Deprecated) Add key in the **web.config** files:

```
<app-settings>
...
<add key="EBSADAuthAutoCreateUsers" value="true"/>
...
</appSetting
```

2. In an xml file, create the mapping between the AD groups and the security roles and business units fromFintechOS Platform. Name the file **ADUserConfiguration.xml**. Overwrite the Business Unit from FintechOS Platform with the business unit from AD.

```
<ADUserConfiguration>
    <SecurityGroup>
          <Name>`AD Group Name`</Name>
           <DefaultBusinessUnitName>`FTOS Business Unit
Name`</DefaultBusinessUnitName>
           <SecurityRoleName>`FTOS Security Role
Name`</SecurityRoleName>
    </SecurityGroup>
</ADUserConfiguration>
```

3. In the root of the WebApp, add the **ADUserConfiguration.xml** file previously created.

Preserving System Users

To preserve the system user's business unit from FintechOS Platform, go to **Vault** and add the following:

Key Path	Key Name	Key Value
kv/<environment>/<application>/app-settings	ADOverwriteBusinessUnit	false

(Deprecated) Adding key in the **web.config** file:

```
<app-settings>
...
<add key="ADOverwriteBusinessUnit" value="false"/>
...
</app-settings>
```

To preserve the system user's security roles from FintechOS Platform and merge them with the ones provided :

Key Path	Key Name	Key Value
kv/<environment>/<application>/app-settings	ADOverwriteUserRoles	false

(Deprecated) Add key in the **web.config** file:

```
<app-settings>
...
<add key="ADOverwriteUSerRoles" value="false"/>
...
</app-settings>
```

Limiting Query Scope on AD

By default, the Lightweight Directory Access Protocol (LDAP) queries are performed on the entire Active Directory (AD).

To avoid unnecessary traffic across domains and return results promptly with maximum speed, limit the scope of active directory queries by adding the following app-settings keys in Vault:

- for queries related to users, add the key `core-setting-adauth-users-container`
- for queries related to groups, add the key `core-setting-adauth-groups-container`.

When AD authentication is enabled, the FintechOS Platform will use the values provided in the app-settings keys.

The keys are optional, if they are not provided the search will be performed on the entire directory.

Setting the users and groups containers in **Vault** secrets:

Key Path	Key Name	Key Value
kv/<environment>/<application>/app-settings	core-setting-adauth-users-container	OU=Utilizatori,DC=acme,DC=ro
kv/<environment>/<application>/app-settings	core-setting-adauth-groups-container	OU=Grupuri,DC=acme,DC=ro

In the example above, the LDAP queries will be performed against the following AD containers:

Users:

- Organizational Unit (OU): Utilizatori
- Domain Component (DC): ro

Groups:

- Organizational Unit (OU): Grupuri
- Domain Component (DC): ro

(Deprecated) Setting the users and groups containers in the **web.config** file:

```
<app-settings>
    <add key="core-setting-adauth-users-
container" value="OU=Utilizatori,DC=acme,DC=ro"/>
    <add key="core-setting-adauth-groups-
container" value="OU=Grupuri,DC=acme,DC=ro"/>
    ....
</app-settings>
```

Customizing Group Membership Checks

To comply with the querying rights set up for your configuration, you can customize how the platform checks if an Active Directory user belongs to a specific Active Directory group.

To do so, add the following secrets in **Vault**:

Key Path	Key Name	Key Value
kv/<environment>/<application>/app-settings	core-setting-adauth-group-query-mode	1 or 2

- Setting the value to **1** uses the `user.GetAuthorizationGroups()` method to retrieve all the groups the user account belongs to, then loops them to see if the target

group is among them.

- Setting the value to **2** uses the `user.IsMemberOf(group)` method to query directly if the user account is part of the target group.

Default value: 1.

(Deprecated) Add key in the web.config files

In the **web.config** file add the `core-setting-adauth-group-query-mode` key in the `<app-settings>` section:

```
<app-settings>
...
<add key="core-setting-adauth-group-query-mode" value="1 or
2"/>
...
</app-settings>
```

Azure Active Directory Authentication

> **IMPORTANT!**
> Starting with release 22.1, the FintechOS Platform uses the FintechOS Identity Provider as the default authentication layer for the FintechOS applications and services. You can configure the FintechOS Identity Provider to act as an identity broker, allowing users to log in to FintechOS applications and services using their existing Azure AD credentials. For more information, see "Using Azure AD as External Identity Provider" on page 134.
> This authentication method is provided only for backward compatibility.

If your organization is using Azure Active Directory (Azure AD) for identity and access management, you can map Azure groups to FintechOS Platform "Security Roles" on page 233 using the OpenID authentication protocol. This allows users to log in to FintechOS Platform using their existing Azure AD credentials.

Configure OpenID Settings

Configuration using **Vault** secrets:

Key Path	Key Name	Key Value
kv/<environment>/<application>/app-settings	EBSDefaultAuthentication	AzureAD

Azure openid configuration:

Key Path	Key Name	Key Value
kv/<environment>/<application>/app-settings	openid-client-id	Azure directory (tenant) id (GUID)
kv/<environment>/<application>/app-settings	openid-application-id	Azure application id (GUID)
kv/<environment>/<application>/app-settings	openid-client-secret	Azure application secret
kv/<environment>/<application>/app-settings	openid-callback-url	http://${portalRoot}/Account/LogonCallback
kv/<environment>/<application>/app-settings	openid-discovery-endpoint	https://login.microsoftonline.com/${tenantId}/.well-known/openid-configuration

User mapping settings:

kv/<environment>/<application>/app-settings	openid-auto-user-roles	Guest,Developer,Registered Users
kv/<environment>/<application>/app-settings	openid-auto-user-organization	ebs
kv/<environment>/<application>/app-settings	openid-auto-user-businessunit	root
kv/<environment>/<application>/app-settings	openid-auto-user-type	Back Office
kv/<environment>/<application>/app-settings	openid-auto-user-remote-roles-add	0|1
kv/<environment>/<application>/app-settings	openid-auto-user-remote-roles-sync	0|1

Configuration Keys

Parameter	Value
`openid-auto-user-roles`	Platform role names, separated by commas. These roles will be added automatically when the Azure AD user is mapped to a FintechOS Platform user. If the role is already defined in the FintechOS Platform, the role should be added in openid-auto-user-roles too.
`openid-auto-user-organization`	Platform organization name. The mapped user will be added in this organization.
`openid-auto-user-businessunit`	Platform business unit name. The mapped user will be added in this business unit.
`openid-auto-user-remote-roles-add`	When set to 1, the roles from Azure AD will be added to the mapped user on user creation, adding the roles found in the values for web.config key="openid-auto-user-roles" (has effect only at user creation). See below how to expose the Azure AD roles in custom claims consumable by FintechOS Platform. Azure AD will be added to the mapped user,
`openid-auto-user-remote-roles-sync`	When set to 1, the roles from Azure AD and the default roles are always synchronized at login. Any roles manually added to FintechOS Platform user are lost.

Parameters

Parameter	Value
`${portalRoot}`	Root URL for the FintechOS Platform web service.
`${tenatnId}`	Azure tenant ID.

(Deprecated) Configuration using **web.config** files:

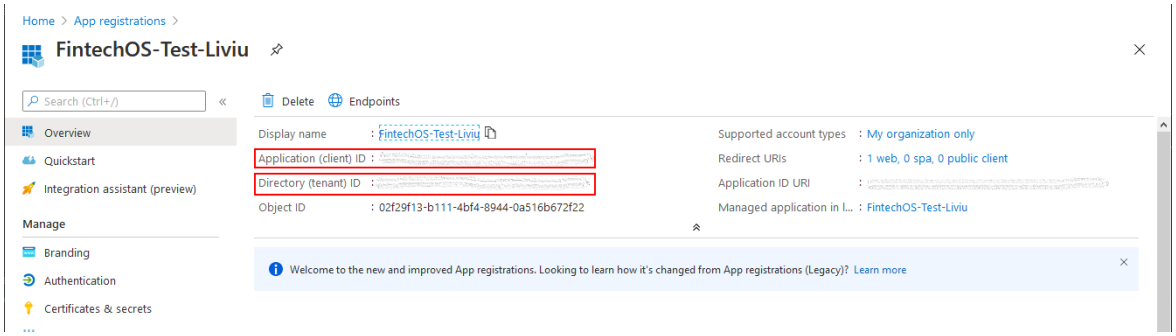
Add the following keys to the `<app-settings>` node in the *web.config* file of your web service (Portal/Studio):

```
<add key="EBSDefaultAuthentication" value="AzureAD" />
<!-- BEGIN AzureAD IDOPEN ID CONFIGURATION -->
<add key="openid-client-id" value="Azure directory (tenant) id (GUID)" />
<add key="openid-application-id" value="Azure application id (GUID)"/>
```

```
<add key="openid-client-secret" value="Azure application
secret"/>
<add key="openid-callback-
url" value="http://${portalRoot}/Account/LogonCallback" />
<add key="openid-discovery-
endpoint"
 value="https://login.microsoftonline.com/${tenantId}/.well-
known/openid-configuration" />
<!-- USER MAPPING SETTINGS -->
<add key="openid-auto-user-roles" value="Registered User,My
default role" />
<add key="openid-auto-user-organization" value="ebs" />
<add key="openid-auto-user-businessunit" value="root" />
<add key="openid-auto-user-type" value="Back Office" />
<add key="openid-auto-user-remote-roles-add" value="0 or
1"/>
<add key="openid-auto-user-remote-roles-sync" value="0 or
1"/>
<!-- END AzureAD IDOPEN ID CONFIGURATION -->
```

To find the *Azure directory (tenant) id (GUID)* and the *Azure application id (GUID)*:

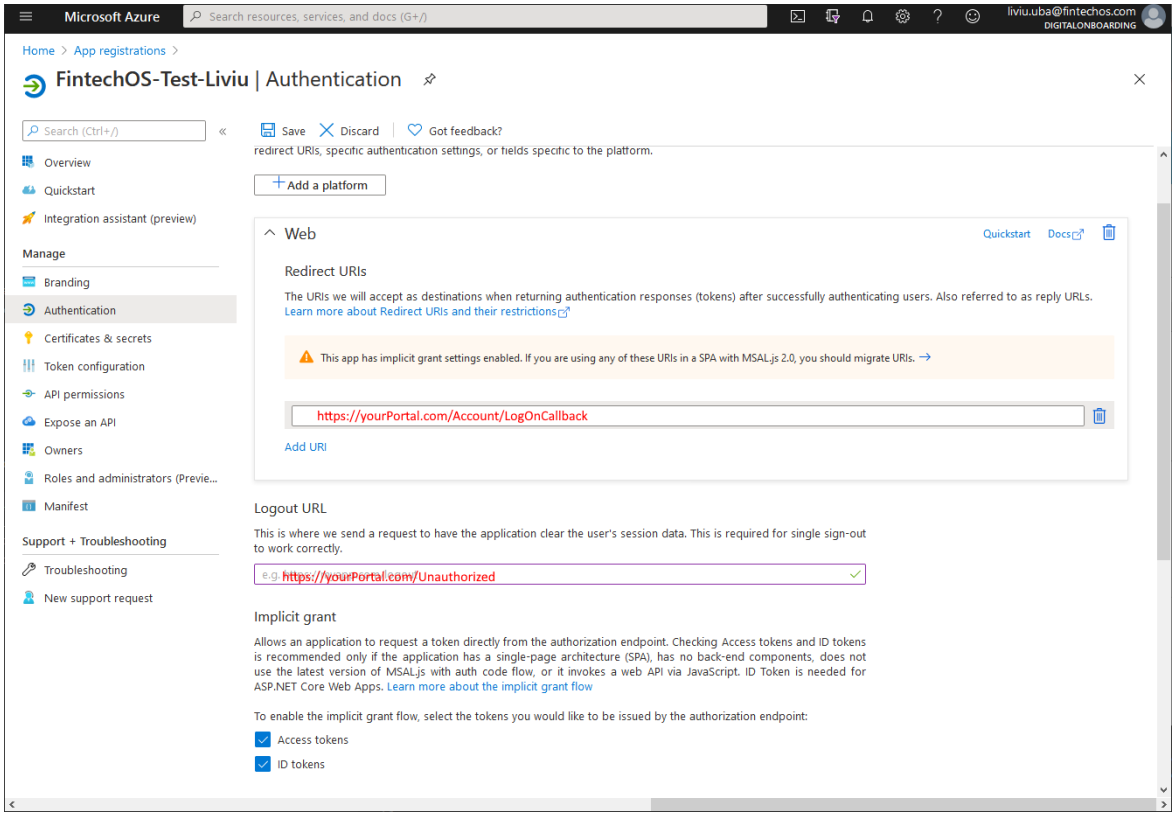
1. Open the **Azure Portal**.
2. Select the **App registrations** service.
3. Select the application you wish to use as a source for identity credentials.
4. The *Azure directory (tenant) id (GUID)* and the *Azure application id (GUID)* will be displayed in the Overview section of the application.



Set up Login/Logout Redirect URIs

In the Azure Portal, in the Authentication section of your registered application, fill in the:

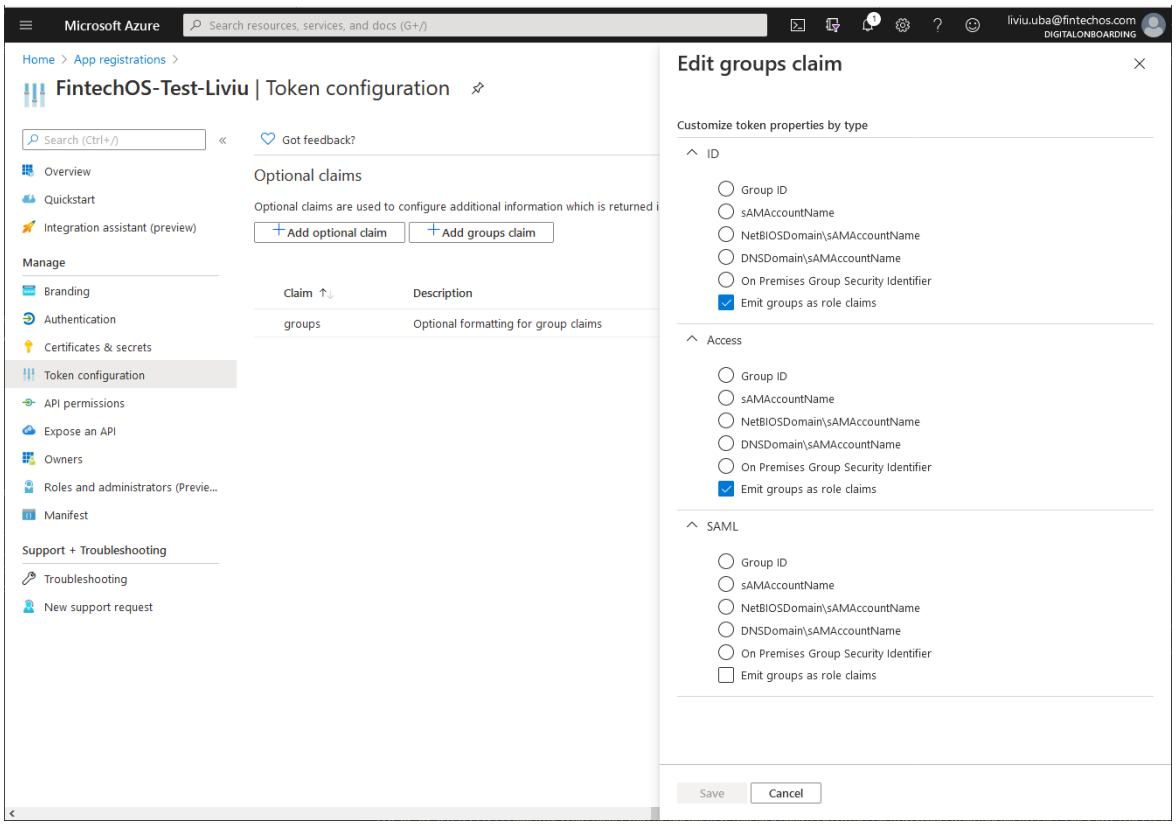
- Login redirect URI: `{$portalRoot}/Account/LogonCallback`
- Logout redirect URI: `{$poratalRoot}/Unauthorized`



Groups Mapping

When a user is authorized with Azure AD, a corresponding system user is created in FintechOS Platform. Default roles for this user, organization, business unit, and user type can be configured in *web.config*. Any Security Role which has not already been created in the system and is mentioned in OpenIdUserConfiguration.xml it will be automatically created.

To synchronize groups created in Azure AD with FintechOS Platform, the administrator of the Azure AD application must include an optional claim named **groups** in the token configuration.



To configure the mappings, an XML file named *OpenIdUserConfiguration.xml* must be placed in the root folder of the web application. Azure AD sends group IDs with the OpenID token, so the mapping must be done between the Azure Group ID and QWPlatform security roles.

```
<root>
    <SecurityGroup>
        <Name>b681734a-5601-435c-b817-465f8e20b7fb</Name>
        <DisplayName>GROUP1</DisplayName>
        <DefaultBusinessUnitName>root</DefaultBusinessUnitName>
        <SecurityRoleName>Registered Users</SecurityRoleName>
    </SecurityGroup>
        ...

    <SecurityGroup>
        <Name>84d47d54-0956-4f9a-b37d-81880374fd46</Name>
        <DisplayName>GROUP2</DisplayName>
        <DefaultBusinessUnitName>root</DefaultBusinessUnitName>
        <SecurityRoleName>Developer,Registered
Users</SecurityRoleName>
    </SecurityGroup>
```

```
</root>
```

IMPORTANT!
Any changes to `OpenIdUserConfiguration.xml` require a manual Application Domain restart.

Authentication with Okta

IMPORTANT!
Starting with release 22.1, the FintechOS Platform uses the FintechOS Identity Provider as the default authentication layer for the FintechOS applications and services. You can configure the FintechOS Identity Provider to act as an identity broker, allowing users to log in to FintechOS applications and services using their existing Okta credentials. For more information, see "Using Okta as External Identity Provider" on page 147.
This authentication method is provided only for backward compatibility.

Okta is a standards-compliant OAuth 2.0 authorization server and a certified OpenID Connect provider.

FintechOS Platform built-in integration with Okta enables users to log in to the Digital Experience Portal using the Okta single-sign on (SSO).

How to Set up the Okta Authentication

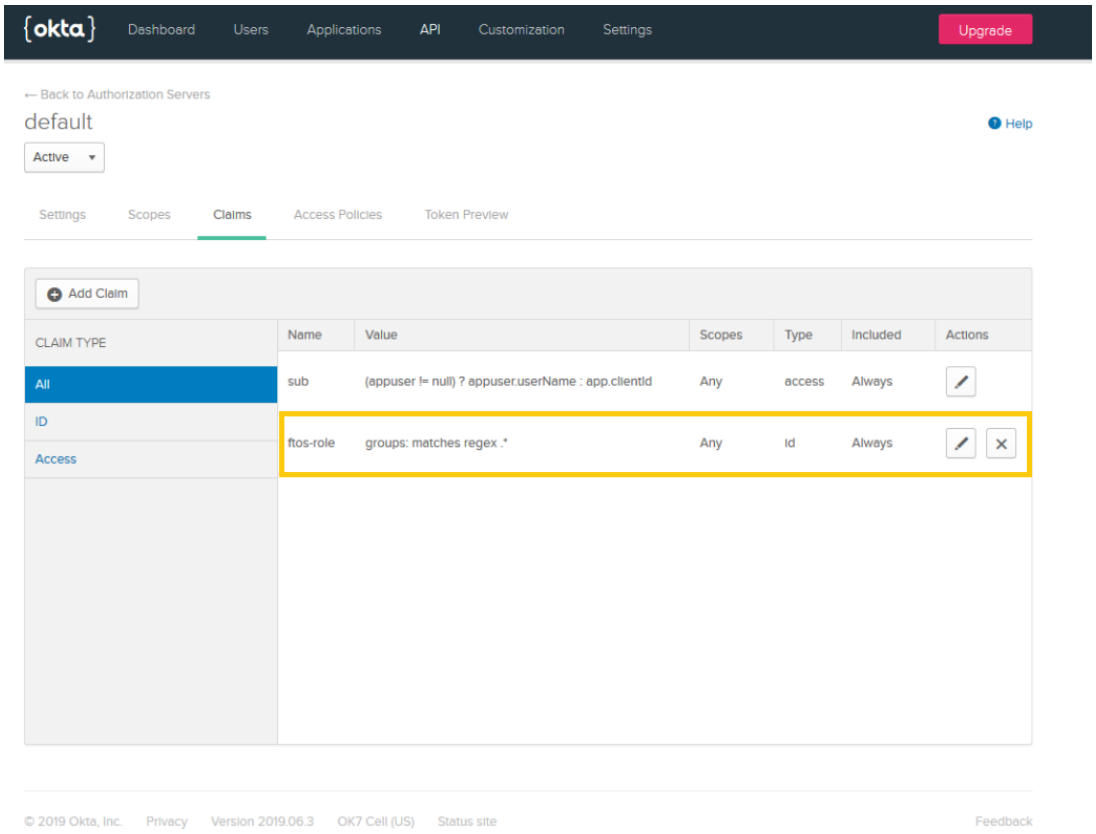
To set up the Okta authentication for your Experience Portal, follow these steps:

Step 1. Create and configure the Okta app

1. Using an Okta admin account, log into Okta and create an Okta application (Application tab > Web > OpenID Connect).
2. From the Applications tab > General > Login, set up the FintechOS Platform callbacks by configuring both the login and the logout redirect URLs, as follows:

login redirect uri	**{$portalRoot}/Account/LogonCallback**
logout redirect uri	**{$portalRoot}/Unauthorized**

3. From the **API** tab > **Authorization Servers**, create an authorization server for the Okta application.
4. Expose the Okta roles in custom claims consumable by FintechOS Platform. To do so, synchronize the user groups created in Okta with FintechOS Platform by creating a custom claim named **ftos-role** mapped to the group metadata in Okta. For more information on how to create a custom claim in the Okta app, see Okta Documentation.



When a user is authorized with Okta, a corresponding system user will be created in FintechOS Platform . In the **web.config** file you can configure default roles for this user, organization, business unit and user type.

Step 2. Configure the Experience Portal

Prerequisite:

Make sure that you know the following values:

- Client ID (from the Okta app, General tab)
- Client Secret (from the Okta app, General tab)
- Discovery Endpoint (from the Okta app, API section > Authorization Servers > Metadata URL)

Configuration using **Vault** secrets:

Set Okta authentication:

Key Path	Key Name	Key Value
kv/<environment>/<application>/app-settings	EBSDefaultAuthentication	Okta

Replace the keys' value with your Okta configuration:

Key Path	Key Name	Key Value
kv/<environment>/<application>/app-settings	openid-client-id	{ClientId}
kv/<environment>/<application>/app-settings	openid-client-secret	{ClientSecret}
kv/<environment>/<application>/app-settings	openid-callback-url	http://${portalRoot}/Account/LogonCallback
kv/<environment>/<application>/app-settings	openid-discovery-endpoint	https://${oktaApplication}.okta.com/oauth2/${authServerId}/.well-known/oauth-authorization-server

User mapping settings:

Key Path	Key Name	Key Value
kv/<environment>/<application>/app-settings	openid-auto-user-roles	Guest,Developer,Registered Users
kv/<environment>/<application>/app-settings	openid-auto-user-organization	ebs

Key Path	Key Name	Key Value
kv/<environment>/<application>/app-settings	openid-auto-user-businessunit	root
kv/<environment>/<application>/app-settings	openid-auto-user-type	Back Office
kv/<environment>/<application>/app-settings	openid-auto-user-remote-roles-add	0|1
kv/<environment>/<application>/app-settings	openid-auto-user-remote-roles-sync	0|1

The table below describes the Okta app configuration keys:

Key	Description
${portalRoot}	The root URL of the Experience Portal.
${authServerId}	The ID of the authorization server associated with the Okta application (default value is default).
${oktaApplication}	The ID of the Okta application.
Key	Description

The table below describes the user mapping configuration keys.

