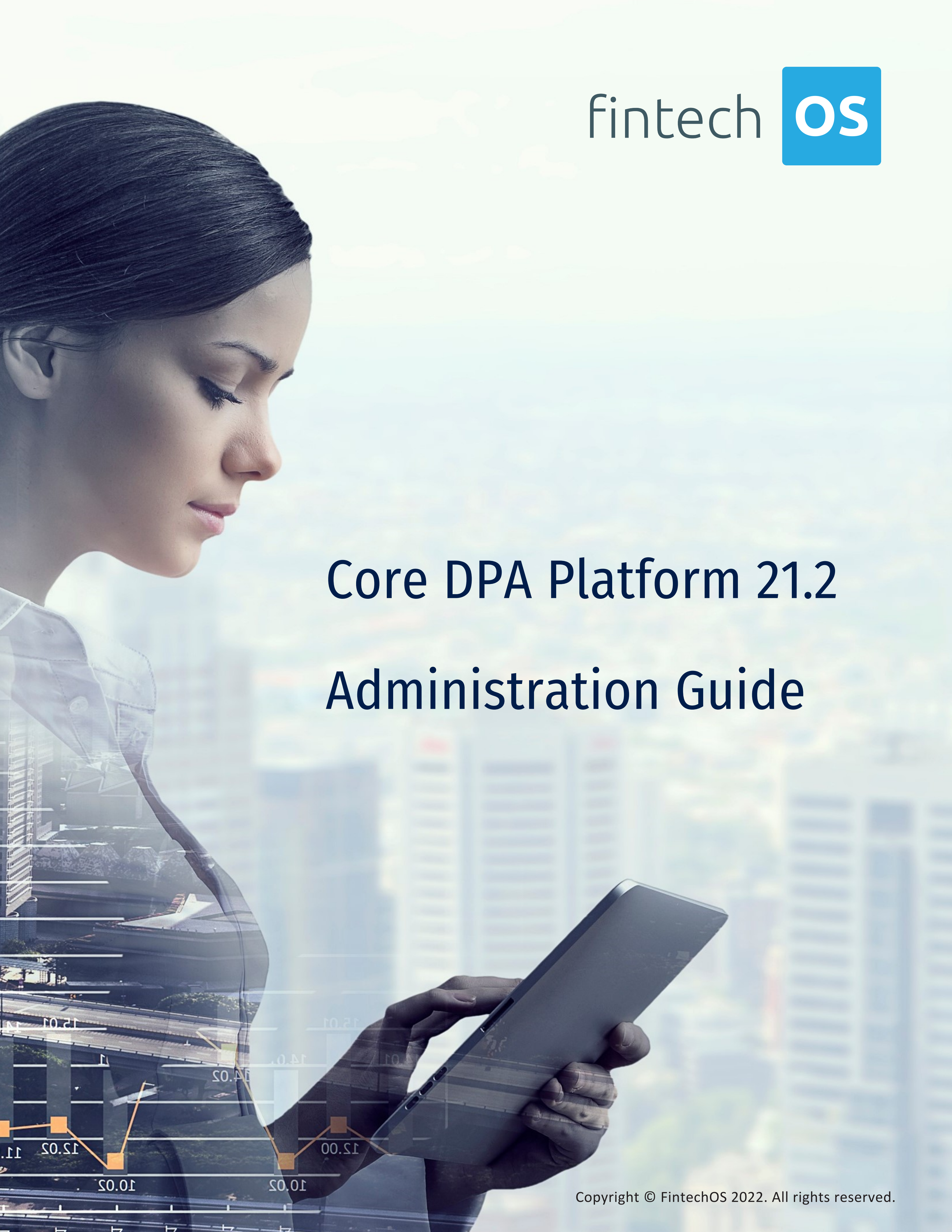


Core DPA Platform 21.2

Administration Guide

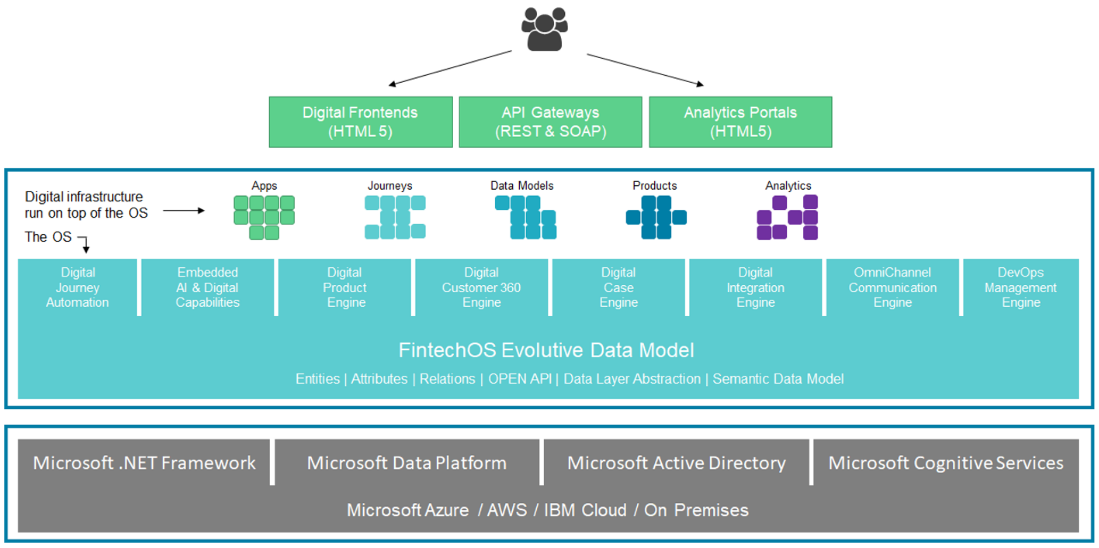
TOC

Overview

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

FintechOS is a highly scalable technology that can be run both on premises and from the cloud.

While deployment on premises are still permitted with the current version of FintechOS we recommend using one of the enterprise cloud providers that FintechOS is compatible with. Below you can see a more detailed technological architecture on how FintechOS runs based on Microsoft, AWS or IBM.


Digital Frontends (HTML 5)
API Gateways (REST & SOAP)
Analytics Portals (HTML5)
Digital infrastructure run on top of the OS
The OS
Apps
Journeys
Data Models
Products
Analytics
Digital Journey Automation
Embedded AI & Digital Capabilities
Digital Product Engine
Digital Customer 360 Engine
Digital Case Engine
Digital Integration Engine
OmniChannel Communication Engine
DevOps Management Engine
FintechOS Evolutive Data Model
Entities | Attributes | Relations | OPEN API | Data Layer Abstraction | Semantic Data Model
Microsoft .NET Framework
Microsoft Data Platform
Microsoft Active Directory
Microsoft Cognitive Services
Microsoft Azure / AWS / IBM Cloud / On Premises

Installation

This section describes how to install the FintechOS Platform.

System Requirements

- "FintechOS Platform System Requirements" below
- "FintechOS Portal System Requirements" on page 12