Parameter	Value
openid-auto-user-roles	The platform role names, separated by colon. These roles will be added automatically when the Okta user is mapped to a platform user.
openid-auto-user-organization	The platform organization name. The mapped user will be added in this organization.
openid-auto-user-businessunit	The platform business unit name. The mapped user will be added in this business unit.
openid-auto-user-remote-roles-add	If set to 1, the roles from the Okta app will be added to the mapped user.
openid-auto-user-remote-roles-sync	If value is 1, the roles from Okta and the default roles are always synchronized at login. Any roles manually added to a Okta user are lost.

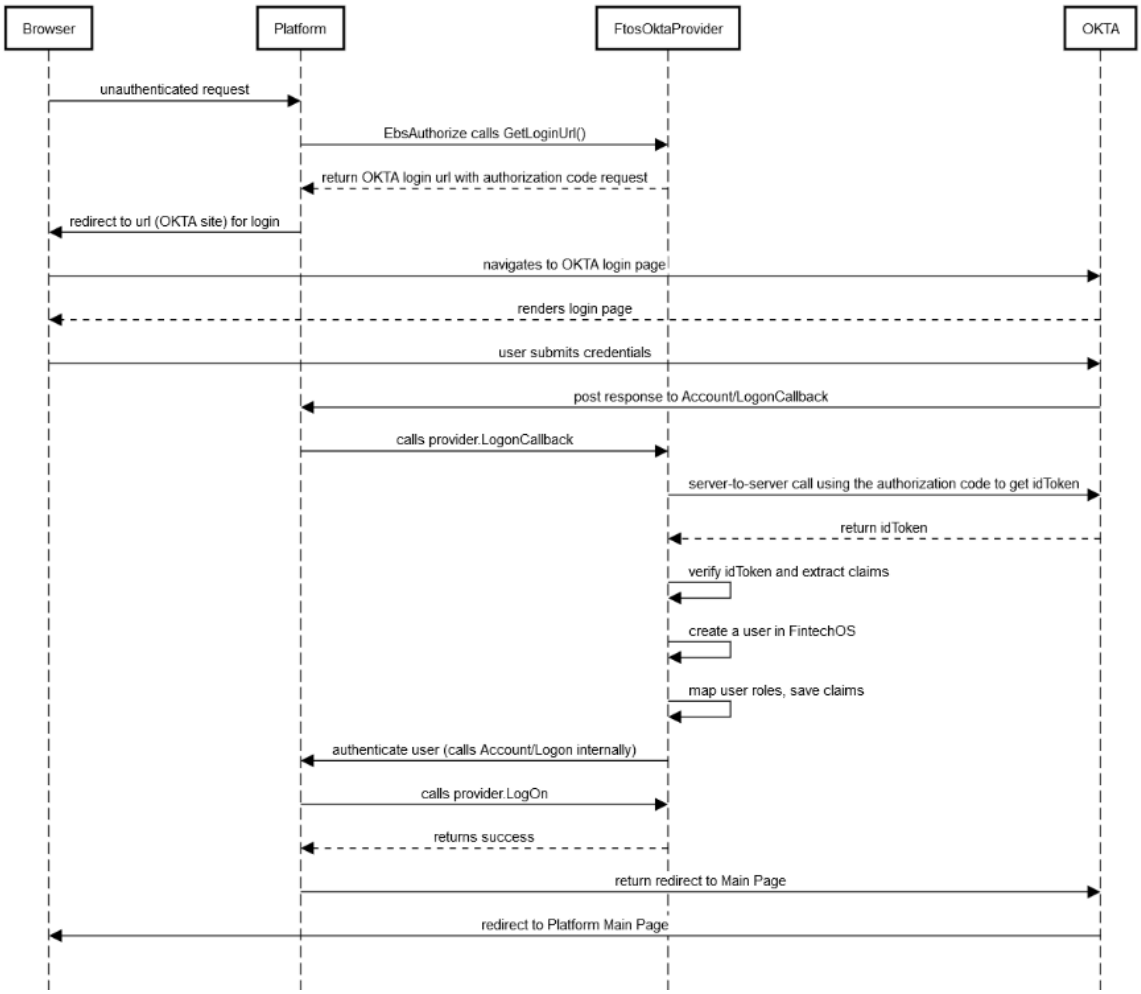
(Deprecated) Configuration using **web.config** files:

Go to the <app-settings> section and add the configuration of your Okta appplication:

```
<!-- 1. Set Okta authentication-->
<add key="EBSDefaultAuthentication" value="Okta" />

<!-- 2. Replace these values with your Okta configuration: -
->
<add key="openid-client-id" value="{ClientId}" />
<add key="openid-client-secret" value="{ClientSecret}" />
<add key="openid-callback-
url" value="http://${portalRoot}/Account/LogonCallback" />
<add key="openid-discovery-endpoint" value=
"https://${oktaApplication}.okta.com/oauth2/${authServerId}/
.well-known/oauth-authorization-server" />

<!-- 3. Map user settings: -->
<add key="openid-auto-user-
roles" value="Guest,Developer,Registered Users" />
<add key="openid-auto-user-organization" value="ebs" />
<add key="openid-auto-user-businessunit" value="root" />
<add key="openid-auto-user-type" value="Back Office" />

<add key="openid-auto-user-remote-roles-add" value="0|1"/>
<add key="openid-auto-user-remote-roles-sync" value="0|1"/>
```

How it Works

The diagram below describes the FintechOS Platform login flow when using Okta authentication.



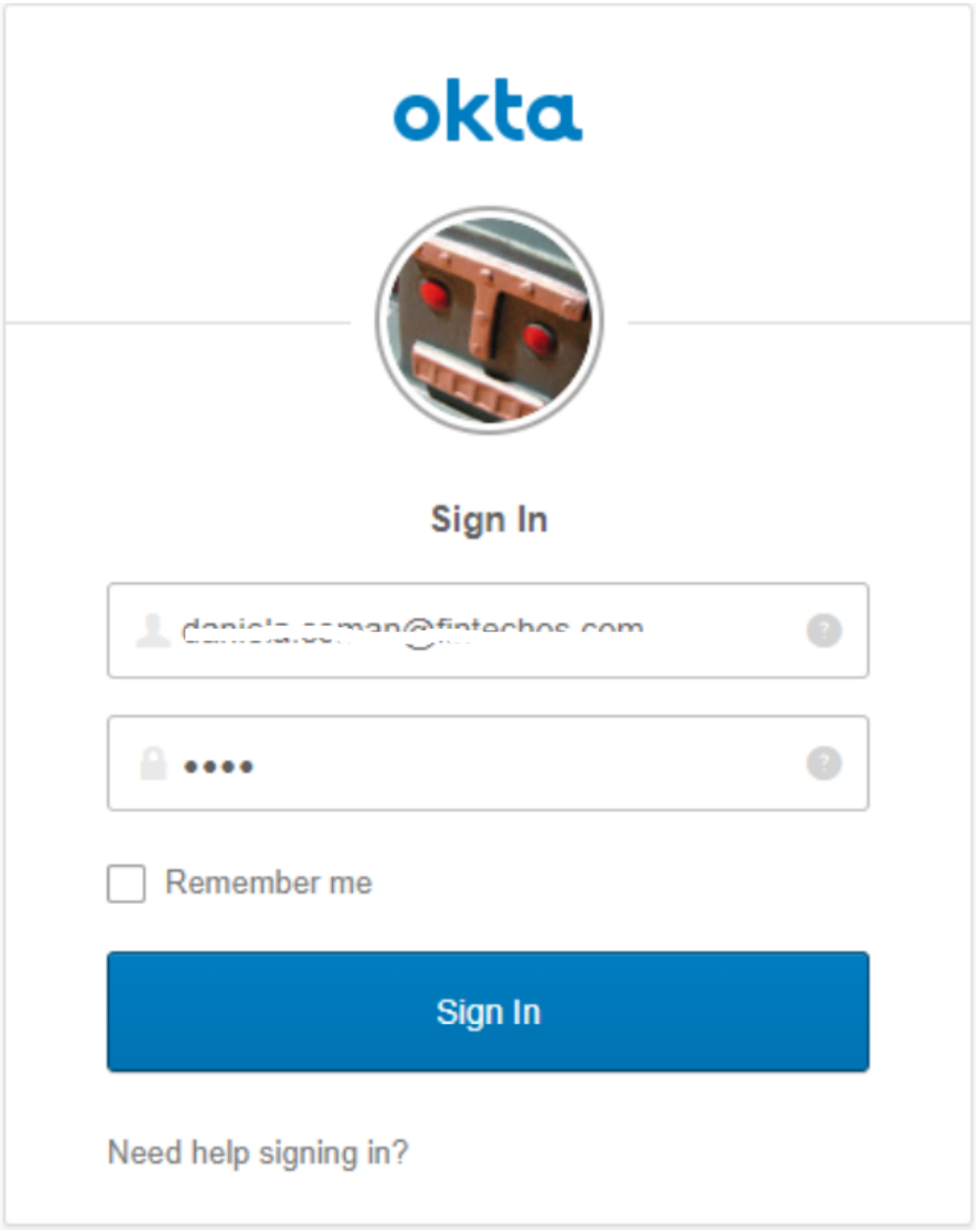
Group mapping in FintechOS

When a user is authorized with Okta, a corresponding system user is created in FintechOS. In web.config file of the FintechOS instance, default roles for this user, organization, business unit and user type are added.

Create a custom claim named **ftos-role** mapped to the group metadata in Okta. This configuration is done for the authorization server associated with the Okta application.

How users log in the Portal

When accessing the Digital Experience Portal URL, users will be redirected to the URL of the authorization server associated with the Okta app. The Okta login page appears.



Once they provide Okta account credentials, they will be logged into the Digital Experience Portal.

When new users are created, they will receive an email notification from Okta which contains instructions and Okta credentials.

Troubleshooting Okta Redirect Error

Error

UnhandledException: System.Web.HttpException (0x80004005): Server cannot set status after HTTP headers have been sent.

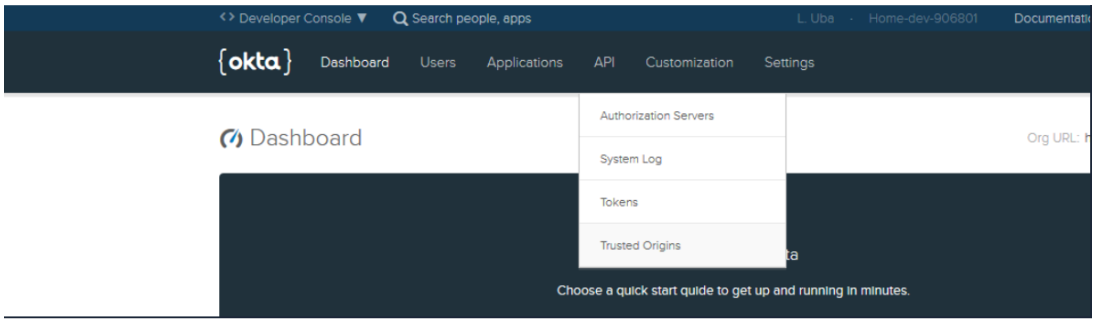
Cause

FTOS OpenID provider does not redirect when the user is still logged in due to the OpenID cookie not being expired too.

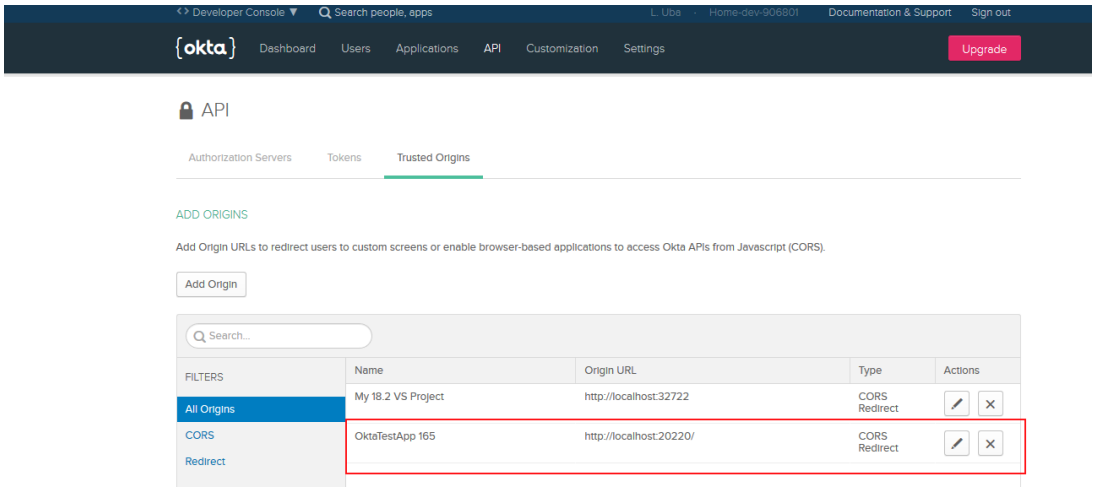
What should I do?

For session expiration to work seamlessly, follow these steps:

1. Using an Okta admin account, log into Okta.
2. Click API tab > Trusted Origins.



3. Allow CORS and Redirect for FTOS portal host.



Authentication with Active Directory Federation Services

This service provided by Microsoft manages the user sign-in information for members of a platform. If your organization is using ADFS for identity and access management of your users, it is possible to map the users already existing in ADFS to FintechOS Security Roles. When a user is authorized with ADFS, a corresponding system user is created in FintechOS. Through ADFS OpenId, users to log in to FintechOS using their existing ADFS credentials.

Add keys to Vault secrets

Key Path	Key Name	Key Value
kv/<environment>/<application>/app-settings	EBSDefaultAuthentication	ADFS

ADFS configuration:

Key Path	Key Name	Key Value
kv/<environment>/<application>/app-settings	openid-client-id	Client identifier configured in ADFS
kv/<environment>/<application>/app-settings	openid-application-id	this value is not used
kv/<environment>/<application>/app-settings	openid-client-secret	ADFS Web API shared secret
kv/<environment>/<application>/app-settings	openid-callback-url	http://{portalRoot}/Account/LogonCallback
kv/<environment>/<application>/app-settings	openid-discovery-endpoint	{adfs server uri}/adfs/.well-known/openid-configuration

User mapping settings:

Key Path	Key Name	Key Value
kv/<environment>/<application>/app-settings	openid-auto-user-roles	Registered User,My default role
kv/<environment>/<application>/app-settings	openid-auto-user-organization	ebs
kv/<environment>/<application>/app-settings	openid-auto-user-businessunit	root

Key Path	Key Name	Key Value
kv/<environment>/<application>/app-settings	openid-auto-user-type	Back Office
kv/<environment>/<application>/app-settings	openid-auto-user-remote-roles-add	0|1
kv/<environment>/<application>/app-settings	openid-auto-user-remote-roles-sync	0|1

Configuration Keys:

Key	Value
openid-auto-user-roles	Platform role names, separated by colon. These roles will be added automatically when the AD user is mapped to a platfrom user
openid-auto-user-organization	Platform organization name. The mapped user will be added in this organization
openid-auto-user-businessunit	Platform business unit name. The mapped user will be added in this business unit
openid-auto-user-remote-roles-add	when value is 1 the roles from AD will be added to the mapped user on user creation. See below how to expose the AD roles in custom claims consumable by FintechOS
openid-auto-user-remote-roles-sync	when value is 1 the roles from AD and the default roles are always synchronized at login. Any roles manually added to a AD user are lost

Parameters:

Parameter	Value
{portalRoot}	root url for FintechOS portal
{adfs server url}	ADFS server url

(Deprecated) Add keys to the web.config file

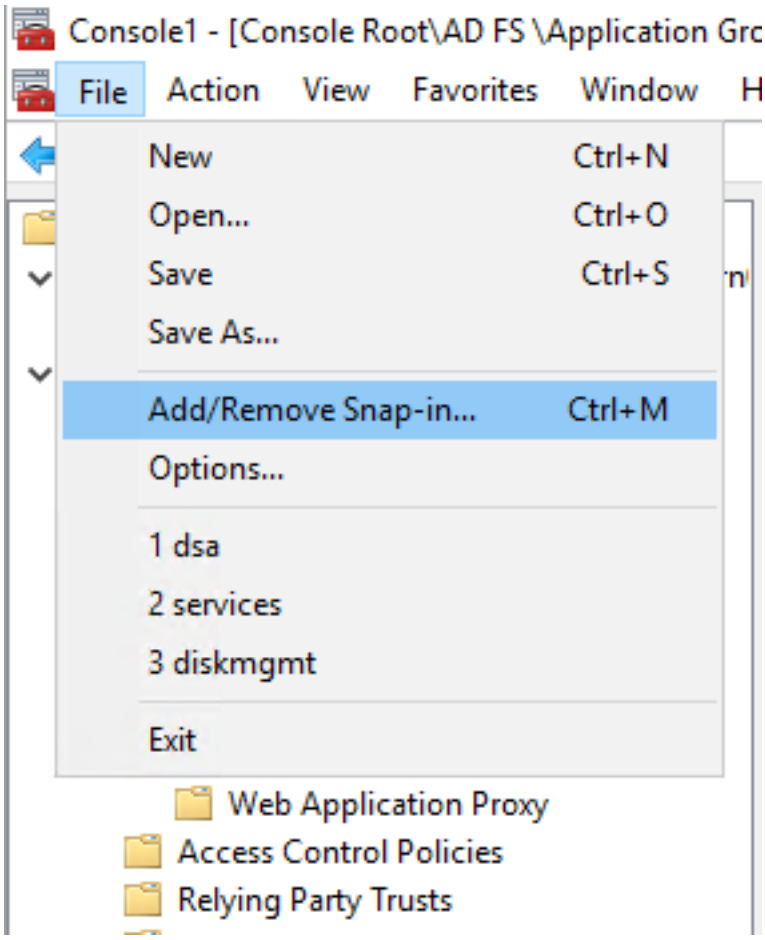
```
<add key="EBSDefaultAuthentication" value="ADFS" />

    <!-- BEGIN ADFS OPENID CONFIGURATION -->

    <add key="openid-client-id" value="Client identifier
configured in ADFS" />

    <add key="openid-application-id" value=" this value is
not used "/>
```

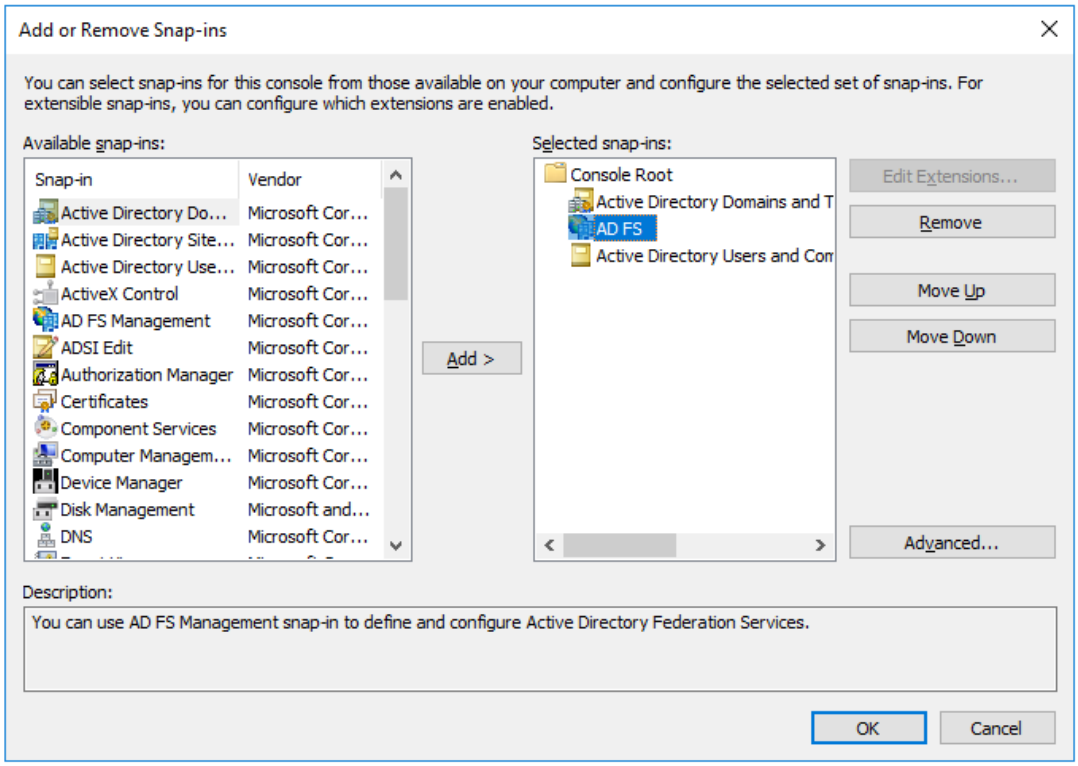
```
    <add key="openid-client-secret" value="ADFS Web API
shared secret"/>

    <add key="openid-callback-url" value="http://
{portalRoot}/Account/LogonCallback" />

    <add key="openid-discovery-endpoint" value="{adfs server
uri}/adfs/.well-known/openid-configuration" />

    <!-- USER MAPPING SETTINGS -->

    <add key="openid-auto-user-roles" value="Registered
User,My default role" />
    <add key="openid-auto-user-organization" value="ebs" />
    <add key="openid-auto-user-businessunit" value="root" />
    <add key="openid-auto-user-type" value="Back Office" />

    <add key="openid-auto-user-remote-roles-
add" value="0|1"/>
    <add key="openid-auto-user-remote-roles-
sync" value="0|1"/>

    <!-- END ADFS OPENID CONFIGURATION -->
```

ADFS configuration

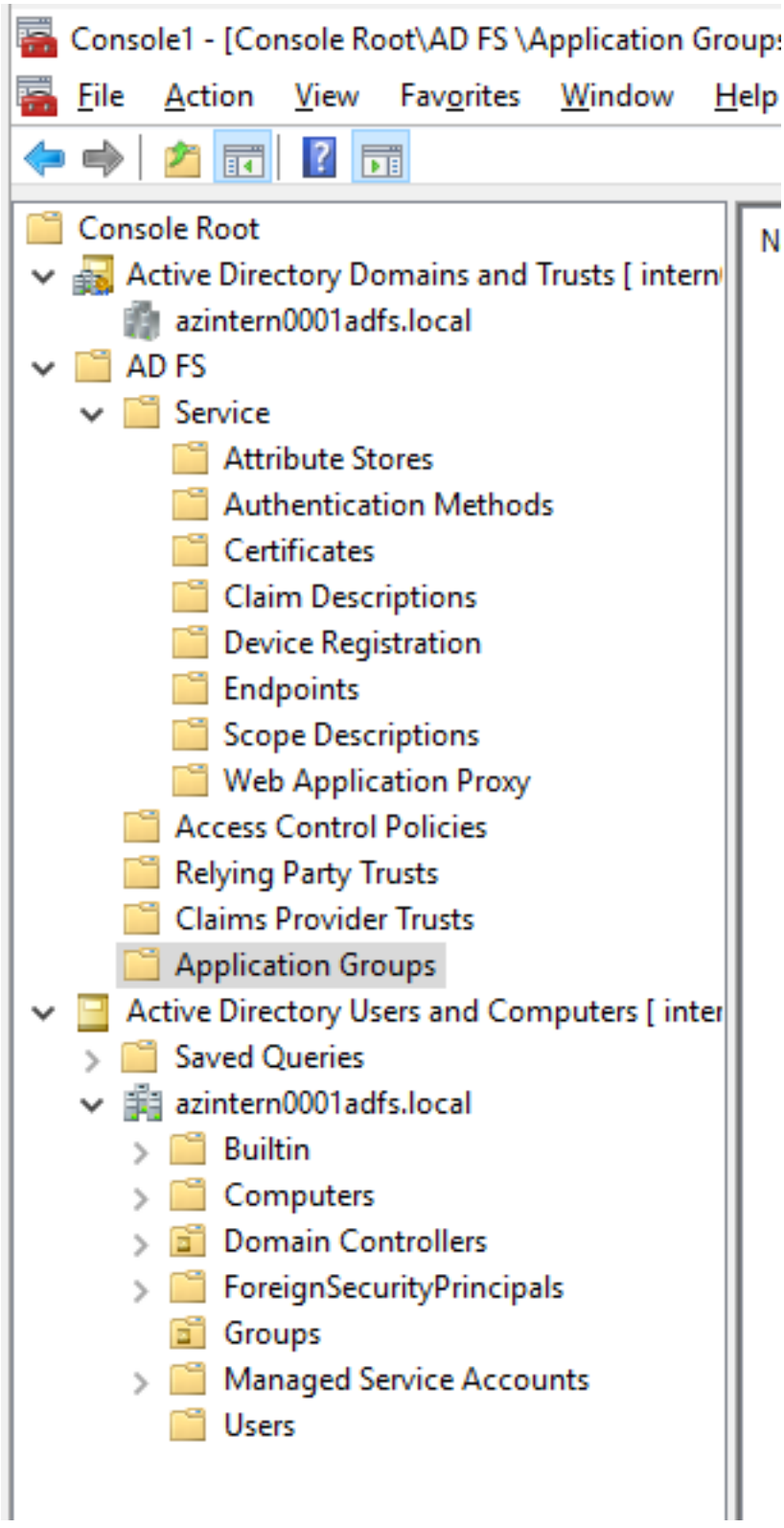
1. On a Windows Server 2016+, on the ADFS server open the Microsoft Management Console (mmc).

2. Add the ADFS snap in if not already added.

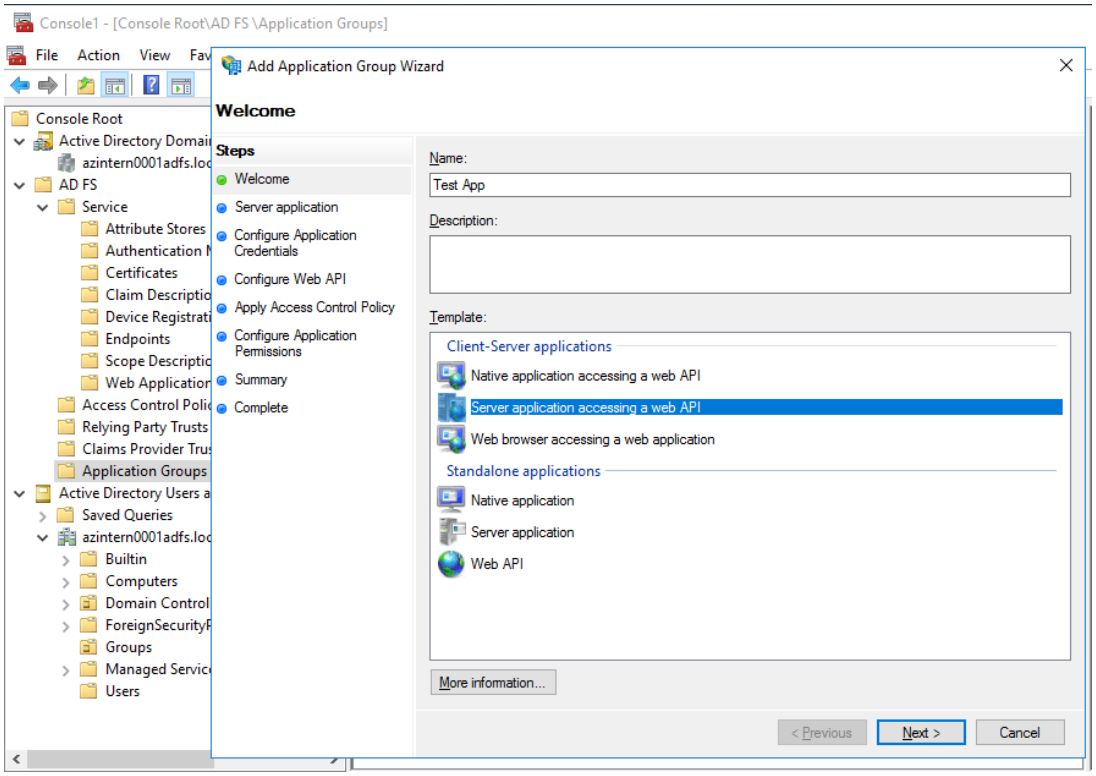




3. Open the ADFS MMC plugin and select the node Application Groups.

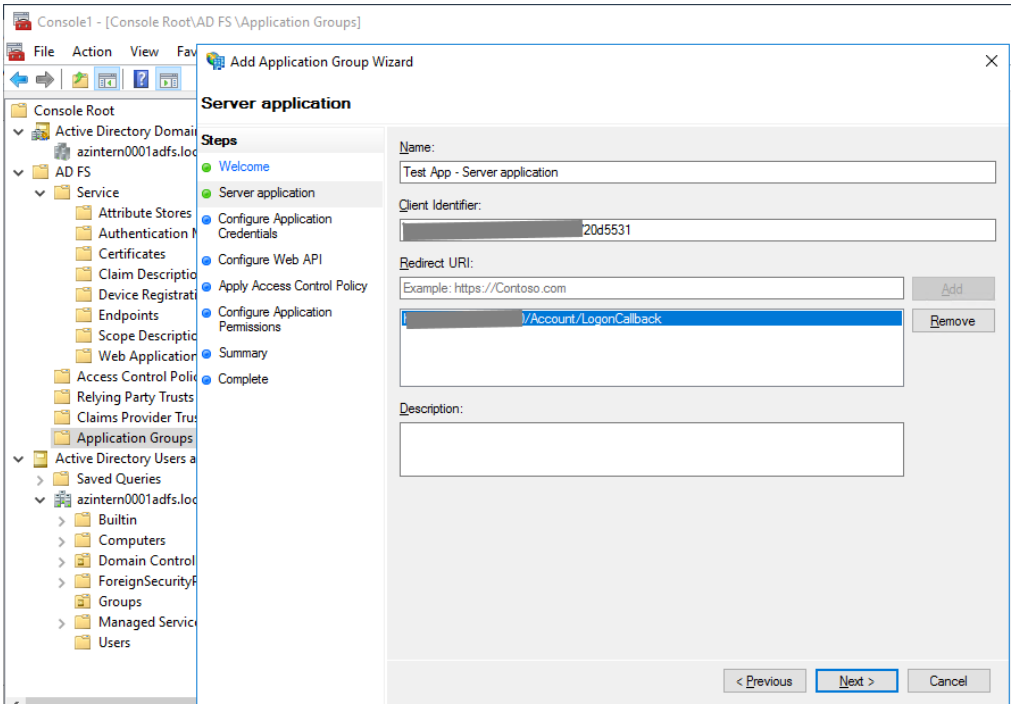

Console1 - [Console Root\AD FS \Application Groups
File Action View Favorites Window Help
Console Root
Active Directory Domains and Trusts [intern
azintern0001adfs.local
AD FS
Service
Attribute Stores
Authentication Methods
Certificates
Claim Descriptions
Device Registration
Endpoints
Scope Descriptions
Web Application Proxy
Access Control Policies
Relying Party Trusts
Claims Provider Trusts
Application Groups
Active Directory Users and Computers [inter
Saved Queries
azintern0001adfs.local
Builtin
Computers
Domain Controllers
ForeignSecurityPrincipals
Groups
Managed Service Accounts
Users

4. Right click and select Add application group. In the template list select Server application accessing a web API.

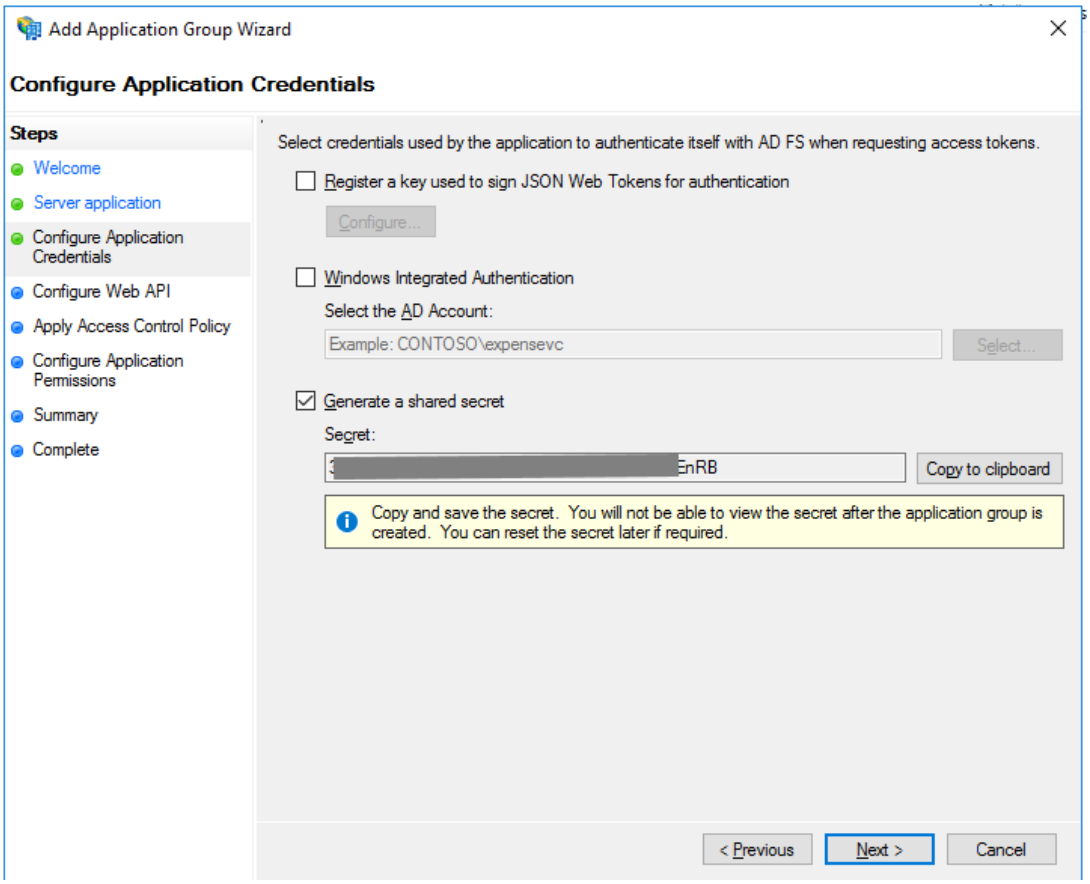


- Configure the client identifier and the redirect (callback) Url.
- Client identifier should be an global unique identifier. This value must be set in the openid-client-id configuration item in FintechOS.
- Redirect (callback) Url must be also be set in the openid-callback-url

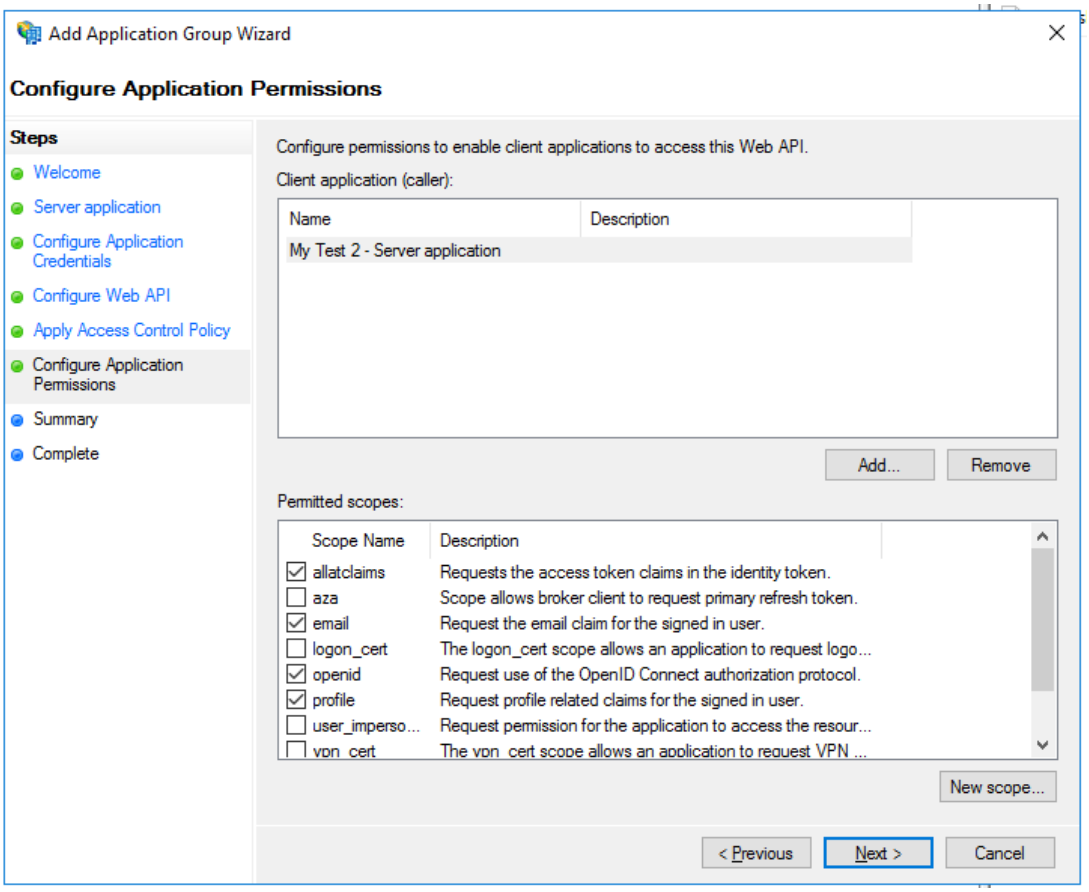
configuration item in FintechOS.



5. Configure the shared secret. The shared secret must be set also in the openid-client-secret configuration item in FintechOS.


Add Application Group Wizard
Configure Application Credentials
Steps
Welcome
Server application
Configure Application Credentials
Configure Web API
Apply Access Control Policy
Configure Application Permissions
Summary
Complete
Select credentials used by the application to authenticate itself with AD FS when requesting access tokens.
Register a key used to sign JSON Web Tokens for authentication
Configure...
Windows Integrated Authentication
Select the AD Account:
Example: CONTOSO\expensevc
Select...
Generate a shared secret
Secret:
[redacted]EnRB
Copy to clipboard
Copy and save the secret. You will not be able to view the secret after the application group is created. You can reset the secret later if required.
< Previous
Next >
Cancel

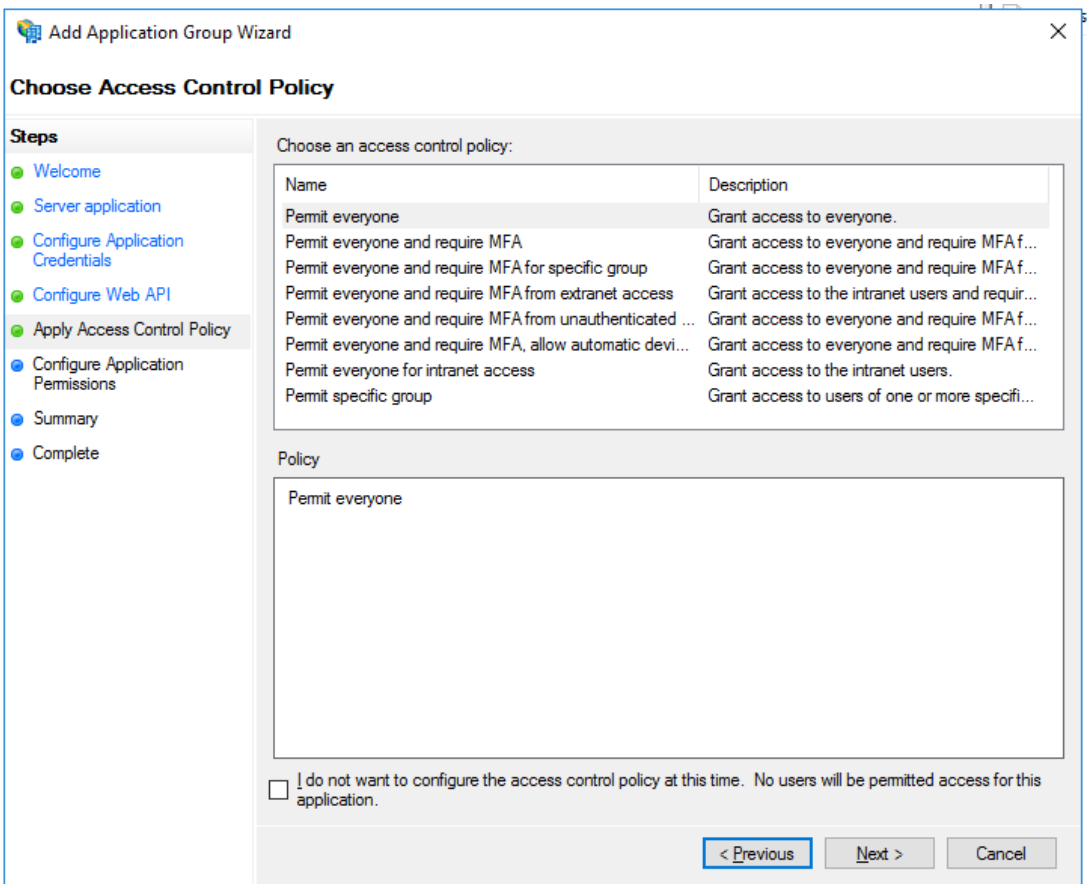
6. Configure the Web API identifier



IMPORTANT!

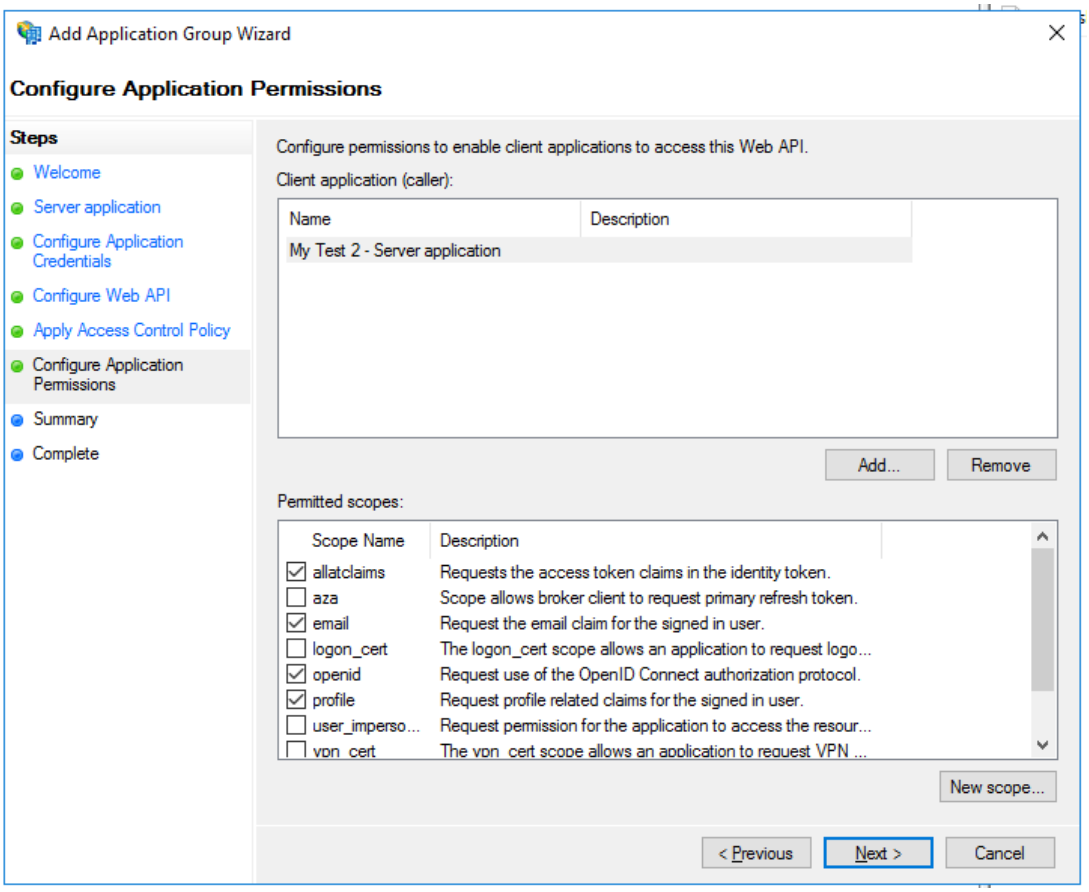
The Web API identifier must be THE SAME identifier as the one used for the CLIENT IDENTIFIER in the first step.

7. Configure Access Control Policy.



8. Configure claims to be sent with the openid token.

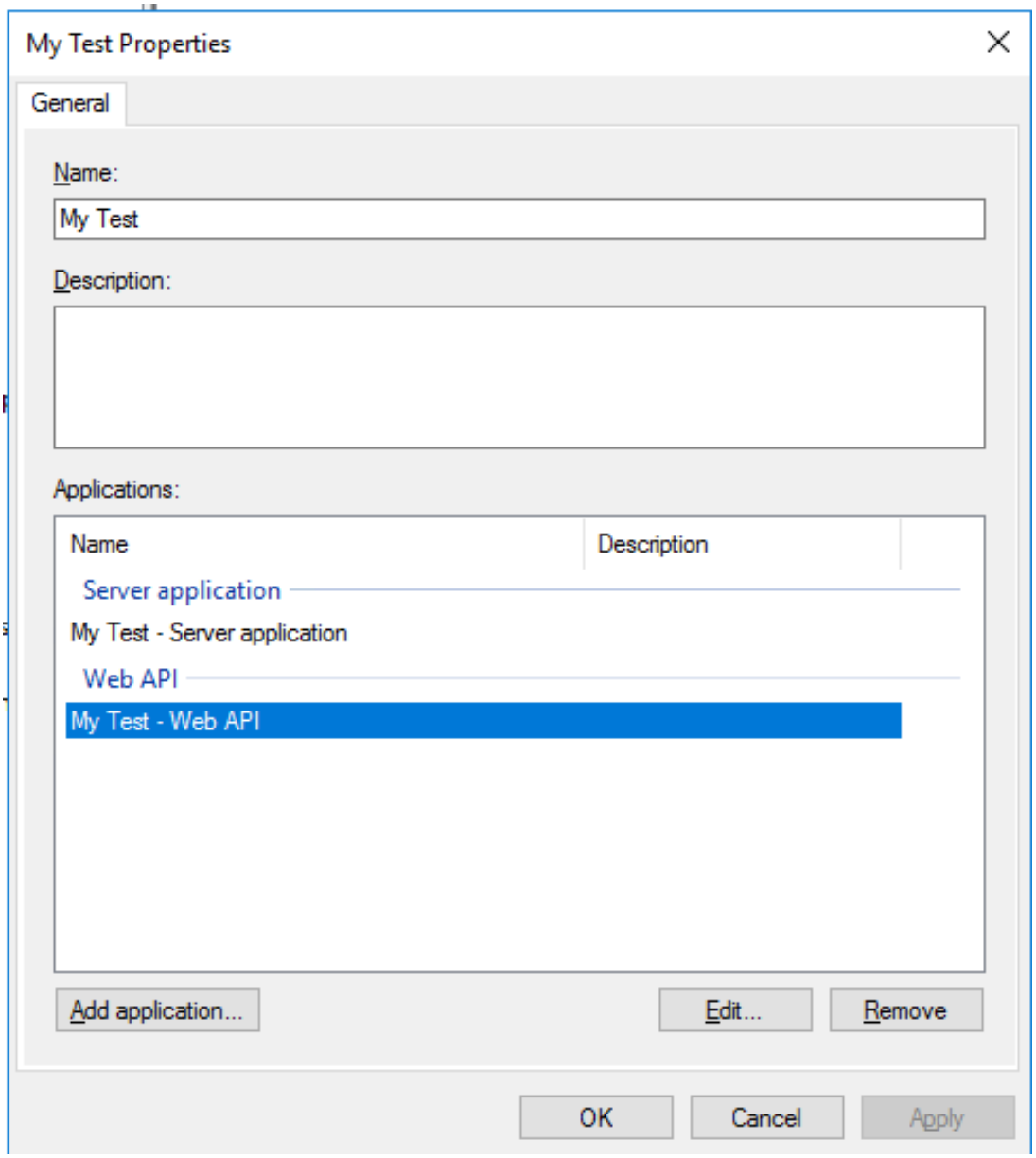
9. Following claims must be included: allatclaims, email, openid, profile.