- "System Requirements for WebRTC Components" on page 13

FintechOS Platform System Requirements

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

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!

NOTE
Starting with 21.2, Internet Explorer is no longer supported.

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 Core DPA Platform uses the following cookies:

Server-Side Encrypted Cookies

These are security cookies used for client-server authentication.

Cookie	Description
.EBSCORE$1	Authentication cookie.
.EBSCORE$1-CSRFToken	CSRF prevention token cookie.
.EBSCORE$1_ PartialToken	Used for partial password authentication (only available with FintechOS Identity Provider).

Client-Side Unencrypted Cookies

These are client-side generated cookies that store a transient state, therefore they do not require encryption.

Cookie	Description
.EBSCORE$1-CorrelationId	Browser session specific cookie used for logging.
.EBSCORE$1-culture	Current language. Preserves language settings between user sessions.
.EBSCORE$1-hasCollapsedHeader	FintechOS Portal header settings.
.EBSCORE$1-ShowTooltipsOnForms	Show/hide tooltips in FintechOS Portal forms.
.EBSCORE$1-palette	Color palette settings for FintechOS Portal.
.EBSCORE$1-theme	Theme settings for FintechOS Portal.
.EBSCORE$1-timezone	User agent's timezone (moment.js client library).
.EBSCORE$1-appBoundaryInteractiveTutorial	Display first-time tutorial in the user interface.