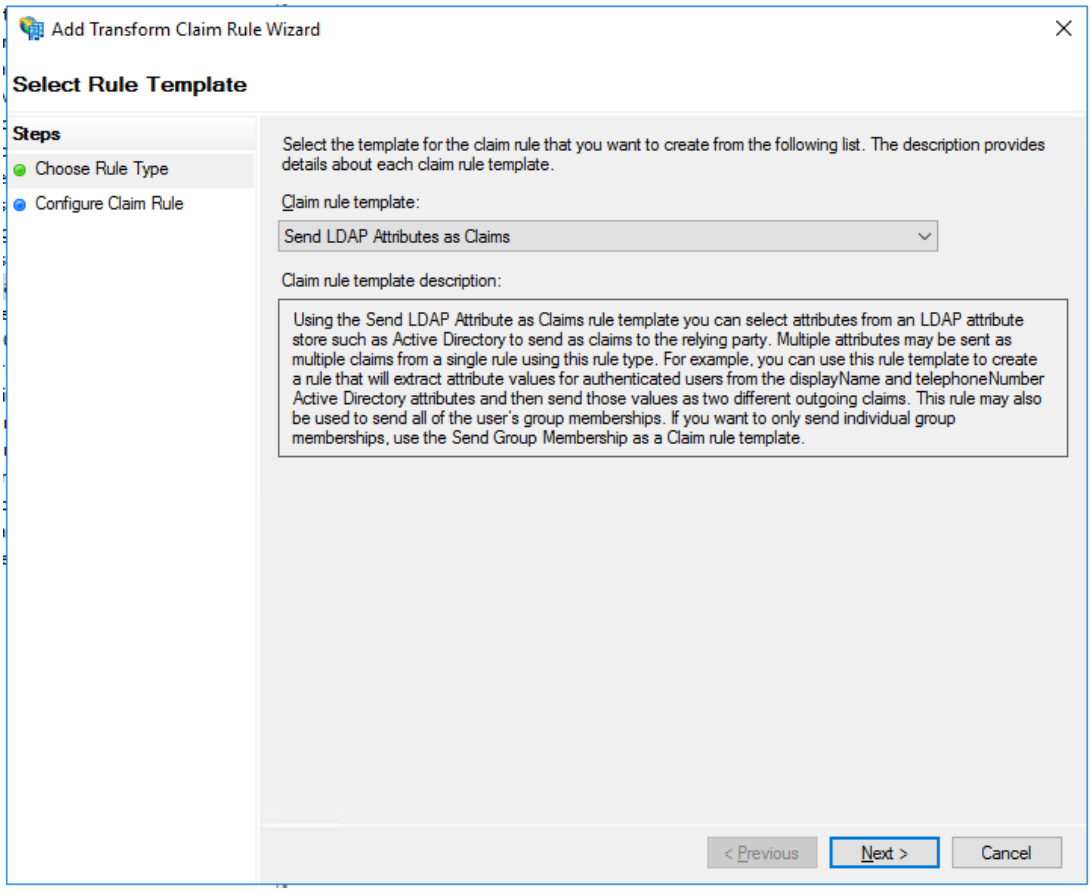
10. Review the configuration in the Summary step and go to Complete step.

> **IMPORTANT!**
>
> In the following steps we need to expose the GROUP INFORMATION, EMAIL, GIVEN NAME and SURNAME information from AD directory to be included in the claims. This will permit the correct mapping of the users to FintechOS.

11. Double click the newly created Application Group.

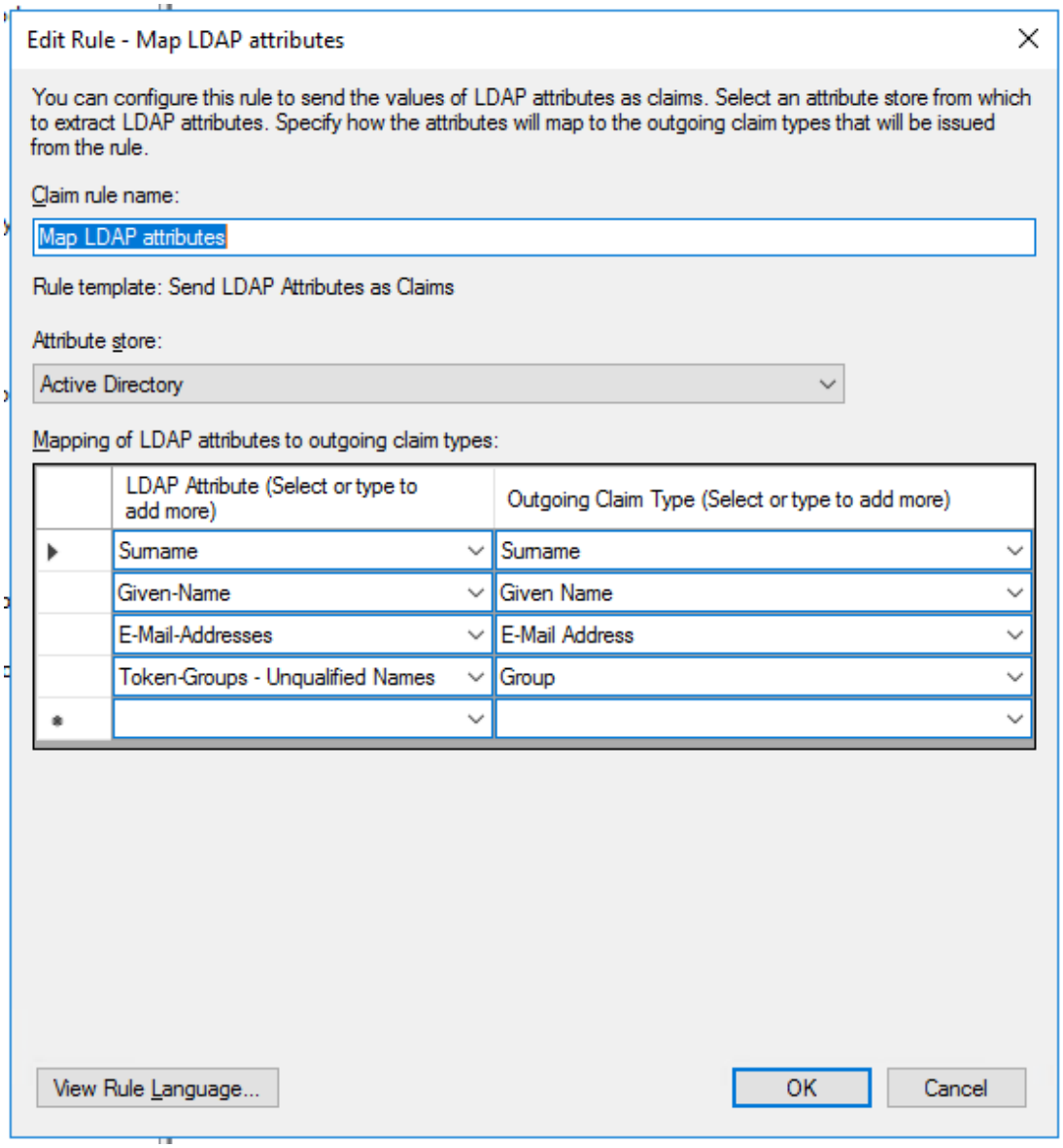


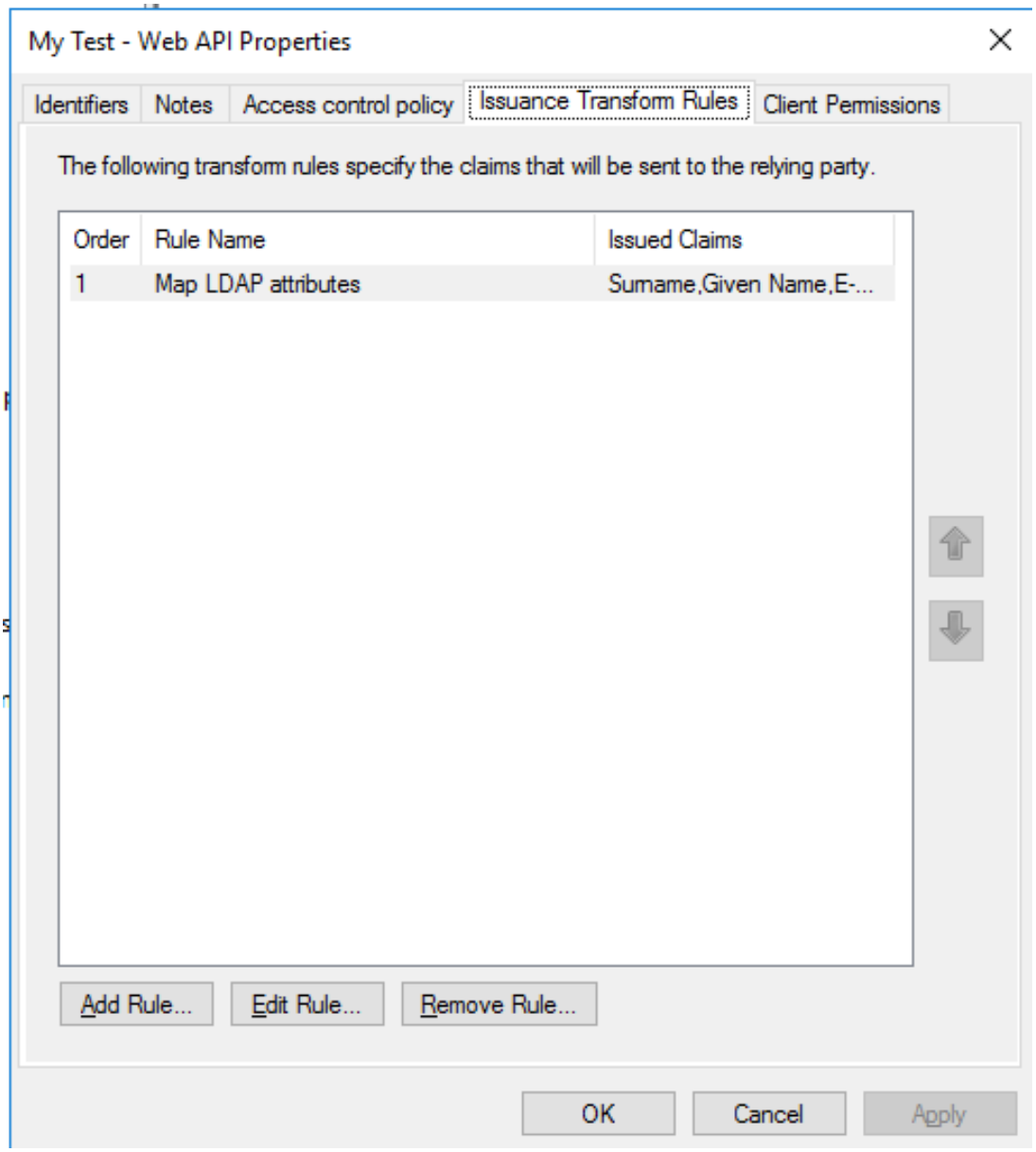
12. Select the Web API element and click the Edit... button.



13. Go to tab Issuance Transform Rules and add a new rule of type Send LDAP Attribues in Claims

14. Map the AD attributes as in the image below:





Group mapping in FintechOS

Once the system user has been created in the FintechOS Studio, it is possible to have default roles for this user, organization, business unit and user type configured in web.config.

To configure the mappings, an XML file named OpenIdUserConfiguration.xml must be placed in the root of the web application. When the ADFS configuration was performed as in the section above, the ADFS token for an user authentication will include a group claim with the names of the Groups where the user is member from AD.

Authentication with AWS Cognito

> **IMPORTANT!**
> Starting with release 22.1, the FintechOS Platform uses the FintechOS Identity Provider as the default authentication layer for the FintechOS applications and services. You can configure the FintechOS Identity Provider to act as an identity broker, allowing users to log in to FintechOS applications and services using their existing AWS Cognito credentials. For more information, see "Using AWS Cognito as External Identity Provider" on page 153.
> This authentication method is provided only for backward compatibility.

This service provided by Amazon Web Services manages the user sign-in information for members of a platform. If your organization is using AWS Cognito for identity and access management of your users, it is possible to map the users already existing in AWS Cognito to FintechOS Security Roles. Through Azure AWS OpenId provider, users to log in to FintechOS using their existing AWS Cognito credentials.

Add keys to Vault secrets

Set AWS Cognito authentication:

Key Path	Key Name	Key Value
kv/<environment>/<application>/app-settings	EBSDefaultAuthentication	AWSCognito

AWS Cognito configuration:

Key Path	Key Name	Key Value
kv/<environment>/<application>/app-settings	openid-client-id	AWS Cognito client id xxxxx
kv/<environment>/<application>/app-settings	openid-client-secret	AWS Cognito client secret yyyyyy

Key Path	Key Name	Key Value
kv/<environment>/<application>/app-settings	openid-callback-url	http://${portalRoot}/Account/LogonCallback
kv/<environment>/<application>/app-settings	openid-discovery-endpoint	https://cognito-idp.xxx/.well-known/openid-configuration

User mapping settings:

Key Path	Key Name	Key Value
kv/<environment>/<application>/app-settings	openid-auto-user-roles	Guest,Developer,Registered Users
kv/<environment>/<application>/app-settings	openid-auto-user-organization	ebs
kv/<environment>/<application>/app-settings	openid-auto-user-businessunit	root
kv/<environment>/<application>/app-settings	openid-auto-user-type	Back Office
kv/<environment>/<application>/app-settings	openid-auto-user-remote-roles-add	0|1
kv/<environment>/<application>/app-settings	openid-auto-user-remote-roles-sync	0|1

Configuration Keys:

Key	Value
openid-auto-user-roles	Platform role names, separated by colon. These roles will be added automatically when the AWS Cognito user is mapped to a platfrom user
openid-auto-user-organization	Platform organization name. The mapped user will be added in this organization
openid-auto-user-businessunit	Platform business unit name. The mapped user will be added in this business unit
openid-auto-user-remote-roles-add	not supported yet

Key	Value
openid-auto-user-remote-roles-sync	not supported yet

Parameters:

Parameter	Value
${portalRoot}	root url for FintechOS portal

(Deprecated) Add keys to the web.config file

In the web.config file of your environment add the following keys.

```
 <add key="EBSDefaultAuthentication" value="AWSCognito" />

    <!-- BEGIN AWS COGNITO IDOPEN ID CONFIGURATION -->

    <add key="openid-client-id" value="AWS Cognito client id
xxxxx" />
    <add key="openid-client-secret" value="AWS Cognito
client secret yyyyyy" />

    <add key="openid-callback-
url" value="http://${portalRoot}/Account/LogonCallback" />

    <add key="openid-discovery-
endpoint" value="https://cognito-idp.xxx/.well-known/openid-
configuration" />

    <!-- USER MAPPING SETTINGS -->

    <add key="openid-auto-user-
roles" value="Guest,Developer,Registered Users" />
    <add key="openid-auto-user-organization" value="ebs" />
    <add key="openid-auto-user-businessunit" value="root" />
    <add key="openid-auto-user-type" value="Back Office" />

    <add key="openid-auto-user-remote-roles-add" value="0"/>
    <add key="openid-auto-user-remote-roles-
sync" value="0"/>

    <!-- END AWS COGNITO ID CONFIGURATION -->
```

Group mapping for users

For each user in FintechOS, default roles can be created in the web.config file for this user, organization, business unit and user type.

1. An XML file named **OpenIdUserConfiguration.xml** must be placed in the root of the web application of FintechOS.

   > **IMPORTANT!**
   >
   > Any changes to OpenIdUserConfiguration.xml require a manual Application Domain restart.

2. The ADFS token for an user authentication will include a group claim with the names of the Groups where the user is member from AD.

Example:

```
<root>
    <SecurityGroup>
        <Name>GROUP1</Name>

<DefaultBusinessUnitName>root</DefaultBusinessUnitName>
        <SecurityRoleName>Registered
Users,Developers</SecurityRoleName>
    </SecurityGroup>
    ...

    <SecurityGroup>
        <Name>GROUP2</Name>

<DefaultBusinessUnitName>root</DefaultBusinessUnitName>
        <SecurityRoleName>GROUP2</SecurityRoleName>
    </SecurityGroup>
</root>
```

Browser Based Multi-Factor Authentication

Multi-Factor Authentication (MFA) adds an extra layer of security on top of the basic authentication methods. It requires users to provide multiple proof of their claimed identity prior to being granted access according to their security roles and permissions.

User access can be granted based on two cumulative conditions:

- Something the user knows (login credentials): username and password.
- Something the user controls (a mobile device or email account used to receive a temporary pass code via SMS/E-mail).

When users access the app, they will be prompted to provide the login credentials associated with their FintechOS Platform account. To make sure account access is protected, after the login credentials are provided, a one-time security pass code is sent to the user's phone (the phone number set in the user account profile) or email address. Once the user enters the code received via the SMS/e-mail, access into the system is granted.

The following authenticator apps have been tested and confirmed as working:

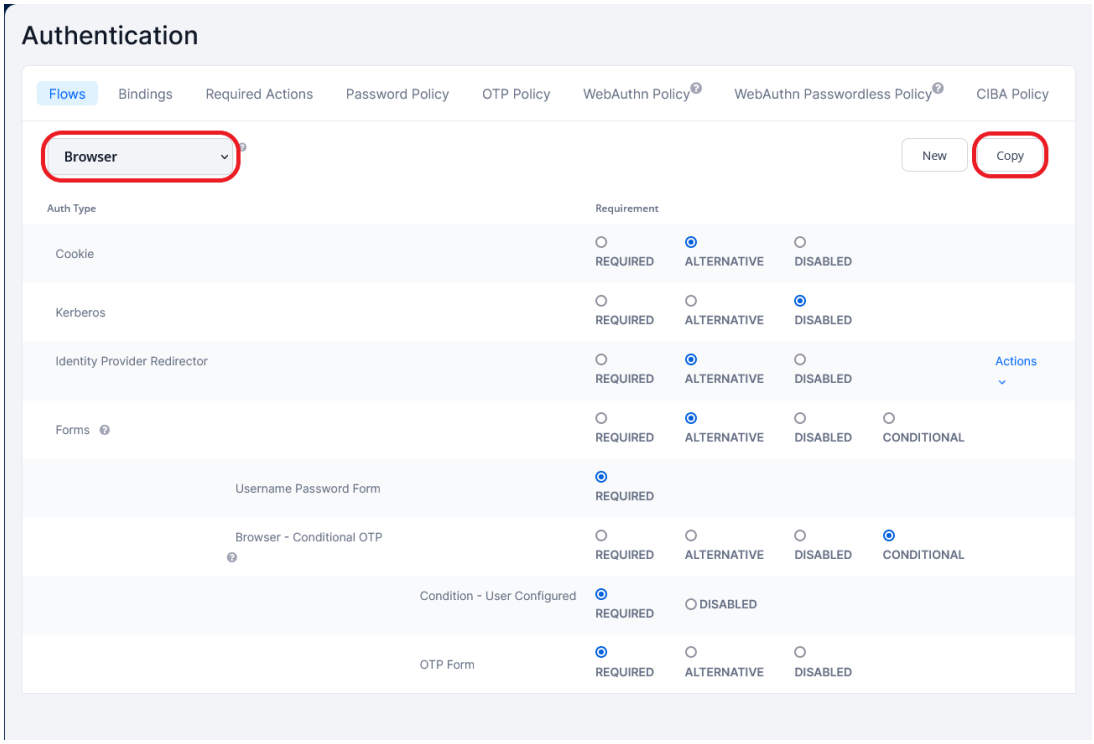
- Microsoft Authenticator
- Google Authenticator
- FreeOTP

Follow the instructions below to set up multi-factor authentication.

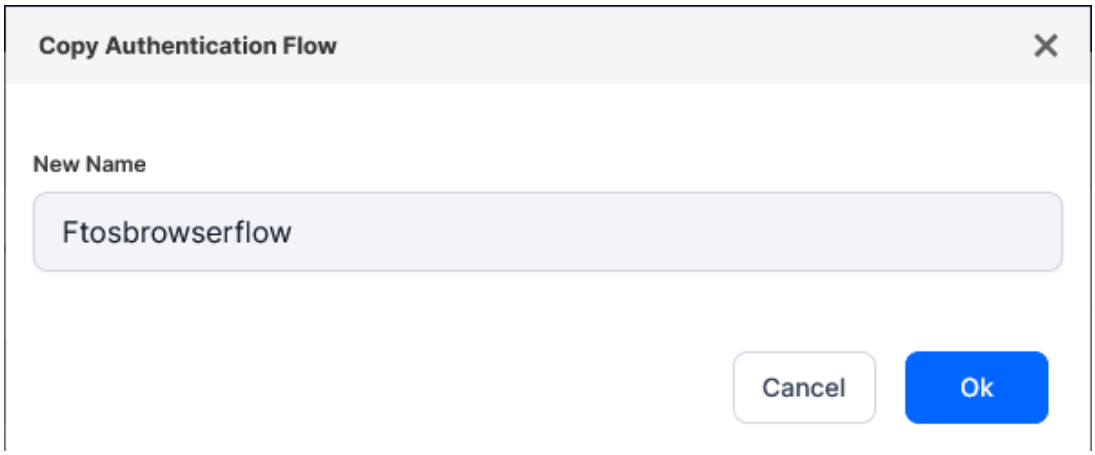
1 Create a Browser Authentication Flow

1. Log in to the FintechOS Identity Provider admin console.
2. Select your FintechOS Platform realm.
3. Select the **Authentication** blade.

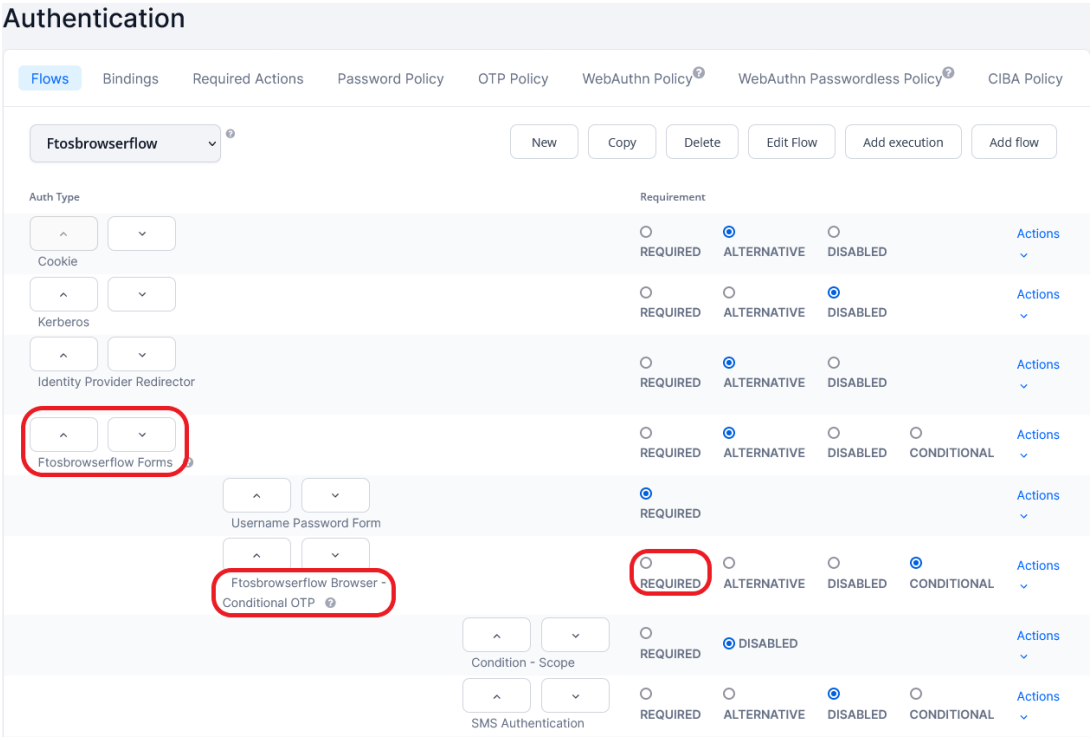
4. In the **Flows** tab, select the **Browser** based authentication flow and click the **Copy** button to create a duplicate.