Automatic On-Premise Installation

This page describes the on-premise deployment process of FintechOS (such as Innovation Studio, Portal, B2C), along with requirements and how to install them, as well as SysPackages installation.

The process described on this page uses automatic installers to deploy both the requirements and the platform. This approach is recommended for fresh deployments of Innovation Studio.

NOTE
This process can only be used for on-prem installation.

The deployment package is used to deliver the following components:

- Local Windows Prerequisites
- Database Configuration on existing SQL Server
- FintechOS Portal Installation
- Innovation Studio Installation
- FintechOS B2C Portal Installation

- FintechOS B2C Proxy Installation
- FintechOS JobServer Installation
- SysPackages Import on existing Innovation Studio instance

Installation Prerequisites

In order to perform a successful installation, the following are required:

- Administrator Privileges
- An SQL Server instance (SQL Express is also accepted) on localhost or an Azure SQL Server.
- The .zip file containing FintechOS Installation files
- The .zip containing the HPFI version you want to install (such as: **FTOS-CORE-RLS-V21.2.1.0-B87-GOLD.zip**for the FintechOS 21.1.0 release).

IMPORTANT!
Make sure to extract the contents of the .zip file with the FintechOS installation files, because you need to navigate to this location (such as: C:\FintechOS_install).

Installation Process

1 Parameter Configuration

Before the installation process can be started, the general and database parameters need to be configured. This is done by editing the **parameters.ini** file with an editor (such as Notepad++), and filling in the values for the attributes below. The path and project are configured in the **[general]** section, and the database parameters in the **[database]** section.

The [general] section

Attribute	Description
APPLICATION	The project name that is included in the IIS App Pool and IIS Application for Portal, Studio, B2CPortal, B2CProxy.
INSTALLATIONPATH	The local path where the FintechOS components are installed. (such as: C:\FintechOS\Studio)
KIT	The folder where the Core DPA Platform .zip file (such as: **FTOS-CORE-RLS-V21.2.0.0-B87-GOLD.zip**) is located.

The [database] section

Attribute	Description
REMOTE	• Set to **NO** to use a local SQL server. • Set to **YES** to use a remote SQL Server (such as: Azure SQL).
SQLSERVER	• Use `localhost` if **REMOTE** is set to **NO** (use local SQL Server). • Use your SQL Server address (example: `myxyzserver.database.windows.net`) if **REMOTE** is set to **YES** (remote SQL Server).
DATABASE	The name of the database created for FintechOS.
USER	The user with which the FintechOS instance connects to the specified database. If the user does not exist, a new one is created.
PASSWORD	The password associated to the **USER** field. If the user exists, please provide the corresponding valid password. Make sure to use at least 8 characters, a capital letter, and a symbol.

IMPORTANT!
Make sure to save the **parameters.ini** file after all the changes are made.

2 Component Installation

Once the parameters are properly configured in the **parameters.ini** file, the components need to be installed. This is done by running the a PowerShell instance with administrator privileges and running the corresponding scripts for the desired component.

To do this, open PowerShel with administrator privileges, navigate to the location where you extracted the installation scripts (such as: C:\FintechOS_install), then run the scripts as follows:

Component	Script	Description
Prerequisites Installation	`1_install_prerequisites.ps1`	Searches for HPFI prerequisites (such as .NET Framework) and installs them if necessary.
Database Configuration	`02_configure_database.ps1`	Creates the database and the user which logs in, according to the **DATABASE**, **USER**, and **PASSWORD** configurations in the **parameters.ini** file.
Portal Installation	`03_install_portal.ps1`	Installs the FintechOS Portal in the location specified in **INSTALLATIONPATH**.
Studio Installation	`04_install_studio.ps1`	Installs the Innovation Studio in the location specified in **INSTALLATIONPATH**.
JobServer Installation	`05_install_jobservices.ps1`	Installs the FintechOS JobServer components (such as Jobserver, MessageBus, MessageComposer).
B2C Portal Installation	`06_install_b2cportal.ps1`	Installs the FintechOS B2C Portal component required for the Business to Client interface. Note that this component is optional and is not required for FintechOS unless you want a Business to Client interface.
B2C Proxy Installation	`07_install_b2cproxy.ps1`	Optional. Installs the FintechOS B2C Proxy component required for the Business to Client interface.

Once you run a script, wait for the component to get installed and a success message is displayed.

3 SysPackage Installation

In order to install a SysPackage, the appropriate values need to be added to the **parameters.ini** file, as well as the corresponding executable to be launched.

Configuring the Parameters

Open the **parameters.ini** file with an editor such as Notepad++ and navigate to the **[syspacks]** section. The following attributes need to be filled in accordingly:

Attribute	Description
SYSPACK	The location of the FintechOS SysPackage .zip file (example: C:\FtosInstall\SysPackages\Professional-Banking-SysPacks-v21.2.6001).
URLSTUDIO	The URL of your Innovation Studio instance.

IMPORTANT!
1. You must run the script on the machine where Innovation Studiois installed.
2. Make sure you know the **user** and the **password** for Innovation Studio.
3. Make sure you have access rights to Studio's database.
4. Make sure to save the **parameters.ini** file after all the changes are made.

Installing the SysPackage

Once you configure the values in the **parameters.ini** file, install the SysPackage as follows:

1. Open a PowerShell instance with administrator privileges and navigate to the folder where the **install_FintechOS.zip** file was extracted.
2. Run the `08_install_syspacks.ps1` script.
3. Insert the user and password provided in the corresponding fields in the **parameters.ini** file. Once the values are provided, the SysPackage is installed.

Upgrade Process

This section guides you through the steps required to upgrade the Core DPA Platform.

Prerequisites

- The .zip file containing the Core DPA Platform version you want to upgrade to. For example, for Core DPA Platform version 21.2.0, should look like `FTOS-CORE-RLS-v21.2.0.0-b87-GOLD.zip`. It also contains the installation script files inside the **install_FintechOS.zip** file.
- The Deployment Package which contains your existing data. Read more about Configuring Data Deployment Packages.

Upgrade Steps

1 Preparing The Data Deployment Package

The first thing that needs to be done is to create the deployment package which includes all the metadata you need to migrate.

You can create and export a data deployment package as explained on this page.

If, in addition to metadata, you wish to include entity data and report templates in a single deployment package, you need to create an enhanced deployment package.

This generates an .xml file with all the required metadata which needs to be imported after the platform upgrade.

2 Upgrading Portal and Studio

IMPORTANT!
Stop/Deactivate all FTOS services (e.g., stop & switch to manual-start Windows services or Web applications) except the FTOS database.

- Extract the contents of the **install_FintechOS.zip** located in the Core DPA Platform installation .zip.
- Open the parameters.ini file, and modify the INSTALLATIONPATH attribute to match the location where the platform is already installed. Close and save the file.

- Run the **PortalInstaller.exe** and **StudioInstaller.exe** files. They are found in **install_FintechOS.zip**. This upgrades the Portal and Studio components of the platform.

3 Upgrading The Services

MainDbServer

You must execute instructions from this section on a machine that can connect and execute T-SQL against **MainDbServer**.

1. Execute in cmd.exe: `"ReleaseKit\SQL\BasicDbUpgrader.exe" -s "MainDbServer" -d "MainDb"`

NOTE
You should be presented with a report on which scripts will be applied on **MainDb**.

2. Execute in cmd.exe: `"ReleaseKit\SQL\BasicDbUpgrader.exe" -s "MainDbServer" -d "MainDb" -g`

- This command will upgrade **MainDb** and it will take a few minutes to complete, depending mostly on how old **MainDb_OldVersionNo** is vs. **ReleaseKit_VersionNo**

PortalWebApp

You must execute instructions from this section on **PortalWebApp_Machine**.

1. Create a new batch file named **PortalWebAppInstaller.Upgrade.bat** in a directory of your choice and add in the following single command line: `powershell.exe -File "ReleaseKit\PortalWebApp\PortalWebAppInstaller.ps1" -p_MainCommand Upgrade -p_InstallDir PortalWebApp_InstallDir`
2. Execute in cmd.exe: `PortalWebAppInstaller.Upgrade.bat`

HINT
This command does the following:

◦ Overwrites all files from **PortalWebApp_InstallDir** with **PortalWebApp** files from

ReleaseKit, except for web.config which is not overwritten in case you customized it
◦ [Re]Starts **PortalWebApp_IisAppPoolName**

3. Open in a web browser **PortalWebApp_LoginUrl** and check the page appears as expected.

DesignerWebApp

You must execute instructions from this section on **DesignerWebApp_Machine**.