5. In the window that opens, assign a name for your new browser authentication flow.

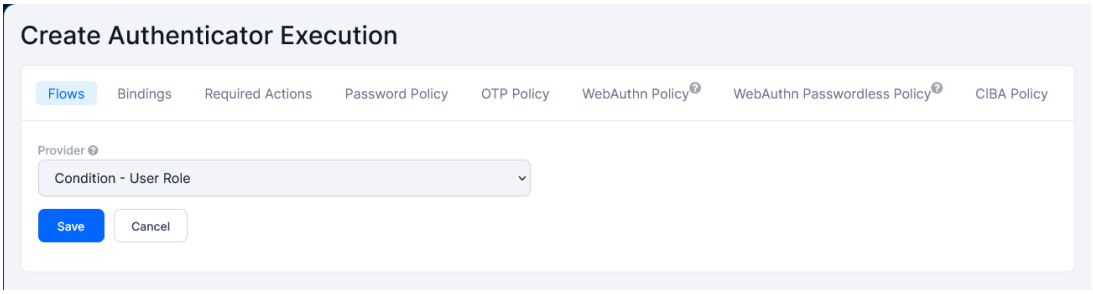


6. Open the newly created authentication flow and, in the **Forms** authentication type, set the **Browser - Conditional OTP** subflow to **Required**.



(Optional) Enable Conditional OTP only for specific roles

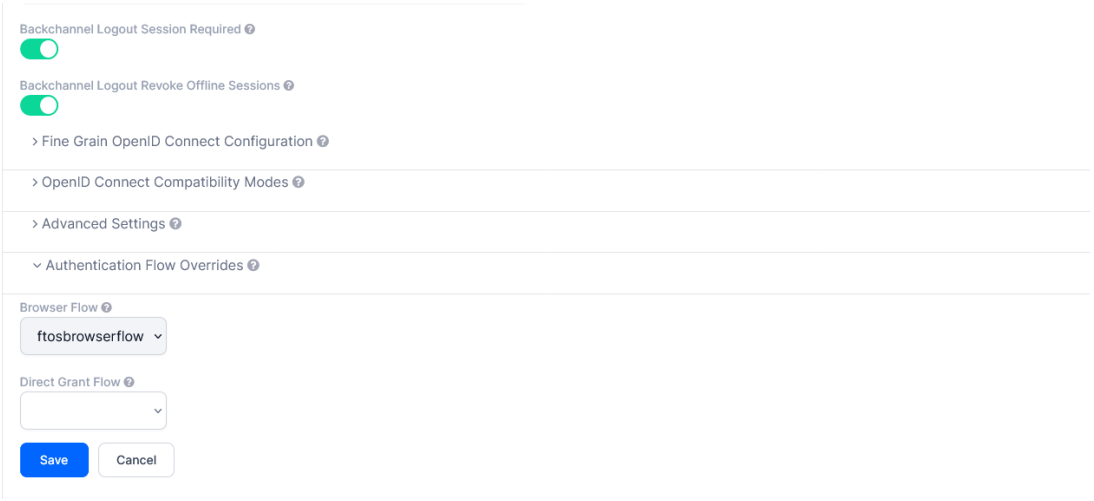
1. Open your authentication flow and, in the **Forms** authentication type, in the **Browser - Conditional OTP** subflow, click the **Actions** drop-down and select **Add execution**.
2. In the Create Authentication Execution screen, select the **Condition - User Role** provider and click **Save**.



3. In the authentication flow window, move the **Condition - User Role** execution above the **OTP Form** execution and, from its **Actions** drop down, select **Config**.
4. In the user authenticator configuration page, add an **Alias** of your choice and select the **Role** which will require multi-factor authentication. If you wish to apply multi-factor authentication to all user roles except the provided role, tick the **Negate output** option.
5. On your authentication flow page, in the **Forms** authentication type, set the **Browser - Conditional OTP** subflow to **Conditional**.

2 Associate the Authentication Flow to a Client

1. In the **Clients** blade, open the FintechOS Identity Provider client you wish to enable multi-factor authentication for, such as a Portal or FintechOS Studio client.
2. In the **Settings** tab, scroll down to the **Authentication Flow Overrides** and expand it.
3. Set the **Browser Flow** option to the flow you created earlier and click **Save**.



Authenticator Reset

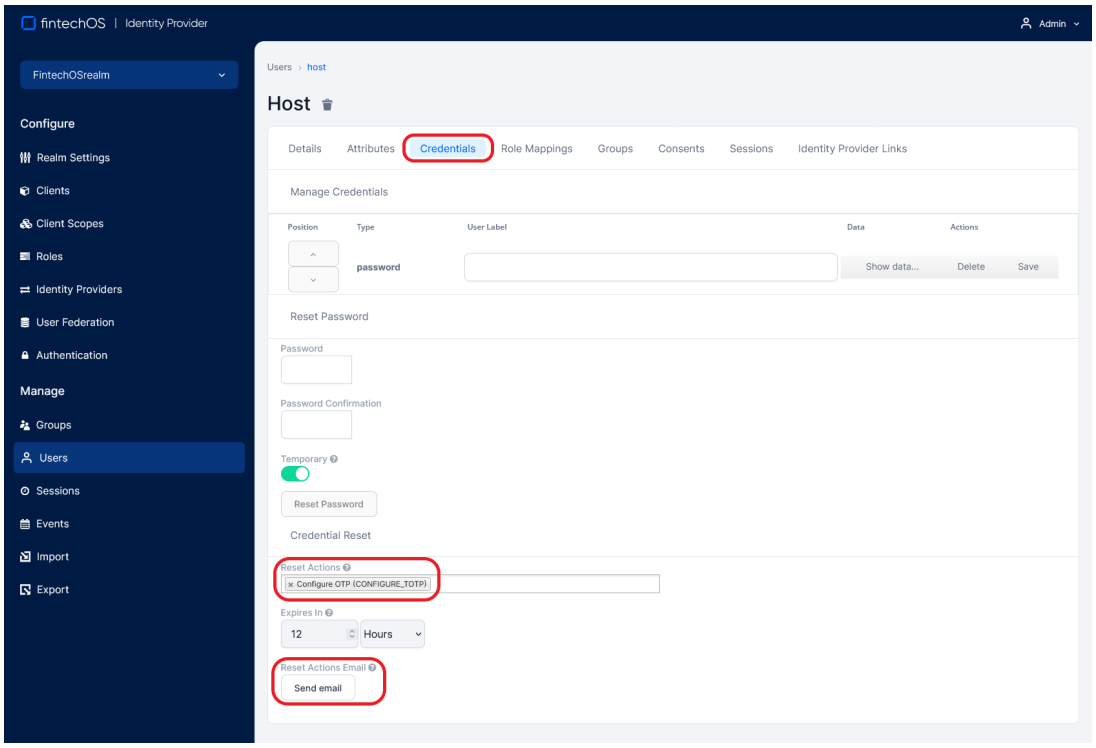
Once set up, the multi-factor authentication will be required when a user logs in for the first time. If a user loses access to the authenticator app or needs to reconfigure for various reasons, a FintechOS Identity Provider administrator must log in to the

administration console and either:

- Delete the created authenticator

or

- From the user's **Credentials** tab, set up a **Configure OTP** reset action.



MFA by Email, SMS

Email and/or SMS Multi-Factor Authentication (MFA) adds an extra layer of security on top of the basic authentication methods. When users access the FintechOS Portal or FintechOS Studio, they will be prompted to provide the login credentials associated with their FintechOS Platform account. To make sure account access is protected, after the login credentials are provided, users are redirected to a secondary login page where they have to enter a one-time security pass code received via email, SMS (the phone number set in the user account profile). Once a user enters the code received via the Email/SMS/IVR(Call Me) option, access to the system is granted.

Follow the instructions below to set up Email/SMS/IVR(Call Me) multi-factor authentication.

1 Provision the Service Pipes App Service in the Designated Resource Group

Make sure the service pipes app service is properly deployed on that Azure RG.

2 Create a Service Pipes Image with Notification Service Dependency

- Create a new repo from the `service-pipes-template`.
- Add the notification service dependency.
- Build and deploy the docker image: use `service-pipes-[projectname]:latest`, test, then create a release.

3 Configure the Configuration Manager with the Notification Providers Values

1. Access the Configuration Manager where the notification service credentials will be stored.
2. Add the `notification.config` key having the following structure:

```
{
                "notificationConfigurations": [
                {
                "realmId": "fintechOSrealm",
                "provider-configurations": [
                {
                "url": "https://api.emailprovider.com",
                "provider": "emailprovider",
                "payload-template": "{\"sender\":
{\"name\":\"SENDER_PLACEHOLDER\",\"email\":\"SENDER_
PLACEHOLDER\"},\"to\":[  {  \"email\":\"RECEIVER_
PLACEHOLDER\",\"name\":\"RECEIVER_
PLACEHOLDER\"}],\"subject\":\"SUBJECT_
PLACEHOLDER\",\"htmlContent\":\"MESSAGE_PLACEHOLDER\"}",
                "default-sender": "noreply@yourdomain.com",
                "communication-channel": "email",
                "authentication": {
                "key": "your-email-api-key"
```

```
                }
                },
                {
                "provider": "smsprovider",
                "url": "https://api.smsprovider.com/",
                "authentication": {
                "username": "yourAccountSid",
                "password": "yourAccountPassword"
                },
                "communication-channel": "sms",
                "payload-template": "From=SENDER_
PLACEHOLDER&To=RECEIVER_PLACEHOLDER&Body=MESSAGE_PLACEHOLDER",
                "default-sender": "+1234567890",
                "phone-number-prefix-to-add": "",
                "phone-number-prefix-to-remove": ""
                }
                ]
                }
                ]
                }
```

Property	Description
url	The service provider's URL that will receive the service request.
payload-template	The template for the request body. It can contain placeholders that will be later replaced with predefined information.
default-sender	If the sender is not provided as an input parameter, this field will be used as the sender of the message. Use only if the service provider supports specifying a sender.
communication-channel	Specifies the communication channel as either `Email`, `SMS`, or `Call Me`.
phoneNumberPrefixToAdd	A string to be prepended to the client phone number before sending the request to the service provider. E.g.: In the FintechOS Identity Provider, the user's phone number is stored in the following format: 40XXXXXXXXX, but the SMS service provider needs the phone number in the following format: +40XXXXXXXXX. To fix this, set the phoneNumberPrefixToAdd to +.

Property	Description
phoneNumberPrefixToRemove	A string to be removed from the beginning of the phone number before sending the request to the service provider. E.g.: In the FintechOS Identity Provider, the user's phone number is stored in the following format: +40XXXXXXXXX, but the SMS service provider needs the phone number in the following format: 40XXXXXXXXX. To fix this, set the phoneNumberPrefixToAdd to +.

4 Test the Notification Service

Use curl requests to send test notifications via email and SMS. These requests need to be authenticated first.

- Authentication

```
curl --location 'https://your-app-service-
url.net/auth/realms/fintechosrealm/protocol/openid-
connect/token' \
                        --header 'Content-Type:
application/x-www-form-urlencoded' \
                        --data-urlencode
'username=yourUsername' \
                        --data-urlencode
'password=yourPassword' \
                        --data-urlencode 'grant_type=client_
credentials' \
                        --data-urlencode 'client_id=service-
pipes-client-id' \
                    --data-urlencode 'client_secret=service-
pipes-client-secret'
```

- Email

```
curl --location 'https://your-app-service-
url.net/services/api/internal/notification' \
                        --header 'X-Sender-Id:
fintechOSrealm' \
                        --header 'X-Communication-Channel: 2'
\
                        --header 'Content-Type:
application/json' \
                        --data-raw '{
```

```
                        "sender": "info@donotreply.com",
                        "receiver": "receiver@example.com",
                        "message": "hello",
                        "subject": "test"
                    }'
```

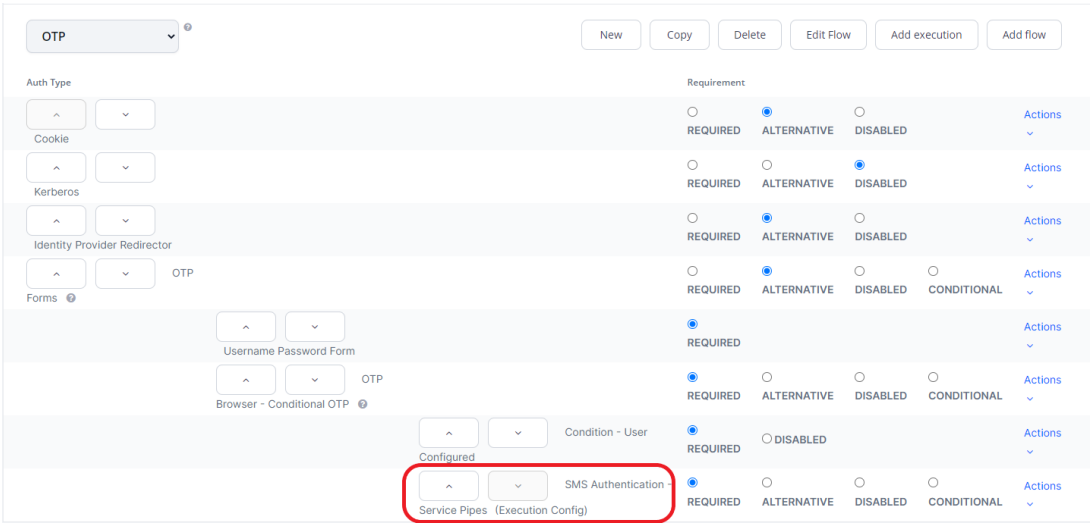
- SMS

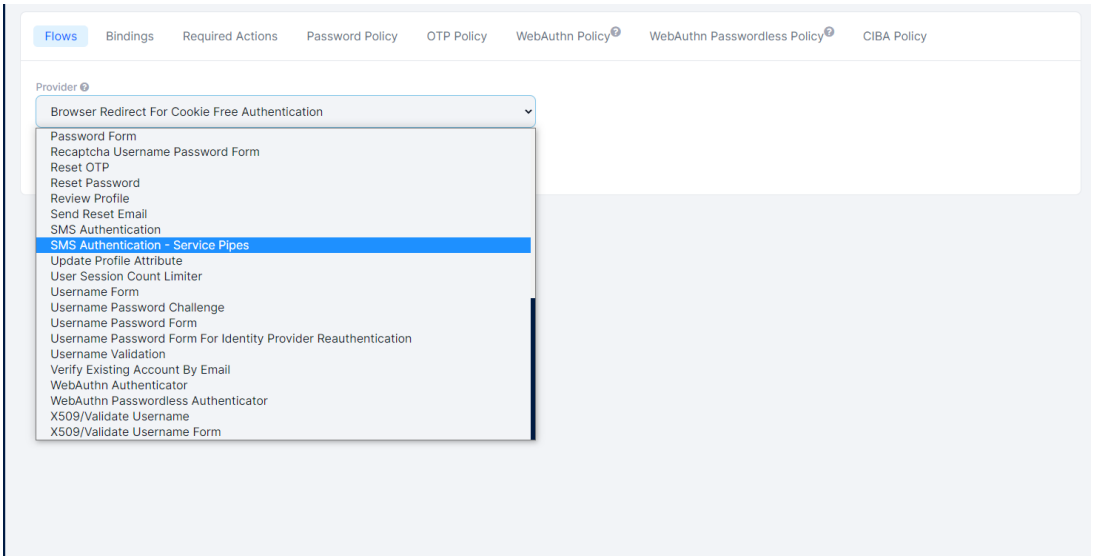
```
curl --location 'https://your-app-service-
url.net/services/api/internal/notification' \
                        --header 'X-Sender-Id:
fintechOSrealm' \
                        --header 'X-Communication-Channel: 1'
\
                        --header 'Content-Type:
application/json' \
                        --data-raw '{
                        "message": "[message]",
                        "receiver": "[RecipientPhoneNumber]",
                        "sender": "[SenderPhoneNumber]"
                    }'
```

5 Create an Authentication Flow

> **IMPORTANT!**
> Make sure that the FintechOS Identity Provider has the MFA plugin installed and uses a theme that includes the MFA login screen.

1. Log in to the FintechOS Identity Provider admin console.
2. Select your FintechOS Platform realm.
3. Select the **Authentication** blade.
4. In the **Flows** tab, create a new authentication flow based on your preferences.
5. In the **Forms**authentication type, set the flow's last execution step to use the **SMS Authentication - Service Pipes** provider.





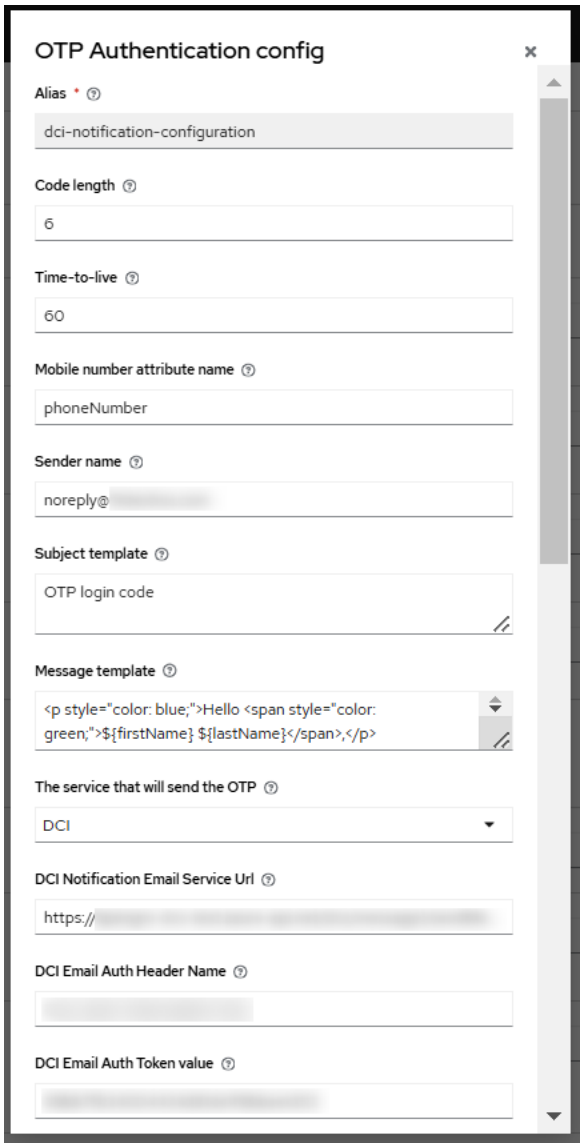
6 Configure the Flow's MFA Execution Step

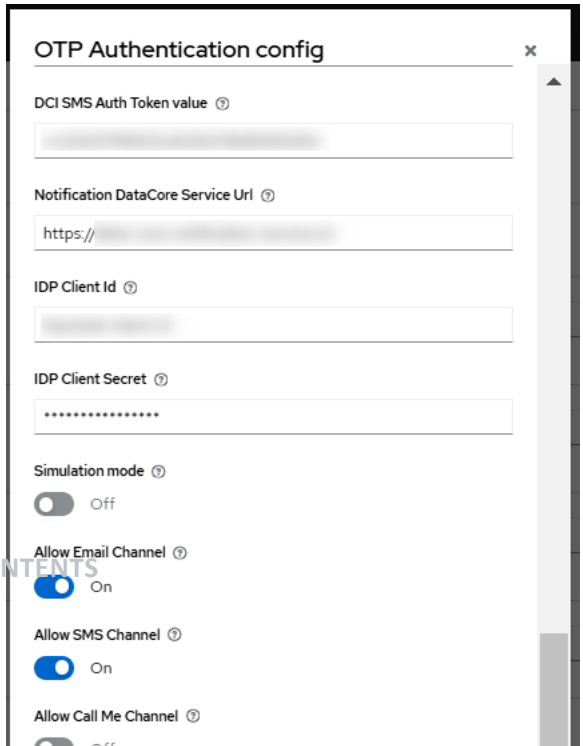
1. Open your authentication flow and navigate to the **SMS Authentication - Service Pipes** execution step.
2. Click the **Actions** drop-down and select **Config**.

3. Set up the following parameters:

    - **Code length**: The length of the one-time password that is going to be sent to the user.
    - **Time-to-live**: The duration in seconds the one-time password is valid.
    - **Mobile number attribute name**: The attribute name that is going to be used to retrieve the user's phone number.
    - **Sender name**: The sender's email address displayed to the end user when email notifications are sent. It is recommended to pick a sender email address other than generic sender IDs such as "SMS", "INFO", "Alert", "no-reply", and so on. These could be automatically rejected by some email services.
    - **Subject template**: The message subject template that is going to be sent to the user when using the email communication channel. You can use the following placeholders: ${firstName}, ${lastName}, ${content}.
    - **Message template**: The message content template that is going to be sent to the user via SMS or email. You can use the following placeholders: ${firstName}, ${lastName}, ${content}.
    - **The service that will send the OTP**: Choose between DCI and DataCore.
    - **DCI Notification Email Service Url**: example: `[environemntURL]/dcs/message1/sendMessage`
    - **DCI Email Auth Header Name**: The DCI authorization header name for email service, for example `X-Apim-Subscription-Key`
    - **DCI Email Auth Token value**: The authorization token for the DCI email service, for example: `01234567890abcdef01234567890abcdef`
    - **DCI Notification SMS Service Url**: Add the service URL if you enforce SMS notification, for example: `[environmentURL]/dcs/message3/sendocbsms`

- **DCI SMS Auth Header Name**: Add the authentication header name if SMS service, for example `X-Apim-Subscription-Key`
- **DCI SMS Auth Token value:** Authorization token for SMS service, for example: `01234567890abcdef01234567890abcdef`
- **Notification DataCore Service Url**: Add service URL if using Data Core, for example: `https://app-envoy-rgname-level.azurewebsites.net/datacore`
- **IDP Client Id**: The client id used for obtaining valid access tokens to call the DataCore service.
- **IDP Client Secret**: The client secret used for obtaining valid access tokens to call the DataCore service.
- **Simulation mode**: When set to True, the actual one-time password will not be sent to the user. Instead, the one-time password will be logged on the FintechOS Identity Provider server.
- **Allow Email channel**: If set to true, the user will be able to select Email as communication channel for receiving the OTP.
- **Allow SMS channel**: If set to true, the user will be able to select SMS as communication channel for receiving the OTP.
- **Allow Call Me channel**: If set to true, the user will be able to select Call Me as communication channel for receiving the OTP.
- **Skip OTP selection screen**: if only one channel is configured, then only that option is displayed to the end user
- **Allow OTP resend mechanism**: allows the user to request another OTP code. The number of retries is configured in the below box **Number of maximum OTP resend retries**.


OTP Authentication config
Alias *
dci-notification-configuration
Code length
6
Time-to-live
60
Mobile number attribute name
phoneNumber
Sender name
noreply@
Subject template
OTP login code
Message template
Hello ${firstName} ${lastName},
The service that will send the OTP
DCI
DCI Notification Email Service Url
https://
DCI Email Auth Header Name
DCI Email Auth Token value


OTP Authentication config
DCI SMS Auth Token value
Notification DataCore Service Url
https://
IDP Client Id
IDP Client Secret
Simulation mode
Off
Allow Email Channel
On
Allow SMS Channel
On
Allow Call Me Channel

4. Click **Save**.

If the user adds the wrong OTP code for 8-10 times, their FintechOS IDP user is disabled and can be enabled by an admin user.

To modify the text in the UI in the OTP validation screen, including the text above the radio buttons for picking the validation option (email, SMS, call), go to the FintechOS IDP, choose your UI theme, and inside the UI theme folder navigate to baseStudio > login > messages > messages_en.properties, and change the key values. For example, to display "Send SMS" in screen UI, modify the following key to show: `otpSelectionFormSms=Send SMS`.

7 Activate the Authentication Flow

Once the authentication flow is configured, in the FintechOS Identity Provider, navigate to **Authentication** > **Bindings** and replace the Browser flow with the newly created flow. This will the set authentication flow globally. However, if you want to enable the flow only for one client, you can navigate to the client page, go to **Settings** > **Authetication Flow Overrides**, and change the **Browser Flow** there.