1. Create a new batch file named **DesignerWebAppInstaller.Upgrade.bat** in a directory of your choice and add in the following single command line:
```
powershell.exe -File "
ReleaseKit\DesignerWebApp\DesignerWebAppInstaller.ps1" -p_
MainCommand Upgrade -p_InstallDir DesignerWebApp_InstallDir
```
2. Execute in cmd.exe: `DesignerWebAppInstaller.Upgrade.bat`

HINT
This command does the following:

◦ Overwrites all files from **DesignerWebApp_InstallDir** with **DesignerWebApp** files from **ReleaseKit**, except for web.config which is not overwritten in case you customized it
◦ [Re]Starts **DesignerWebApp_IisAppPoolName**

3. Open in a web browser **DesignerWebApp_LoginUrl** and check the page appears as expected.

JobServer

You must execute instructions from this section on **JobServer_Machine**.

1. Execute in cmd.exe: `sc.exe stop JobServer_WinSvcName`

2. Copy with overwrite all files from **ReleaseKit_Dir\JobServer** to **JobServer_InstallDir** except for the following files:

- connections.config
- FTOS.JobServer.Service.exe.config

- schedule.config
- services.config
- serviceSettings.config

3. For each of the files excepted at the previous step, analyze the differences between version from **ReleaseKit_Dir** and that from **JobServer_InstallDir** using a text file compare tool and merge changes into the version from **JobServer_ InstallDir** without breaking existing customizations

4. Execute in cmd.exe: `sc.exe start JobServer_WinSvcName`

MessageComposer

You must execute instructions from this section on **JobServer_Machine**.

1. Follow all steps from JobServer \ Standard job configuration \ Upgrade
2. Execute in cmd.exe: `sc.exe stop JobServer_WinSvcName`
3. Copy with overwrite all files from **ReleaseKit_ Dir\JobServer.Plugins\MessageComposer** to **JobServer_InstallDir**, except for the following files:

- connections.config
- schedule.config
- services.config
- serviceSettings.config

4. For each of the files excepted at the previous step, analyze the differences between version from **ReleaseKit_Dir** and that from **JobServer_InstallDir** using a text file compare tool and merge changes into the version from **JobServer_ InstallDir** without breaking existing customizations.
5. Execute in cmd.exe: `sc.exe start JobServer_WinSvcName`

4 Importing Deployment Packages

The final step is to import all your metadata in the upgraded environment. This makes sure all your data is migrated and ready to be used. Importing is done by running the **SysPkgDeployer** tool from the command line, using the following syntax:

```
CD /D %~dp0 "%~dp0\FtosPkgDeployer\FtosSysPkgDeployer.exe" -i -a -s
"StudioLink" -u AdminStudioUser -p user_password -z DataBaseServer
-v DB_user -k DB_user_password -d "TheNameOfTheDataBase" -r
"%~dp0\01 DeploymentPackages\*.zip"
pause
```

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 Folder

When building a web application that requires users to upload or download files (documents, images, etc.), file storage can be an important aspect of the application architecture.

Where Should I Store Files?

FintechOS platform supports multiple storage providers for storing the uploaded or generated user files. When building web applications using FintechOS technology, you've got a few choices for where to store your files:

- "Local File System Storage" on the next page
- "Azure Blob Storage" on page 27
- "Amazon S3 Buckets Storage" on page 29

The local file system refers to either a local path on the application server or a shared folder on the network containing the application server. While it is the default storage provider, you might be running out of disk space or you might find it a very challenging task to ensure that files are properly backed up and available at all times.

If you'll be storing large blobs of content, you might want to consider one of the other options. Storing files in a file storage service like Amazon S3 Buckets or Azure Blob is a great option if you'll be storing large blobs of content. Not only you stay rest assured that your data is replicated and backed up, but they also ensure scalability and high availability.

This section walks you through the steps needed to configure the "UploadEbs" storage provider /location as needed.

Local File System Storage

There are no special configurations that have to be made in order to use it other than setting the name of the root folder.