A user who logs in to FintechOS Portal or FintechOS Studio, will be directed to the one-time password form and will receive the required password via Email, SMS or IVR (Call Me).

Deprecated Multi-Factor Authentication

IMPORTANT!
Starting with release 22.1, the FintechOS Platform uses the FintechOS Identity Provider as the default authentication layer for the FintechOS applications and services. Alternate identity providers and the corresponding multi-factor authentications are supported only for backward compatibility.
For information about the FintechOS Identity Provider multi-factor authentication, see "Browser Based Multi-Factor Authentication" on page 200.

Multi-Factor Authentication (MFA) adds an extra layer of security on top of the basic authentication methods. It requires users to provide multiple proof of their claimed identity prior being granted access to resources based on business need to know according to their security role and granted permissions.

User access can be granted on two-factors:

- Something the user knows (login credentials): username and password.
- Something the user has (pass code received via an SMS/E-mail or mobile soft token).

When users access the app, they will be prompted to provide the login credentials associated with their FintechOS Platform account. To make sure account access is protected, after the login credentials are provided, a one-time security pass code is sent to the user's phone (the phone number set in the user account profile) or email. Once the user enters the code received via the SMS/e-mail, access into the system is granted.

SMS-based Two-Factor Authentication

SMS-based two-factor authentication is the most popular choice when it comes to multi-factor authentication as most users have their own mobile phones and have them handy when logging into apps.

How it works?

Users will be granted access to the FintechOS Platform app following these two steps:

- Users will navigate to the FintechOS Platform web app and they will provide their account credentials (username and password). Based on the app configuration, the credentials can be either local or from external providers. To make sure account access is protected, after the login credentials are provided, a one-time security pass code is sent to the user's phone (the phone number set in the user account profile).
- In the web app, they provide the pass code received via SMS. Access into the system is granted.

How to set up the SMS-based MFA?

Setting up the SMS-based multi-factor authentication is a two-step process:

1 Enable Multi-Factor Authentication

On the server where the FintechOS Platform installation package resides, go to the **web.config** file and add the following settings:

- To the **<configSections>** tag, add the **multifactorAuthentication** section:

```
<configuration>
    <configSections>
        ...
         <section name="multiFactorAuthentication" type=
"EBS.Core.Authentication.Common.MultiFactorAuthentication.Co
nfig.MultiFactorAuthenticationSection,
EBS.Core.Authentication.Common" />
    </configSections>
<configuration>
```

- Add the **<multiFactorAuthentication>** tag:

```
<configuration>
...
    <multiFactorAuthentication
xmlns
=
"http://fintechos.com/ebs/schemas/multiFactorAuthentication"
 enabled="true">
        <providers>
            <provider
name="SMS" enabled="true" default="true">
                <type fullName=
"EBS.Core.Authorization.Services.Security.SmsMultiFactorAuth
enticationProvider, EBS.Core.Authorization.Services" />
                <properties>
                    <property
name="ChannelProvider" value="GatewaySmsOTP" />
                    <property
name="CommunicationChannel" value="Sms" />
                    <property
name="MaxNumberOfAuthenticationRetries" value="3" />
                    <property
name="MaxNumberOfSmsSendings" value="3" /> </properties>
            </provider>
        </providers>
        <runtime>
            <providers>
                <provider name="SMS">
                    <roles>
```

```
                        <role name="*" /> </roles>
                </provider>
            </providers>
        </runtime>
    </multiFactorAuthentication>
</configuration>
```

Where:

- `multiFactorAuthentication/@enabled` - controls if MFA is enabled or not. Default value: false;
- `multiFactorAuthentication/providers/provider/@enabled` - controls if a specific MFA provider is enabled or not. Default value: true;
- `multiFactorAuthentication/providers/provider/@default`
  - When MFA is enabled, there can be at most one provider marked with default="true".
  - The provider marked with default="true" will be selected for MFA when there are multiple providers available for user's roles and the user hasn't selected any preferred communication channel or his preferred communication channel is not present into the configured providers list.
  - Default value: false.
- on `multiFactorAuthentication/providers/provider/properties`:
  - `ChannelProvider` - the channel provider used by the SMS MFA provider to send text messages. Its value must be the Name of one of the records from **EbsMetadata.FTOS_DPA_ChannelProvider** table;
  - `CommunicationChannel` - the communication channel used by the SMS provider to send text messages. Its value must be the Name of one of the records from **EbsMetadata.FTOS_DPA_CommunicationChannel** table;
  - `MaxNumberOfAuthenticationRetries` - the user has up to MaxNumberOfAuthenticationRetries chances to enter the correct code. If this

threshold is reached the user will be redirected to the login page. Default value: **3**;

- `MaxNumberOfSmsSendings` - if needed the user may request a resending of the code for up to MaxNumberOfSmsSendings times. If this threshold is reached the user will be redirected to the login page. Default value: **3**;

- `multiFactorAuthentication/runtime/providers/provider[@name='SMS']/roles` will include a <role name=""/> child for each role that contains users that have to be authenticated through this provider. Note that a `<role name="*"/>` means that all roles will be taken into account;

If the multi-factor authentication is activated, at the next profile change, users will have to provide their phone number (in the Edit System User, My Account page, the Phone Number field is mandatory).

Once you've activated the SMS-based authentication, you need to configure the Job Server for Multi-Factor Authentication.

2 Configure the Job Server for MFA

> **IMPORTANT!**
> The MessageBus (OCS) plugin for the FintechOS Job Server already includes the configurations required for multi-factor authentication (see the *FintechOS Installation Guide* for details about MessageBus (OCS) installation).
>
> - If you have the MessageBus (OCS) plugin installed, skip this step.
> - If you are using the standard Job Server configuration, follow the instructions below to configure the multi-factor authentication settings.

1. On the server where the FintechOS Platform installation package resides, go to the **schedule.config** file and add the following section:

```
<triggers>
...
```

```
    <trigger>
        <name>FTOS.OCB.OTP</name>
        <startTime>02.11.2017 11:00</startTime>
        <endTime>03.11.2080 11:02</endTime>
        <repeatCount>-1</repeatCount>
        <rescheduleAfterRun>false</rescheduleAfterRun>
        <async>false</async>
        <expression>0/10 * * * * ?</expression>
        <services>

<service>FTOS.OCB.SendMessagesServiceSmsOTP</service>

<service>FTOS.OCB.UpdateStatusServiceSmsOTP</service>

<service>FTOS.OCB.UpdateExpiredMessageServiceSmsOTP</service>
        </services>
    </trigger>
</triggers>
```

2. On the server where the [[[Undefined variable General.PlatformName]]] installation package resides, go to the **services.config** file and add the following sections:

```
<serviceList>
...
    <!--OTP-->
    <service>
        <name>FTOS.OCB.SendMessagesServiceSmsOTP</name>
        <type>class</type>
        <method></method>

<
class
>
FTOS.MessageBus.ScheduledServices.SendMessagesService</class>

<assembly>FTOS.MessageBus.ScheduledServices</assembly>-->

<execParams>
provider=gateway;providerSetting=gatewaySmsOTP</execParams>
    </service>
    <service>
        <name>FTOS.OCB.UpdateStatusServiceSmsOTP</name>
        <type>class</type>
        <method></method>
```

```
        <class>
FTOS.MessageBus.ScheduledServices.UpdateStatusMessagesService
</class>

<assembly>FTOS.MessageBus.ScheduledServices</assembly>-->

<execParams>
provider=gateway;providerSetting=gatewaySmsOTP</execParams>
    </service>
    <service>

<name>FTOS.OCB.UpdateExpiredMessageServiceSmsOTP</name>
        <type>class</type>
        <method></method>
        <class>
FTOS.MessageBus.ScheduledServices.UpdateExpiredMessageService
</class>

<assembly>FTOS.MessageBus.ScheduledServices</assembly>-->

<execParams>
provider=gateway;providerSetting=gatewaySmsOTP</execParams>
    </service>
</serviceList>
```

Configure Multi Factor Authentication to use an SMS Service provider

In the web.config file, set the ChannelProvider property of the MFA provider with value “FTOSApiSms“.

Example

```
<multiFactorAuthentication xmlns
="http://fintechos.com/ebs/schemas/multiFactorAuthenticatio
n" enabled="true">
        <providers>
            <provider name="SMS" enabled="true">
                <type fullName
="EBS.Core.Authorization.Services.Security.SmsMultiFactorAut
henticationProvider, EBS.Core.Authorization.Services" />
                <properties>
                    <property
name="ChannelProvider" value="FTOSApiSms" />
                    <property
name="MaxNumberOfAuthenticationRetries" value="3" />
```

```
                    <property
name="MaxNumberOfSmsSendings" value="3" />
                    <property
name="MessageTemplate" value="myMessageTemplate_SmsApi" />
                </properties>
            </provider>
        </providers>
        ...
</multiFactorAuthentication>
```

Password reset SMS for the log-in credentials

To set up password reset confirmation:

Add section

```
<configSections>
...
  <section name="passwordReset"
type="EBS.Core.Web.MVC.PasswordResetConfig, EBS.Core.Web.MVC"/>
</configSections>
```

Add configuration element

```
<configuration>
...
  <passwordReset xmlns="urn:EBS.Core.Web.MVC">
    <confirmation channelProvider="" messageTemplate=""
enabled="true"/>
  </passwordReset>
...
</configuration>
```

where:

- enabled - if true, after the completion of the password reset flow a message will be sent to user's phone number. Default value: false;

- channelProvider - the provider that will be used to send the message. Must be one of “GatewaySmsOTP“ or “FtosApiSms“;
- messageTemplate - the template that will be used to create the message. Must be a record from FTOS_CMB_ActionTemplate entity.

If the configuration element is missing the message will not be sent.

Email-based Two-Factor Authentication

Email-based two-factor authentication is a popular choice when it comes to multi-factor authentication.

How it works?

Users will be granted access to the FintechOS Platform app following these two steps:

- Users will navigate to the FintechOS Platform web app and they will provide their account credentials (username and password). Based on the app configuration, the credentials can be either local or from external providers. To make sure account access is protected, after the login credentials are provided, a one-time security pass code is sent to the user’s email address.
- In the web app, they provide the pass code received via email. Access into the system is granted.

How to set up the Email-based MFA?

Setting up the email-based multi-factor authentication is a two-step process:

Step 1 Enable Multi-Factor Authentication

On the server where the FintechOS Platform installation package resides, go to the **web.config** file and add the following settings:

- To the **<configSections>** tag, add the **multifactorAuthentication** section:

```
<configuration>
    <configSections>
        ...
        <section name="multiFactorAuthentication" type=
"EBS.Core.Authentication.Common.MultiFactorAuthentication.Co
nfig.MultiFactorAuthenticationSection,
EBS.Core.Authentication.Common" /> </configSections>
    <configuration>
```

- Add the **<multiFactorAuthentication>** tag:

```
<configuration>
    ...
    <multiFactorAuthentication
xmlns
=
"http://fintechos.com/ebs/schemas/multiFactorAuthentication"
 enabled="true">
        <providers>
            <provider
name="Email" enabled="true" default="true">
                <type fullName=
"EBS.Core.Web.MVC.Security.EmailMultiFactorAuthenticationPro
vider, EBS.Core.Web.MVC" />
                <properties>
                    <property
name="ChannelProvider" value="GatewayEmailOTP" />
                    <property
name="CommunicationChannel" value="Email" />
                    <property
name="MaxNumberOfAuthenticationRetries" value="3" />
                    <property
name="MaxNumberOfEmailSendings" value="3" /> </properties>
            </provider>
        </providers>
        <runtime>
            <providers>
                <provider name="Email">
                    <roles>
                        <role name="*" /> </roles>
                </provider>
            </providers>
        </runtime>
    </multiFactorAuthentication>
```

```
</configuration>
```

Where:

- `multiFactorAuthentication/@enabled` - controls if MFA is enabled or not. Default value: false;
- `multiFactorAuthentication/providers/provider/@enabled` - controls if a specific MFA provider is enabled or not. Default value: true;
- `multiFactorAuthentication/providers/provider/@default`
    - When MFA is enabled, there can be at most one provider marked with default="true".
    - The provider marked with default="true" will be selected for MFA when there are multiple providers available for user's roles and the user hasn't selected any preferred communication channel or his preferred communication channel is not present into the configured providers list.
    - Default value: false.
- on `multiFactorAuthentication/providers/provider/properties`:
    - `ChannelProvider` - the channel provider used by the Email MFA provider to send email messages. Its value must be the Name of one of the records from **EbsMetadata.FTOS_DPA_ChannelProvider** table;
    - `CommunicationChannel` - the communication channel used by the Email provider to send email messages. Its value must be the Name of one of the records from **EbsMetadata.FTOS_DPA_CommunicationChannel** table;
    - `MaxNumberOfAuthenticationRetries` - the user has up to MaxNumberOfAuthenticationRetries chances to enter the correct code. If this threshold is reached the user will be redirected to the login page. Default value: **3**;

- `MaxNumberOfEmailSendings` - if needed the user may request a resending of the code for up to MaxNumberOfSmsSendings times. If this threshold is reached the user will be redirected to the login page. Default value: **3**;

- `multiFactorAuthentication/runtime/providers/provider [@name='Email']/roles` will include a <role name=""/> child for each role that contains users that have to be authenticated through this provider. Note that a `<role name="*"/>` means that all roles will be taken into account;

> **NOTE**
>
> For the platform authentication to be successful, make sure all users have assigned the configuration security role.

Once you've activated the Email-based authentication, you need to configure the Job Server for Multi-Factor Authentication.

Step 2. Configure the Job Server for MFA

> **IMPORTANT!**
>
> The MessageBus (OCS) plugin for the FintechOS Job Server already includes the configurations required for multi-factor authentication (see the *FintechOS Installation Guide* for details about MessageBus (OCS) installation).
>
> - If you have the MessageBus (OCS) plugin installed, skip this step.
> - If you are using the standard Job Server configuration, follow the instructions below to configure the multi-factor authentication settings.

1. On the server where the [[[Undefined variable General.PlatformName]]] installation package resides, go to the **schedule.config** file and add the following section:

```
<triggers>
    ...
    <trigger>
```

```
        <name>FTOS.OCB.OTP</name>
        <startTime>02.11.2017 11:00</startTime>
        <endTime>03.11.2080 11:02</endTime>
        <repeatCount>-1</repeatCount>
        <rescheduleAfterRun>false</rescheduleAfterRun>
        <async>false</async>
        <expression>0/10 * * * * ?</expression>
        <services>

<service>FTOS.OCB.SendMessagesServiceEmailOTP</service>

<service>FTOS.OCB.UpdateStatusServiceEmailOTP</service>

<service>
FTOS.OCB.UpdateExpiredMessageServiceEmailOTP</service>
        </services>
    </trigger>
</triggers>
```

2. On the server where the [[[Undefined variable General.PlatformName]]] installation package resides, go to the **services.config** file and add the following sections:

```
<serviceList>
    ...
    <!--OTP-->
    <service>
        <name>FTOS.OCB.SendMessagesServiceEmailOTP</name>
        <type>class</type>
        <method></method>

<
class
>
FTOS.MessageBus.ScheduledServices.SendMessagesService</class>

<assembly>FTOS.MessageBus.ScheduledServices</assembly>-->

<execParams>
provider=gateway;providerSetting=GatewayEmailOTP</execParams>
    </service>
    <service>
        <name>FTOS.OCB.UpdateStatusServiceEmailOTP</name>
        <type>class</type>
        <method></method>
```

```
        <class>
FTOS.MessageBus.ScheduledServices.UpdateStatusMessagesService
</class>

<assembly>FTOS.MessageBus.ScheduledServices</assembly>-->

<execParams>
provider=gateway;providerSetting=GatewayEmailOTP</execParams>
    </service>
    <service>

<name>FTOS.OCB.UpdateExpiredMessageServiceSmsOTP</name>
        <type>class</type>
        <method></method>
        <class>
FTOS.MessageBus.ScheduledServices.UpdateExpiredMessageService
</class>

<assembly>FTOS.MessageBus.ScheduledServices</assembly>-->

<execParams>
provider=gateway;providerSetting=GatewayEmailOTP</execParams>
    </service>
</serviceList>
```

Register TLS Client Certificates

Client certificates allow you to access web services that require client authentication via the TLS/SSL protocol. Once a certificate is registered, you can refer it in your server side scripts and include it in API calls.

To register a TLS Client Certificate add the following secret in Vault:

Key Path	Key Name
kv/<environment>/<application>/app-settings	automation-client-certificate-clientCert1

Key Name	Key Value
automation-client-certificate-clientCert1	"{ 'storeName': 'My', 'storeLocation': 'LocalMachine', 'thumbPrint': 'd77621fa50114404a6e5820c6d066b019c13fdd8', 'description':'Client certificate for Api1' }"

You must provide a programmatic **name**, preceded by the `automation-client-certificate-` prefix. For instance, in the example above, the name of the client certificate is going to be *clientCert1*.

The **value** is provided in JSON format and must be XML escaped. For simpler scenarios you can use single quotes instead of double quotes. The JSON value has the following structure:

```
{
    "storeName": "My",
    "storeLocation": "LocalMachine",
    "thumbPrint": "d77621fa50114404a6e5820c6d066b019c13fdd8",
    "description": "Client certificate for Api1",
    "checkValidity": true
}
```

Property	Description
storeName	You can populate the storeName property with one of the following values: • AddressBook - X.509 certificate store for other users. • AuthRoot - X.509 certificate store for third-party certificate authorities. • CertificateAuthority - X.509 certificate store for intermediate certificate authorities. • Disallowed - X.509 certificate store for revoked certificates. • My - X.509 certificate store for personal certificates. • Root - X.509 certificate store for trusted root certificate authorities. • TrustedPeople - X.509 certificate store for directly trusted people and resources. • TrustedPublisher - X.509 certificate store for directly trusted publishers.
storeLocation	You can populate the storeLocation property with one of the following values: • CurrentUser - X.509 certificate store used by the current user. • LocalMachine - X.509 certificate store assigned to the local machine.
thumbPrint	This is the thumbprint of the client certificate.
description	A user-friendly description of the certificate. This information will be displayed in the code editor's intelligent code completion suggestions.

Property	Description
checkValidity	• true - Even if the thumbprint is found, the API returns the certificate only if the root issuer in the certificate build chain is part of the trusted root certification authorities. • false - For development or testing purposes.

(Deprecated) Add configuration in web.config files:

```
<app-settings>
    ...
    <add key="automation-client-certificate-
clientCert1" value="{ 'storeName': 'My', 'storeLocation':
'LocalMachine', 'thumbPrint':
'd77621fa50114404a6e5820c6d066b019c13fdd8',
'description':'Client certificate for Api1' }"/>
    or
    <add key="automation-client-certificate-
clientCert1" value="{ &apos;storeName&apos;: &apos;My&apos;,
&apos;storeLocation&apos;: &apos;LocalMachine&apos;,
&apos;thumbPrint&apos;:
&apos;d77621fa50114404a6e5820c6d066b019c13fdd8&apos;,  &apos
;description&apos;:&apos;Client certificate for Api1 a&apos;
}"/>
    ...
</app-settings>
```

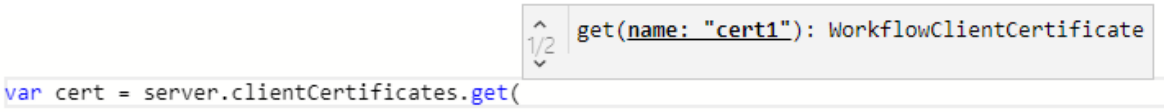
Usage in server-side scripts

The automation API supports referencing client certificates and passing them in the httpGet/httpPost functions. For more information, see the Server SDK Reference Guide documentation.

```
var cert = server.clientCertificates.get('clientCert1');
var getResult = httpGet('https://server.com/route1', {}, {
    clientCertificate: cert
});
var postResult = httpPost('https://server.com/route2', myPostData,
{
```

```
    clientCertificate: cert
});
```

In the code editor, the `server.clientCertificates.get` function provides automatic code completion suggestions for the registered client certificates.



Configure JSON Web Token (JWT) Providers

An access token is a security key issued by an authorization server to provide access to Web APIs and other protected resources.

To access a resource that uses JWT token authentication, you need to register the connection settings for that resource's authentication provider in Vault secrets. Once the provider is set up, you can refer it in your server-side scripts to retrieve an access token for the supported resources.

To register a JWT authentication provider, add the following secrets:

Key Path	Key Name
kv/<environment>/<application>/app-settings	feature-jwt-token-provider-a
kv/<environment>/<application>/app-settings	feature-jwt-token-provider-b

For each key, you must provide a programmatic name, preceded by the `feature-jwt-token-provider-` prefix. The programmatic name must also match the `name` property provided in the key's value. For instance, in the example above, the name of the providers are going to be `a` and `b` respectively.

The value is provided in JSON format and must be XML escaped. For simpler scenarios you can use single quotes instead of double quotes, as exemplified above.

The following properties are generic and apply to all authentication providers:

Property	Description
`name` *(required)*	The provider's name. The value has to be unique in the provider keys registry collection.
`type` *(required)*	The type of provider. Currently, only the Azure Active Directory token provider is supported, so the only valid value is `azure-ad-provider`. More authentication providers may become available in the future.
`timeout` *(optional)*	The service timeout in milliseconds, indicating for how long the application should wait for the token to be generated. Default value: 10000 (10 seconds).

In addition to the generic properties, you must also provide settings that are particular to each authentication provider, in accordance with their specifications. Currently, only Azure Active Directory is supported, with additional providers to be potentially added in the future. For Azure Active Directory, the following properties apply:

Property	Description
`scopeForAccessToken` *(required)*	Azure AD scope for which the access is requested.
`instance` *(required)*	Azure AD instance name.
`tenantId` *(required)*	Azure AD tenant ID.
`clientId` *(required)*	Azure AD client ID.
`clientSecret`	The application client secret used to retrieve the access token. The property is mutually exclusive with `clientCertificate`, but one of them must be set. The client secret must exist in the targeted Azure AD instance.
`clientCertificate`	The certificate used to retrieve the access token. The property is mutually exclusive with `clientSecret`, but one of them must be set. The certificate should be registered in the targeted Azure AD instance. For more information about client certificates, see "Register TLS Client Certificates" on page 226.

(Deprecated) Add keys in the web.config file

```
<app-settings>
   ...
   <!-- Token provider configuration using client secret -->
   <add key="feature-jwt-token-provider-a" value="{
    'name': 'a',
    'type': 'azure-ad-provider',
```

```
    'scopeForAccessToken': 'api://xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx/.default',
    'instance': 'https://login.microsoftonline.com/',
    'tenantId': 'yyyyyyyy-yyyy-yyyy-yyyy-yyyyyyyyyyyy',
    'clientId': 'zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzzzzzz',
    'timeout': 10000,
    'clientSecret': '~xyzxyz-xxxxxxxxxx-yyyyyyyyy-zzz~x'
}"/>


   <!-- Token provider configuration using client
certificate -->
   <add key="feature-jwt-token-provider-b" value="{
    'name': 'b',
    'type': 'azure-ad-provider',
    'scopeForAccessToken': 'api://xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx/.default',
    'instance': 'https://login.microsoftonline.com/',
    'tenantId': 'yyyyyyyy-yyyy-yyyy-yyyy-yyyyyyyyyyyy',
    'clientId': 'zzzzzzzz-zzzz-zzzz-zzzz-zzzzzzzzzzzz',
    'timeout': 10000,
    'clientCertificate': {
        'storeName': 'My',
        'storeLocation': 'CurrentUser',
        'thumbPrint':
'xyzxyzxyzxyzxyzxyzxyzxyzxyzxyzxyzxyzxyzxyz',
        'description': 'Client certificate for wso2-devel',
        'checkValidity': false
    }
}"/>

</app-settings>
```

Usage in server-side scripts

To retrieve a JWT access token in a server-side script, use the `getJwtTokenByProviderName` function in the code editor. For example:

```
var myToken = getJwtTokenByProviderName('a')
```

For more information, see the Server SDK Reference Guide.