To set the name of the root folder, go to the **web.config** file, open it and to the **appSettings** node, add the application setting **UploadFolder**, as described below:

```
<configuration>
 ...
 <appSettings>
       ...
       <add key="UploadFolder" value="path_to_root_folder" />
  </appSettings>
  </configuration>
```

Depending on where the root folder resides, make sure that you properly set the value of the UploadFolder setting:

- subfolder of the application folder: "~/path/to/uploadfolder/";
- local folder on application server, the full path to local folder, like:

  "c:\path\to\uploadfolder"
- network shared folder: "\\server\path\to\uploadfolder";

NOTE
If in the **web.config** file you do not set the **UploadFolder** setting, it is automatically set to the default value, that is, "~/UploadEBS/".

Automatically Create File Upload Subfolders

IMPORTANT!
This feature is available only for local file system storage. It is not available for Azure Blob Storage or Amazon S3 Buckets Storage.

You can automatically group uploaded files into folders based on the last three characters in their file name (excluding the file extension). To do so, add an **feature-uploadfolder-autocreate-subfolders** key with a value of **1** in the web.config file:

```
<add key="feature-uploadfolder-autocreate-subfolders" value="1">
```

This will save each uploaded file in a `-.files`\xyz subfolder of the upload folder, where *xyz* represents the last three characters of the file name. For example, a file called MyDoc_0caf99b6-549d-48f7-8747-5e3eb82753fd.txt will be saved in a folder structure similar to:

```
...\
    UploadEBS\
        -.files\
            3fd\
                MyDoc_0caf99b6-549d-48f7-8747-5e3eb82753fd.txt
```

Setting the feature-uploadfolder-autocreate-subfolders key value to **0** disables the feature.

This feature is backward compatible. If a requested file is not stored in the above folder structure, it will be read from the main upload folder or the entity specific upload folder respectively.

Azure Blob Storage

To configure FintechOS to store user files in Azure Blob, follow these steps:

1. Go to the web.config file and open it.
2. Add a `ftosStorageService` section to the **<configSections>** element:

```
<configuration>
        <configSections>
            ...
            <section
name
=
"ftosStorageService"

type
="EBS.Core.Utils.Services.Config.StorageServiceConfigSection,
EBS.Core.Utils"/>
        </configSections>
<configuration>
```

3. Add a `ftosStorageService` section (note the AzureBlob type) as child of **<configuration>** element:

```
<configuration>
 ...
 <ftosStorageService type="AzureBlob">
       <settings>
            <setting
name="connectionString" value="connection_string"/>
            <setting name="rootContainer" value="root_
container"/>
       </settings>
  </ftosStorageService>
</configuration>
```

where:

- *connectionString* is the connection string FintechOS is using to connect to an Azure Blob container;
- *rootContainer* is the root container name where the user files will be stored.

Azure Resource Manager templates support

To enable automatic deployment through ARM templates, the `connectionString` and `rootContainer` settings must be configured in the **<appSettings>** element of the web.config file:

```
<appSettings>
...
    <add key="ftosStorageService-AzureBlob-
connectionString" value="connection_string" />
    <add key="ftosStorageService-AzureBlob-
rootContainer" value="root_container" />
</appSettings>
```

IMPORTANT!
Values set in the **<appSettings>** keys take precedence over the values set in the **<ftosStorageService>** settings node.

Amazon S3 Buckets Storage

To configure FintechOS to store user files in Amazon S3 Buckets, follow these steps:

1. Go to the web.config file and open it.
2. To the <configSections> element, add the following two sections: ftosStorageService and aws, as described below:

```
<configuration>
        <configSections>
        ...
            <section
name
=
"ftosStorageService"

type
="EBS.Core.Utils.Services.Config.StorageServiceConfigSection,
EBS.Core.Utils"/>
            <section name="aws" type="Amazon.AWSSection,
AWSSDK.Core"/>
        </configSections>
 <configuration>
```

3. Add <ftosStorageService> tag (note the AmazonS3Bucket type) as child of configuration element:

```
<configuration>
        ...
        <ftosStorageService type="AmazonS3Bucket">
            <settings>
                <setting name="AWSAccessKey" value="access_
key" />
                <setting name="AWSSecretKey" value="secret_
key" />
                <setting name="BucketName" value="bucket_
name"/>
            </settings>
        </ftosStorageService>
</configuration>
```

where:

`AWSAccessKey` and `AWSSecretKey` are used by FTOS to sign the requests made to AWS. For more information, see Access Keys (Access Key ID and Secret Access Key).