Authorization

In FintechOS Platform, access to specific resources (authorization) is done via security role-based access which enables you to

- Protect information from being mishandled by users.
- Ensure that users have access to information based on business need to know.

This section covers platforms' critical aspects of segregation of duties and data ownership.

Security Roles

Users with elevated privileges (admin users) can control data access by setting up the organizational structure to protect sensitive data and configuring various organization layers to allow communication, collaboration or reporting.

To set up the organizational structure, they need to create the business units, security roles, and assign users the appropriate security roles to map the job-related responsibilities with the required level of access privileges within the platform.

You can grant even more granular access privileges in FintechOS Platform, by associating security roles to digital journeys, digital journey steps, business workflows, dashboards, endpoints and DB tasks. The data is automatically filtered based on the privileges and level of access defined within the security role via the security items.

The lowest level of access privileges you can grant to users in FintechOS Platform is on attribute level. You can choose if a specific attribute (field) is to be mandatory, recommended or optional, by selecting the desired option from the Required Level drop-down:

- None – The field is optional. No error message will be displayed if the field is empty.
- Recommended – A blue dot will be displayed on the upper-left corner of the field in the user interface to indicate that it might be useful to fill in the field.

- Required - A red dot will be displayed on the upper-left corner of the field in the user interface to indicate that it is a mandatory field. The end user will not be able to add a new record if the field will be left blank.

NOTE

- You can only add required attributes to entities which have no records (empty entities), so if you try adding a required attribute to an entity for which you already have required attributes stored within the database, you'll receive an error message.
- You can add required attributes without creating constraints in the database, from entity form/digital journey configuration page, Advanced tab > After Events tab, by providing a code in the JavaScript field and the capabilities of field options.

For information on how create security roles and how to provide granular access to entities, digital journeys and dashboards, see the *FintechOS Studio User Guide*.

Data Ownership

In FintechOS Platform, data ownership is given by the security roles, which allows you to manage complex scenarios of access privileges and the level of access.

Admin users are the ones who can define the organizational structure, create users and assign the security roles according to the business need-to-know, inline with their job responsibilities.

The information presented in the user menu and the actions a user is able to perform are aligned with the security roles assigned.

For information on how create the organizational structure, add users and assign security roles, see the *FintechOS Studio User Guide*, section Security.

Password Security

FintechOS users can log into the FintechOS Studio by using FintechOS credentials: username and password. After successfully logging in, users can access the FintechOS resources based on the privileges granted by the security role assigned.

FintechOS Platform has various options in place to ensure password security:

- prevent users from logging in using a wrong password
- set the password to expire
- allow users to recover their password
- set password complexity
- forbid users setting their password matching previous passwords
- forbid users logging in with expired passwords
- lock users who have been inactive for a specific number of days

In order to comply with any password policies that might be enforced within your organization, you can customize the FintechOS Platform password complexity by using server scripting (see section Customize Password Complexity Rules using Server Scripting).

When users will choose to reset their password, an email is sent to the email address associated with their FintechOS Platform account. FintechOS Platform offers a default email template that is used for password reset. It's easy to customize the default email template, or by using server scripting.

If the Forgot Password feature has been activated, users will be able to reset their password from the login page by providing either their emails address or their username.

In addition to the forgot password security, you can also forbid access for users who have been idle for a specific period of time.

Forgot Password

Utilize the FintechOS Identity Provider user interface to enable the forgot password feature for users who cannot recall their credentials and need to reset their password to regain access to their Studio environment.

1. Go to the FintechOS identity provider UI for your environment. From the drop-down, select your realm.
2. Select **Realm settings** and click the **Login** tab.
3. Under **Login screen customization**, set **Forgot Password** to **On** to show a link on login page for user to click when they have forgotten their credentials.

Configure Password Change

FintechOS Platform provides you with various options to configure password change:

- set the period of time (in hours) to pass until users are able to change their password.
- set the period of time (in days) allowed before a password must be changed.
- configure password change based on the password history.

Setting password minimum age

The minimum password age setting determines the period of time (in hours) that a password can be used before the users can change their password.

To set the password minimum age, on the server where the FintechOS Platform installation package resides, add the following:

In **Vault** secrets:

Key Path	Key Name	Key Value
kv/<environment>/<application>/app-settings	core-setting-ebsauth-password-min-age	24

Where **value** is the number of hours until users can change their password.

If **value** is empty or a negative value or the key is missing from **web.config** the **minimum password age** is set to 0 hours allowing immediate password changes, which is not recommended.

When using the minimum password age, we recommend you to configure the password history as well. This way you prevent users to changing their password with the same password.

(Deprecated) Add password minimum age key in web.config files:

```
<add key="core-setting-ebsauth-password-min-
age" value="24"/>
```

Setting password expiry

The maximum password age setting determines the period of time (in days) that a password can be used before the system requires the user to change it.

Key Path	Key Name	Key Value
kv/<environment>/<application>/app-settings	core-setting-ebsauth-password-max-age	30

Where **value** is the number of days allowed before a password expires and should be changed. The maximum number of days is limited to 999. If value is empty, 0 or a negative value or the key is missing, the password expiration feature is disabled, that is, the password never expires, which is not recommended.

If the user tries to authenticate with an expired password the login page will provide the user with the option to reset the password only if the reset password feature is enabled.

(Deprecated) Add password maximum age key in web.config files:

```
<add key="core-setting-ebsauth-password-max-
age" value="30"/>
```

Configuring password change based on password history

FintechOS Platform provides you with the password history features which allows you to set whether a new password is checked against passwords stored in the user's password history. This prevents the user from re-using a recently used password.

To configure the password change to take into consideration user's password history, on the server where the FintechOS Platform installation package resides, go to the **Vault** and add the following secret:

Key Path	Key Name	Key Value
kv/<environment>/<application>/app-settings	core-setting-ebsauth-password-history-depth	5

Where **value** is the number of historical passwords that will be checked when a user tries changing the password. If the user tries to set one of the old passwords then the system will forbid user to use that password. If **value** is empty, 0 or a negative value or the key is missing from the **web.config** file, the password history feature is not enabled (i.e. the user can change the password with the same password).

(Deprecated) Add password change based on password history key in web.config files:

```
<add key="core-setting-ebsauth-password-history-
depth" value="5"/>
```

Setting password about to expire notifications

You might want to remind users that they should change their passwords within x days before their password expired. FintechOS allows you to set such a notification to be shown on a web page and also customize the notification message.

To set the password expiry notification, on the server where the FintechOS Platform installation package resides, go to **Vault** and add the following secret:

Key Path	Key Name	Key Value
kv/<environment>/<application>/app-settings	core-setting-ebsauth-password-about-to-expire-days-until-expiration	30

If the number of days until the password will expire is less than the **value** specified, a page with the remaining days will be shown.

The notification message is localizable, so in order to be properly interpreted by the system, make sure that the text is a json array.

(Deprecated) Add password expiry notificationin the web.config file:

```
<add key="core-setting-ebsauth-password-about-to-expire-days-until-expiration" value="30"/>
```

To customize the notification message ,add the following secret in **Vault**:

Key Path	Key Name	Key Value
kv/<environment>/<application>/app-settings	core-setting-ebsauth-password-about-to-expire-meesage	[{'en-GB':'Password will expire in {10} days.'},{'ro-RO': 'Parola va expira in {10} zile.'}]

When the language is set to Romanian the message will be : "Parola va expira in {10} zile.", where {10} is the number of days until the password will expire.

The Server SDK function `usersAboutToExpirePasswords(int passwordExpireDaysMax)` enables you to get the list of users for which the password will expire in '**passwordExpireDaysMax**' days or less.

(Deprecated) Add customize the notification message key in web.config file:

```
<add key="core-setting-ebsauth-password-about-to-expire-message" value="[{'en-GB':'Password will expire in {10} days.'},{'ro-RO': 'Parola va expira in {10} zile.'}]"/>
```

Skipping the password expiry rule for specific security roles

> **NOTE** To ensure higher security, we recommend you to use this feature only in rare specific cases, e.g., for admin accounts.

To set password never expire for users who have specific security roles, add the following secret:

Key Path	Key Name	Key Value
kv/<environment>/<application>/app-settings	core-setting-ebsauth-password-expired-excepted-role	securityRole

The users with the security role specified in the value will never have to reset the password due to the password expiry rule.

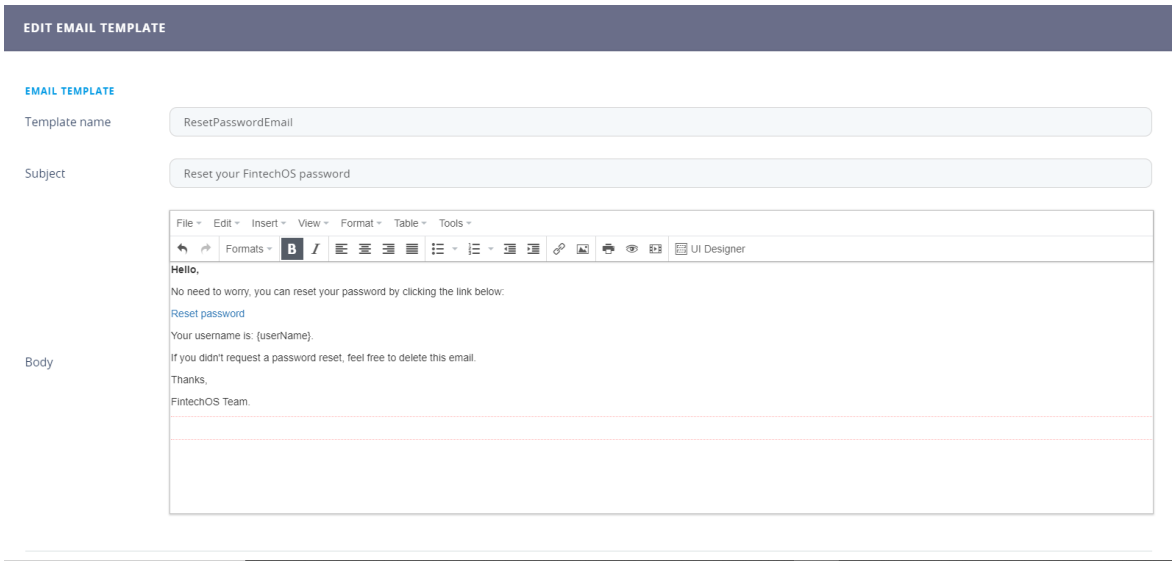
(Deprecated) Add key in web.config to set password never to expire

```
<add key="core-setting-ebsauth-password-expired-excepted-role" value="securityRole"/>
```

Reset Password Global Email Template

The default email template for password reset is named: “ResetPasswordEmail” and it is included in the installation script.

To see the content of the default email template, from the Admin menu, click Email Templates. The Email Templates List page appears. Double-click on the “ResetPasswordEmail” record. The Edit Email Template page will be displayed.



> **NOTE** You can change the content of the default email template based on your preferences, but make sure to include in the template the following tokens: **username** and **generatedToken**, otherwise, the email sent to users will contain incomplete information.

You can also customize the email template by using server scripting. For information on how to do it, see Customize Reset Password Email Template using Server Scripting

Customize Reset Password Email Template

You can customize the reset password email template using server scripting (automation scripts) by following two steps:

Step 1. Add a specific secret in Vault

Add the following secret in Vault file in order to provide the name of the automation script name which customizes the email template.

Key Path	Key Name	Key Value
kv/<environment>/<application>/app-settings	ResetPasswordEmailTemplateWorkflowName	WorkflowName

If you do not provide the name of the automation script for email template customization, the system will search for an on-demand automation server script named “FTOS_ResetPasswordEmail”. For backwards compatibility, the system also searches for ‘ResetPasswordEmail’.

> **NOTE** The FTOS_ResetPasswordEmail” on-demand automation server script does not exist by default; you have to create it.

(Deprecated) Add key in web.config file:

In the **web.config** files, on the server where the FintechOS Platform installation package resides:

```
<app-settings>
        <add
key
=
"ResetPasswordEmailTemplateWorkflowName"
 value="WorkflowName"/>
        ...
</app-settings>
```

Step 2. Create FTOS_ResetPasswordEmail on-demand automation script

The automation script offers customization based on associated user and roles.

The new password reset email template must be returned as the “emailTemplate” key of the Values property:

```
var user = context.Values["user"];
for(var i=0;i<user.Roles.length;i++)
        {
            let role = user.Roles[i];
            if (role.Name == "special")
                {
                context.Values["emailTemplate"] =
"SpecialEmailTemplate";
                break;
                }
        }
```

The user value has the following format:

```
{
    "UserName" : "user1",
    "BusinessUnitId" : "guid",
    "DisplayName" : "user display name",
    "Email" : "user email",
    "ExternalId" : "guid",
    "OrganizationId" : "guid"
    "Roles" :
        [
            {
                "SecurityRoleId" : "guid",
                "Name" : "role name 1"
            },
            {
                "SecurityRoleId" : "guid",
                "Name" : "role name 2"
            },
        ]
}
```

For information on how to create an on-demand server automation scripts, see the *FintechOS Studio User Guide*, section *Creating On-demand Server Automation Scripts*.

Global Password Complexity Settings

For the default Membership provider, the complexity of the password is controlled by the following settings in the **web.config** file:

- minimum required password length
- minimum required non alpha numeric characters
- password strength regular expression

web.config settings for password complexity:

```
<membership defaultProvider="SqlProvider"
    userIsOnlineTimeWindow = "20>
    <providers>
        <add name="CustomMembership"
```

```
        type="EBS.Core.Authentication.Providers.CustomMembership"
        connectionStringName="EbsSqlServer"
        ...
        minRequiredNonalphanumericCharacters="1"
        minRequiredPasswordLength="7"
        passwordStrengthRegularExpression="(?=.*[A-Z].*[A-Z])(?=.*
[#@$*!&amp;])(?=.*[0-9].*[0-9])(?=.*[a-z].*[a-z].*[a-z])"
        />
    </providers>
</membership>
```

You can also customize the password complexity by using server scripting. For more information, see Customize Password Complexity Rules using Server Scripting.

Customize Password Complexity Rules

You can customize the password complexity using server scripting (automation scripts) by following two steps:

Step 1. Add a specific secret in Vault:

Add the following secret in Vault in order to provide the name of the automation script name which configures the password complexity.

Key Path	Key Name	Key Value
kv/<environment>/<application>/app-settings	ResetPasswordRulesWorkflowName	WorkflowName

If you do not provide the name of the automation script for password complexity customization, the system will search for an on-demand automation server script named “FTOS_ResetPasswordRules”.

> **NOTE** The “FTOS_ResetPasswordRules” on-demand automation server script does not exist by default; you have to create it.

(Deprecated) Add key in web.config files:

On the server where the FintechOS Platform installation package resides, add in the **web.config** file:

```
<app-settings>
        <add
key="ResetPasswordRulesWorkflowName" value="WorkflowName"/>
        ...
</app-settings>
```

Step 2. Create FTOS_ResetPasswordRules on-demand automation script

The server automation script offers customization based on password content and associated user and roles.

For information on how create an on-demand server automation script. For information on how to create an on-demand server automation scripts, see the *FintechOS Studio User Guide*, section *Creating On-demand Server Automation Scripts*.

Do not permit passwords containing letter 'z'

```
var password = context.Values["password"];
if (password.match(/z/))
        throwException("Password contains letter z");
```

Context contains two keys in the **Values** property:

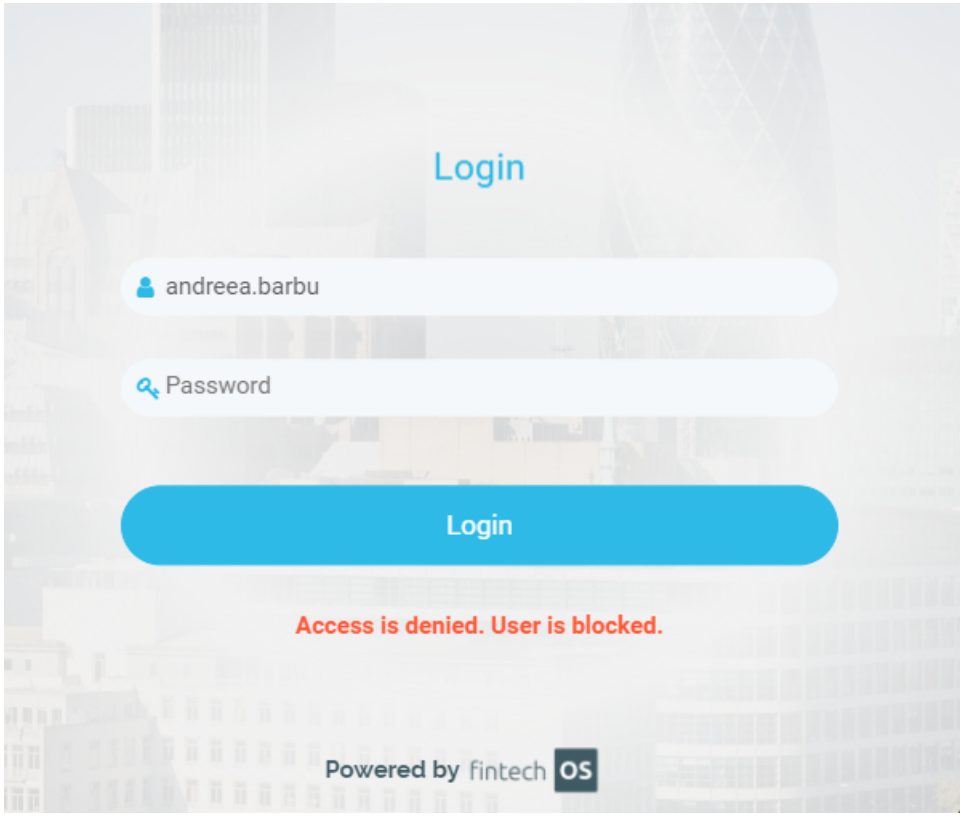
- password
- user, which contains a JSON similar to:

```
{
    "UserName" : "user1",
    "BusinessUnitId" : "guid",
    "DisplayName" : "user display name",
    "Email" : "user email",
    "ExternalId" : "guid",
    "OrganizationId" : "guid"
    "Roles" :
        [
            {
                "SecurityRoleId" : "guid",
                "Name" : "role name 1"
            },
```

```
            {
                "SecurityRoleId" : "guid",
                "Name" : "role name 2"
            },
        ]
}
```

Temporary Blocked User

A temporary blocked user is the account that has opened the FintechOS Portal, inserted the wrong password for that said account for a maximum of five times or for a maximum that was set previously and cannot access the Portal anymore.



When the temporary block time interval has passed and the user wishes to reset the password, click **Forgot password** and follow the steps to receive the email with reset password which implies opening the e-mail and follow the reset password link. The user is unblocked so that reset password flow can be followed.

How to setup the number of retries Portal

In order to create the setup, add the following configuration in Vault:

To configure the the maximum amount of retries (the default value is zero):

Key Path	Key Name	Key Value
kv/<environment>/<application>/app-settings	core-setting-ebsauth-account-lockout-duration	5
kv/<environment>/<application>/app-settings	feature.reset-password	1

When using the EbsAuth provider

For those who are using the EbsAuth provider, to web.config insert an up to date key ("core-setting-ebsauth-account-lockout-duration") to set the amount of minutes the user will not be able to access the account after having tapped in the false password. The key inserted earlier can be zero/ can be a negative value/ empty, then only the administrator has the power to unblock the account for the user.

> **IMPORTANT!**
> The userId present in TemporarilyLockedAccount is deleted when an admin unlocks an account.

The system entity "**TemporarilyLockedAccount**" tracks the modifications happening when an user's account is blocked after having inserted the wrong password.

The feature temporary blocked user has the key set to a positive value, the user wishes to open the FintechOS Portal and **EbsAuth provider** states that the account has been locked after failed attempt to log in, there are two situations:

Firstly, the user was previously temporarily locked out with the current date/ time bigger than the Lockeduntil value, the user will be automatically unlocked, the data from the system entity "**TemporarilyLockedAccount**" is eased and the user will be able to use the Portal. However, when the current date/ time is smaller than the Lockeduntil value, the system will not automatically unlock, but the block will be effective.

Secondly, when the user is not blocked, the LockedUntil value is equal to aspnet_ Membership.LastLockoutDate plus value of "account-lockout-duration". Then, when the current date/ time is bigger than the LockedUntil value, the user's account is

automatically unblocked and the user will be able to use the Portal. Nevertheless, when the current date/ time is equal or smaller than the LockedUntil value, there is an entry in the TemporarilyLockedAccount and the user cannot log in the Portal.

Send Notifications for Locked Accounts or Password Resets

To set up the notifications users receive when their account is locked (after reaching the maximum number of failed login attempts) or when they need to reset their passwords, on the server where the FintechOS platform resides, go to the **web.config** file and add the following settings:

```
<configuration>
    <configSections>
        ...
        <section name="ebsAuthProvider" type
="EBS.Core.Authentication.Common.Configuration.EBSAuthProviderConfi
g, EBS.Core.Authentication.Common"/>
        ...
    </configSections>
    ...
    <ebsAuthProvider>
        <notifications>
            <communicationChannels>
                <channel
name
="myChannel"

channelProvider
="channelProvider" communicationChannel="communicationChannel"/>
            </communicationChannels>
            <notificationTypes>
                <type
enabled
="true"

name
="UserLockedOutOnLastLogin"
 messageTemplate="messageTemplate" from="no-reply@myCompany.com">
                    <supportedChannels>
                        <supportedChannel name="myChannel"/>
                    </supportedChannels>
```

```
                </type>
                <type
enabled
="true"

name
="UserResetPasswordEmail"
 messageTemplate="messageTemplate" from="no-reply@myCompany.com">
                    <supportedChannels>
                        <supportedChannel name="myChannel"/>
                    </supportedChannels>
                </type>
            </notificationTypes>
        </notifications>
    </ebsAuthProvider>
    ...
</configuration>
```

The notifications are set by configuring the `ebsAuthProvider` section with the communication channels and templates used to send the locked account and password reset messages.

communicationChannels

Defines the communication channels available for sending notifications. For each `channel`, you can configure the following settings:

Setting	Description
`name`	Name used to identify the channel used to send notifications.
`channelProvider`	Provider used by the communication channel, such as GatewayEmailOTP or FTOSApiSms. Its value must be the Name of one of the records from **FTOS_DPA_ ChannelProvider** entity.
`communicationChannel`	The type of channel by which the notification will be sent. Its value must be the Name of one of the records from **FTOS_DPA_CommunicationChannel** entity. **IMPORTANT!** Currently, only email and SMS communication channels are supported. More channel types may be added in the future.

Custom email providers

If you wish to use an automation script to send your notifications via a custom email processor, configure the communication channel based on the following model:

```
<communicationChannels>
...
    <channel name="Email_With_
AutomationScript"

channelProvider="CustomEmailProvider" communicationChannel="Email">
        <customProperties>
            <property
name="AutomationScriptName" value="myAutomationScript"/>
        </customProperties>
    </channel>
...
</communicationChannels>
```

Where `myAutomationScript` is the name of the automation script that will process the notification message. The automation script's `context.Data` object will include a data structure called `emailInfo`, which you can use for your custom processing:

```
...
"Data": {
    "emailInfo": {
      "from": "sender@a.com",
      "to": "recipient@b.com",
      "cc": null,
      "bcc": null,
      "body": "email body",
      "subject": "email subject"
    }
}
...
```

notificationTypes

Defines the types of notification that will be sent automatically to the users. For each notification `type`, you can configure the following settings:

Setting	Description
`enabled`	true/false. Activates or deactivates the notification type.

Setting	Description
`name`	• UserLockedOutOnLastLogin - Notify the user after reaching the maximum number of failed login attempts. • UserResetPasswordEmail - Send the user a message with the password reset link.
`messageTemplate`	Content template used for the notification message. For information on how to work with personalized content templates, see the Hyper-Personalization Automation User Guide. Depending on the type of notification, you can insert the following tokens in the content template: • UserLockedOutOnLastLogin - {{user_display_name}} and {{application_name}}. For example: *The user {{user_display_name}} was blocked for {{application_name}}. Too many login attempts.* • UserResetPasswordEmail - {{user_display_name}} and {{password_reset_link}}. For example: *Hello {{user_display_name}}. Use the following link to reset your password: {{password_reset_link}}.*
`from`	Default email sender address or telephone number from which the notification was sent.
`supportedChannels`	Communication channels available for sending the notifications (based on the entries defined in the "communicationChannels" on page 249 section). If the user has a preferred communication channel configured, the notification uses the first matching supported channel. If there is no such supported channel, the first supported channel that is enabled is used instead.

Random Character Password Authentication

This method of authentication does not require the full password, only random characters are typed in by the user. This is done in order to mitigate potential "person in the middle" type of cyber attacks. The number of asked random characters is 3. For

example, if the password is "MyPassword" the user might be asked to provide the chars on positions 1,3,6 ('M','P','s'). The positions (indexes) asked are different on each attempt. The number of failed log-ins that block the user is 5. To support this, a new column was added to `EbsMetadata.SystemUser`, called `PartialPass`.This is populated by a JSON with the details necessary to validate the random character login.

Add the following setting in Vault, under `<app-settings>` to enable the feature:

- ebsauth-partial-password to true (default this is false),
- within the keys that have the following structure:

Key Path	Key Name	Key Value
kv/<environment>/<application>/app-settings	core-setting-ebsauth-partial-password	true

(Deprecated) Add key in the **web.config** file:

```
<app-settings>
...
<add key="core-setting-ebsauth-partial-password"
value="true"/>
...
</app-settings>
```

Architecture

1 Capture the Username

In order to determine the password identity (such as the password length), the username is captured firstly. Once the identity is set, random characters can be extracted from the password.

2 Generate the random characters

When the user is asked to input random characters, they are from the entire range of the password, and not just the minimum required length.

Unauthorize Inactive Users

The company's security policies might require that users who have been idle for a specific number of days are forbidden access to the company's resources.

FintechOS Platform provides you with two SDK functions which enable you to identify any inactive FintechOS Platform users and disable their access as an extra security measure for protecting your FintechOS Platform resources against unauthorized access:

- `inactiveUsers(int daysOfInactivity)` - get the list of users who have not been active in FintechOS Platform in the last number of days specified by the **daysOfInactivity** parameter.
- `unauthorizeUser(string userName)` - makes the user who has the username specified by the **userName** parameter not authorized.

Session Expiration Time

Each session is timed to a specific interval during which if the user presents no inactivity, the Studio/Portal will expire. To set the session timing, the key core-setting-tokenExpiresIn is used and it functions with a specific time syntax. The availability time frame when working inside the Studio and Portal is set using the following syntax d/day/days m/min/minutes h/hour/hours s/sec/seconds. For example, it is possible to set:

- 3 d 5 h
- 3 days 5 hours 3 minutes 20 seconds
- 3 d/days 5 h 3 m/min 20 s/sec

The default value is 20 minutes.

The necessarily changes are made in the **web.config**. If core-setting-tokenExpiresIn is not found in the config, the legacy appSetting TokenExpiresIn is loaded.

Example:

How to set the time for when the Portal/Studio should log out the user in Vault:

Key Path	Key Name	Key Value
kv/<environment>/<application>/app-settings	core-setting-tokenExpiresIn	2d 12h 3m 5s
kv/<environment>/<application>/app-settings	core-setting-tokenExpiresIn	600
kv/<environment>/<application>/app-settings	TokenExpiresIn	1200
kv/<environment>/<application>/app-settings	TokenExpiresIn	1h30m

How to set the time for when the Portal/Studio should log out the user in web.config :

```
<add key="core-setting-tokenExpiresIn" value="2d 12h 3m
5s"/>
<add key="core-setting-tokenExpiresIn" value="600"/> <!--
seconds-->
<add key="TokenExpiresIn" value="1200" />
<add key="TokenExpiresIn" value="1h30min" />
```

OTP Login Session

The OTP login session expiry time can be configure in the **web.config** file. It is done as follows:

```
<multiFactorAuthentication
xmlns="http://fintechos.com/ebs/schemas/multiFactorAuthentication"
enabled="true" otpTimeout="120">
```

The otpTimeout attribute is configured in seconds. The default value is 300 seconds. If a negative value is inserted, then it defaults to 300 seconds.

File Upload Malware Scanning

To configure anti-malware scanning for file uploads, add the following secrets in the "Configuration Manager" on page 46:

Key Path	Key Name	Key Value
kv/<environment>/<application>/app-settings	feature-upload-malware-detection	1
kv/<environment>/<application>/app-settings	feature-upload-malware-use-remote	1
kv/<environment>/<application>/app-settings	feature-upload-malware-endpoint	API endpoint
kv/<environment>/<application>/app-settings	feature-upload-malware-apikey	subscription-key
kv/<environment>/<application>/app-settings	feature-upload-malware-timeout	30

Key	Description
`feature-upload-malware-detection`	Set to 1 to enable anti-malware scanning or 0 to disable it.
`feature-upload-malware-use-remote`	Legacy setting. Always set it to 1 (0 was for a local anti-malware engine available only on legacy on-premise deployments).
`feature-upload-malware-endpoint`	API endpoint of the scan engine.
`feature-upload-malware-apikey`	Subscription key for the scan engine.
`feature-upload-malware-timeout`	Duration in minutes to wait for a response from the scan engine before rejecting the file. Default: 30.

(Deprecated) Add keys in **web.config** files:

```
<add key="feature-upload-malware-detection" value="1"/>

<add key="feature-upload-malware-use-remote" value="1"/>

<add key="feature-upload-malware-endpoint" value="API
endpoint"/>
<add key="feature-upload-malware-
apikey" value="subscription-key"/>

<add key="feature-upload-malware-timeout" value="30"/>
```

Log Management

A log is a record of the events occurring within an organization's systems and networks. Logs are composed of log entries. Each entry contains information related to a specific event that has occurred within a system or network.

FintechOS logs contain records related to computer security (which are generated by sources such as antivirus software, firewalls, and intrusion detection and prevention systems), as well as logs generated by operating systems (on servers, workstations, and networking equipment) and applications.

Security Logs

Irrespective of the device or application, it is imperative that log data has accurate time stamps. In FintechOS, logs are generated at the following levels:

Levels:

1. **Applications**

   Applications log their activity in correlation with the business processes they support, particularly those operations that modify permissions or access rights.

   These logs generally include:

   - The business operation that was requested.
   - Whether the request was accepted or denied.
   - The time and date the operation was performed (Start and end times may be appropriate for long operations).
   - Who initiated the operation.
   - System and network resources used.
   - Any information needed for business process controls.
   - Client hardware and software characteristics.

2. **Systems**

   Many components of the IT infrastructure generate logs. Examples of these components include:

   - Operating Systems.
   - Web servers.
   - Database servers.
   - Print servers.
   - File servers.
   - Authentication servers.
   - DHCP servers.

- DNS servers.
- E-mail server logs.

3. **Network devices**

   Many components of the network infrastructure generate logs. Examples of these components include:

   - Routers.
   - Switches.
   - Wireless access points.
   - Network-based firewalls, intrusion detection and prevention systems, next-generation firewalls.

   These logs typically describe flows of information through the network, but not the individual packets contained in that flow. Information logged for a flow should include:

   - Network (IP) addresses or telephone numbers of the end points.
   - Service identifiers (port numbers) for each of the end points.
   - Whether the flow was accepted or denied.
   - Date, time, and duration of the flow.
   - Number of packets and bytes used by the flow.

4. **Microsoft Azure**

   Three types of platform logs are generated which record the following types of actions:

   - Azure Active Directory Reports – detailed changes made in Azure AD and login activity.
   - Activity logs – record operations performed on an Azure resource – such as creating a VM.

- Resource logs – capture operations performed within an Azure resource, such as querying a database or writing to a storage bucket.

Both Activity Logs and Resource Logs use the Log Analytics workspace to store and access these logs. Also, Log Analytics data is used/analyzed by SIEM to generate alerts/incidents based on defined analytics rules.

Operating System Logs and Application Logs

FintechOS operating system logs and application logs typically hold a variety of information, including computer security-related data.

Logs

1. **Operating System logs**

   Operating systems (OS) for servers, workstations, and networking devices (e.g., routers, switches) usually log a variety of information related to security. FintechOS is logging the most common types of security-related OS data, as follows:

   - System Events - System events are operational actions performed by OS components, such as shutting down the system or starting a service. Each event is timestamped and other supporting information may include event, status, and error codes; service name; and user or system account associated with an event.
   - Audit Records - Audit records hold security event information such as successful and failed authentication attempts, file accesses, security policy changes, account changes (e.g. account creation and deletion, account privilege assignment), and use of privileges.

2. **Applications/platform logs**

Logging level can take one of the following values: Verbose, Debug, Information, Warning, Error, Fatal. A minimum level of logging for each logging sink can be set.

FintechOS is actively using two logging sinks which can be enabled together or separately: file logging and Application Insights logging. We also have SEQ logging already added to the platform and can add more Serilog sinks in future FintechOS releases (based on request).

Type of logs that we expect to meet:

- Platform
    - Exceptions in the execution of the FTOS platform logic.
    - Warning/Info messages added by the platform developers to have a better view on what happened in a certain workflow.
    - Errors where the exceptions are handled, and information is provided about the cause on why a specific logic failed.
    - Digital developers can decide to add their own logging with the desired log level in the automation scripts written for client specific implementations.
    - Items related to observability on the application performance/statistics are not handled at the platform level and should be extracted using, for example, AppInsights out-of-the-box capabilities.
    - Trace.log files configured by default with periodical switching to new files of logs. Trace log files are grabbed by external sources. (SIEM, log server).
- Database

  There are four types of logging events which can be stored in the DB:

- "FintechOS Platform API Logging" on page 269
  - Optional setting – based on customer requirements
  - Can be filtered by source type (i.e. OpenApi, Data Service) and method name
  - Logs are stored in EbsLogs.ApiLog
  - Attention needs to be paid to filtering and cleaning the logging table, as the high number of requests can increase the DB size considerably
  - Starting with 22.1.0 this can also be sent directly to the same logging sink as platform logging (no filtering on security, it is all or nothing)
- CRUD Operations Logging
  - Optional setting – based on customer requirements
  - It can only be turned on and off without any other granularity settings – logging the CRUD events on all entities
  - Logs stored in EbsLogs.UniversalLog
  - Starting with 22.1.0 this can also be sent directly to the same logging sink as platform logging (no filtering on security, it is all or nothing)
- Authentication Events logging
  - Starting with 24.1, go to the Identity Provider to enable logging user login data.
  - Data is stored in the Event_Entity table in the IDP database.
- "Data Audit" on page 267

  Optional setting which by default is turned off

> **NOTE**
>
> FintechOS is logging all CRUD and API operations executed in the platform, by default, in a separate database schema named EbsLogs. Database administrators can restrict read access for this schema and grant insert rights only for the SQL login used by the FintechOS platform. Granular rights can be set up.

3. **DCS/DCI (Digital Cloud Services and Digital Cloud Infrastructure)**

   Logs generated by integration with FTOS technology partners hold data needed for solution debugging, sometimes files sent to the partners and, in some cases, initial requests and responses. Logs are stored on virtual machines where solutions reside. Restricted access to these virtual machines is enforced.

Enable Telemetry

1. In **Configuration Manager** on your environment, go to **Portal > App Settings**.
2. Configure the `azure-appinsights-logging` key and add the following: `"enabled=1; apiKey=[apikey]; sdkLogLevel=Information; logLevel=Warning; flushInterval=10s;"`
   - `sdkLogLevel` - the logging level for the ftos.logging.log server SDK method;
   - `logLevel` - the level for logs generated by the platform, for example when a job in the job server starts or ends.
3. Enable the `azure-appinsights-logging` key.
4. Configure the `azure-appinsights-telemetry-logging` key, and add one of the following values:

- `0` - telemetry data is not recorded;
- 1 - data on incoming HTTP requests received by the Portal/Studio;
- 2 - records detailed information on dependencies, such as SQL queries, HTTP calls to other systems, the time it took for each server-side SDK method to execute, etc.).

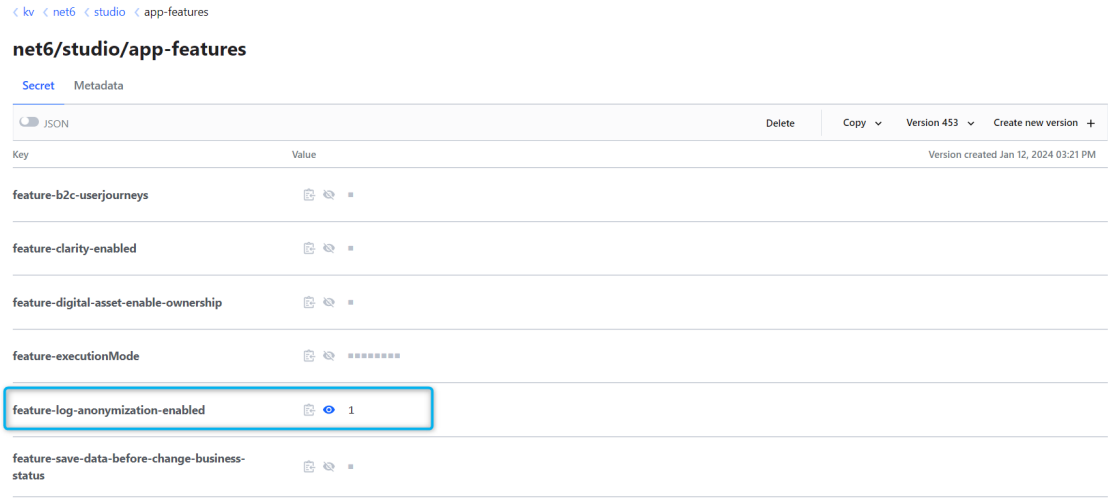
Anonymize Logs

The log entries that contain sensitive information, such as usernames, passwords, credit card numbers, personal identification numbers, can be anonymized in the platform, thus assuring compliance with GDPR standards. You can anonymize the properties when a JSON object or array is logged using the log SDK method. The sensitive information is replaced with `N/A` in anonymized logs.

Enable or Disable Log Anonymization

Log anonymization is enabled by default on environments, and you can check in the Configuration Manager.

1. Go to **App Features** in **Configuration Manager** on your environment.
2. Add the value `1` to the `feature-log-anonymization-enabled` to enable anonymization. Add `0` to disable it.



Define Logs to be Anonymized

Once the log anonymization feature was enabled in **Configuration Manager**, you must now define which logs to be anonymized.

1. In FintechOS Studio, go to System Parameters.
2. Double click on `sys-log-anonymization-settings` to open this system parameter.
3. In the **Paramenter Value** field, add the logs to be anonymized, eg.: username, password, credit card number. Make sure to separate parameters with a comma.
4. Click **Save and close** when done.

> **NOTE**
> The message can be string, object, or json format.

Example

In this example, we are anonymizing multiple customer emails and first names using a script that contains an array with 2 objects. In this scenario, Log anonymization is already enabled.

1. Edit the `sys-log-anonymization-settings` and add `email, firstName` in the **Parameter Value** field.
2. In Studio, create a new server script with an array with 2 objects:

```
example1 = new Object();
example.email = "example1@mail.com";
example.firstName = "Name";

example2 = new Object();
example.email = "example2@mail.com";
example.firstName = "First";

log([example1, example2]);
```

3. Create an endpoint for the script.

4. Open the console from the browser and execute the script: `ebs.callActionByName("example_script")`.
5. Check the logs in App Insights and notice that the elements are anonymized.

You can anonymize multiple customer emails and first names using a script that contains 2 elements in Json:

1. Edit the `sys-log-anonymization-settings` and add `email, firstName` in the **Parameter Value** field.
2. In Studio, create a new server script that has 2 elements in Json:

```
log([{"email":"example1@mail.com", "firstName" =
"Name"}, {"email":"example2@mail.com", "firstName" =
"First"}]))
```

3. Create an endpoint for the script.
4. Open the console from the browser and execute the script: `ebs.callActionByName("example_script", (res)=>console.log(res))`.
5. Check the logs in App Insights and notice that the elements are anonymized.

Filter SDK logs

You can set a different level for SDK logs so that you can view certain logs on development environments. In this way, you can differentiate between logs at platform level and logs from server automation scripts. You can filter logs by automation script, digital asset, machine name, user ID, correlation ID.

Enable Filters for Logs

Follow the steps below to enable or disable the filters for logs:

1. In Configuration Manager, go to **Studio > App Settings**.
2. Add `sdkLogLevel: Info;` to the `azure-appinsights-logging` entry. You can change the value from Info to any other log level that you need, such as `Debug` or `Error`.

The log levels are the same as log levels for Microsoft Azure.

Define Filters

You need to add some settings for the logs. Follow the steps below to define which property you want to filter out:

1. In Configuration Manager, go to **Studio > App Settings**.
2. Edit `sdk-log-filters` to add `{"automationscript":["automation_script_name"], "digital-asset":["digital_asset_name"]}`. Keep in mind that by adding more than one property, the filtering method is either/or.
3. Changes will take effect without restarting, in no more than 20 seconds.

> **NOTE**
> Automation scripts part of digital assets are displayed if the digital asset is specified in `sdk-log-filters`.

Example

In the following example we're filtering by two automation scripts, `test_script_1` and `test_script_2`, and one digital asset, named `test_da`.

1. In Configuration Manager, go to **Studio > App Settings**.
2. Edit `sdk-log-filters` to add `sdk-log-filters: {"automationscript":["test_script_1", "test_script_2"], "digital-asset":["test_da"]}`.
3. Open Studio and run the following commands from the Console, one-by-one:

```
ebs.callActionByName('test_script_1')
ebs.callActionByName('test_script_2')
ebs.callActionByName('test_da')
```

4. Check the logs from App Insights. The scripts and the digital asset are displayed in the logs.

Data Audit

The forth pillar of FintechOS Platform security, logging, provides you with comprehensive audit trail of what happened at any given time and who performed the action.

The logging configuration is specified within the **web.config** file. The platform uses the log.NET component for logging and it generates a **trace_roll.log** file and multiple **trace_roll.dd-mm-yyyy.n.log** files. The log files are saved in the web directory.

NOTE FintechOS API logs and FintechOS LOGS are kept in different audit tables.

Entity Audit

FintechOS Platform has an extensive audit functionality that can be enabled for any entity, allowing change tracking at entity level.

Using the FintechOS Studio, users can activate the auditing feature for a specific entity, by selecting the **Is Audited** checkbox. When auditing is enabled, the platform creates and maintains a system entity named **{entityName}_ADT** where all changes to the initial entity are recorded including: the type of changes on the entity, when the changes have been made and by whom.

When the user navigates to the list view of an entity with audit enabled the **History** button will be available on the toolbar.

Clicking the **History** button will open the History List view which lists all the changes associated to the current entity instance (the associated ADT entity).

When navigating to the detail view for an audited entity, the **History** button will open a list with all the changes associated to the current entity instance.

To programmatically navigate to the audit logs use the commands below:

To get all audit logs for the specified entity (where, the ID is the entity ID):

```
'entity/{entityName}/history/viewAll/{id}'
```

To get audit logs for the specified operation:

```
'entity/{entityName}/history/{operation}'
```

To get audit logs for two specific operations:

```
'entity/{entityName}/history/{opOne}/{opTwo}'
'entity/{entityName}/businessTransactions/{id}'
```

The data audit is independent of entity records (when the **Audit enabled** checkbox is selected on entity). An unique identifier (UID) is automatically added by the system to records. When users delete records, based on the UID, the action is logged into the audit trail.

The History List view which lists all the changes associated to the current entity instance (the associated ADT entity) has a new column, Unique Identifier (UID).

If the user deletes an income of a customer. the action is logged into **{entityName}_ADT**. The user can consult anytime the History List page on that customer entity and see that the income has been has been deleted, when and by whom.

FintechOS Platform Logging

FintechOS Platform logs all CRUD operations executed in the platform, by default, in a separate database schema named EbsLogs.UniversalLog.

Database administrators can restrict read access for this schema and grant insert rights only for the SQL login used by the FintechOS Platform.

How to Configure the Logging of CRUD Operations

To configure this feature, go in **Vault** and set the feature-universal-logging setting, as desired. By default, it is set to **0**, that is, the feature is disabed.

In **Vault** secrets:

Key Path	Key Name	Key Value
kv/<environment>/<application>/app-settings	feature-universal-logging	0|1|false|true

(Deprecated) Add key in the **web.config** files:

```
<configuration>
      <app-settings>
                  ...
                  <add key="feature-universal-
logging" value="0|1|false|true"/>
      </app-settings>
</configuration>
```

FintechOS Platform API Logging

The FintechOS Platform can log calls over the FintechOS Platform API (REST and WCF) and DataService CRUD operations.

Where Are the API Logs Stored?

The log entries are stored based on the platform's "Observability" on page 68 settings (e.g. system console, local file storage, Seq structured log server, or Azure Application Insights service). API log entries can be identified by their **event-issuer** log context property which is set to **InboundRequests**.

What API Information is Logged?

The following "Logging context" on page 73 properties can be tracked:

Log Context Property	Description	Example
base-user-id	Authenticated user ID (for system users, not external users).	4afdc8a9-eb91-4359-81d6-c3a462fae866
CorrelationId	GUID which correlates the initial call from one of the internal FintechOS Platform components or from an external client application with the subsequent calls triggered between FintechOS Platform components.	420e4312-aa09-4d3e-8cfb-32aca1b54694
RequestId	Request ID	f9ab337a-57d0-4cb2-b0a6-1c51ea72b45c
source-type	Type of API call • OpenApi • ApiService • DataService • Account	DataService
method-name	API method name	CallAction
client-ip	Client IP	192.168.30.18

Log Context Property	Description	Example

request	API request parameter as JSON	{
		"ActionId": "b2063100-ee43-4d3c-b891-5ace19978c64",
		"ActionName": "TestScript",
		"EntityName": "workflow",
		"IsList": true,
		"EntityIds": [
		"f69b8eee-4684-4fa6-93f5-536f9145c8af",
		"a55d3165-e6c0-4545-8734-58124dc263da",
		"e5eb71d1-3dc4-4bbe-96c4-92887df3611c"
		],
		"Data": null,
		"Values": []
		}

exception	Response error	
success	• True • False	True
call-duration	Call duration in milliseconds	110.5123

Log Context Property	Description	Example
response-content	Content of the response	{ "UIResult": null, "Message": null, "IsSuccess": true, "ClientScript": null, "Serialized": "{\"UIResult\":null,\"Message\":null,\"IsSuccess\":true,\"ClientScript\":null,\"Serialized\":null,\"ErrorCode\":0}", "ErrorCode": 0 }
event-issuer	Always set to **InboundRequests** and is used to filter the API entry logs from other logging events.	**InboundRequests**
response-message	Response message, sub-property for response-content.	
request-headers	List of headers (with security related headers like AccessToken removed).	

Severity

The severity level of the log entries is set by default to **Information**. In case of an error (e.g.: the `exception` log context property is not null), it will be set to **Error**.

How to Configure API Logging

To configure API logging, in the "Configuration Manager" on page 46, configure the `kv/<environment name>/<portal instance>/app-settings/ApiLoggingConfig` key as shown below:

```
{
```

```
    "sources": [
        {
            "name": "DataService",
            "methods": [
                {
                    "name": "CallAction"
                },
                {
                    "name": "GetById",
                    "input": {
                        "properties": [
                            {
                                "name": "A",
                                "excluded": true
                            }
                        ]
                    }
                }
            ]
        }
    ]
}
```

Source names:

- OpenApi (REST endpoint)
- ApiService (WCF endpoint)
- DataService (MVC endpoint)

The configuration allows filtering the API logging elements at different levels of granularity: source, method (action), input (request), output (result), input property, output property.

The user can enable logging for all methods of a source by specifying "*" as the method name. Any explicitly defined method will override the full logging from "*".

> **NOTE**
> When a property is excluded, it will not be serialized in the log.

Enable API Logging

Once you have set up API logging, in the "Configuration Manager" on page 46, set the `kv/<environment name>/<portal instance>/feature-log-api-inbound` key to **1** to enable it (setting the key to 0 disables API logging).

> **IMPORTANT!**
> Setting the `feature-log-api-inbound` key value to **true** instead of **1** will not work.