BucketName is the root bucket name where the user files will be stored.

4. Add the **aws** section as child of the configuration element:

```
<configuration>
...
        <aws region="aws_region">
        </aws>
</configuration>
```

NOTE
The only required attribute is **region**. For a complete list of available regions, see 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>`

For a list of allowed elements in the AWS section, see Configuration Files Reference for AWS SDK for .NET.

Importing and Exporting Deployment Packages

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

Deployment packages are text-based so they can be version controlled to have their history inspectable with text-diff tools.

In FintechOS, you have three options for importing and exporting deployment packages, as follows:

- In Innovation Studio, from the DevOps menu > Deployment Packages. For more information, see the Innovation Studio, section Deployment Packages.
- Using the customization set methods of the API. For information on how to import and export packages using customization sets), see FintechOS API documentation.
- From the command line by using the **FtosPkgDeployer** tool.

The **FtosPkgDeployer** tool is available in the release subdirectory, *\Tools\FtosPkgDeployer*, 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 **FtosPkgDeployer** tool , see the built-in help by running the command prompt as admin and executing `FtosPkgDeployer.exe` without arguments. The tool is using the POST CUSTOMIZATION SET method of the FintechOS API.

File-Type Upload Control

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

This feature is particularly useful in preventing users from uploading wrong file types, thus saving time from investigating what went wrong 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, on the server where the FintechOS installation package resides, go to the **web.config** file, open it and add the following setting:

```
<appSettings>
    ....
    <add key="feature.upload.filetype-check" value="true" />
</appSettings>
```

File-Type Upload Processing

Once the File-Type Upload control is enabled, upon file uploads using client scripts (using the ebs.upload function) or server automation scripts (using the uploadFile 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.

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 API a Standalone Web App

FintechOS gives you the ability to set the FintechOS 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 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, go to the **web.config** file and enable the API server, as provided below:

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

```
...
</configuration>
```

Activating Localization Debug Mode

FintechOS 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 the **Web.config** file of your application and change the value of the following key:

From:

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

To:

```
<add key="ebs:debugLocalization" value="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.
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.

Configure Notifications for Operations

In the Core DPA Platform it is possible to receive notifications for operations such as scheduled jobs. This is done when options such as **Send Notification On Error** or **Send Notification On Success** are available in Innovation 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 inside the Core DPA Platform installation folder (for example: `C:\FintechOS\MyProject\JobServer`). 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.

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:

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 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 the <appSettings> node of the *web.config* file, add keys for the hub name and endpoint settings of each client application, based on the model below:

```
<appSettings>
    ...
    <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">
</appSettings>
```

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**.

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.
Subject	Message queue subject set in the sendMobileNotifications function call that initiated the notification push.
Body	Message received by the recipient.

Attribute	Description
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 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 message queue, there are going to be 5 entries (one for each user). On the Azure notification hub, after sending the request from FintechOS, 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.

Collect Logging Data in Azure Application Insights

You can use the Azure Application Insights application performance management service to collect logging information independently from the *trace.log* local file. 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 *trace.log* file are visible instantly. Messages sent to Azure Application Insights might be visible after a short delay ranging from seconds to minutes.

Prerequisites

An Azure Application Insights subscription.

Configuration

Azure Application Insights Logging Configuration

To configure Application Insights logging, use the `feature-logging-azure-appinsights` key in the `<appSettings>` node in the *web.config* file:

```
<appSettings>
...
    <add key="feature-logging-azure-appinsights" value="enabled=1;
apiKey=API_KEY; logLevel=Warning; flushInterval=1m">
...
</appSettings>
```

Values	Data Type	Default Value	Description
enabled	boolean	0	Enables or disables Azure Application Insights logging: true/false, on/off, 0/1.
apiKey	guid string	empty	Instrumentation Key from your Azure Application Insights subscription.
logLevel	string	Error	Minimum logging level. Possible values: Fatal, Error, Warning, Information, Debug, Verbose. **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	timespan	1m	Delay between message batches sent to Azure, to be specified in .NET Timespan format or in Jira timespan format: 1h, 1d, 1h30m, 00:00:05, etc.

Azure Application Insights Telemetry Configuration

Telemetry is enabled by default. To disable Azure Application Insights telemetry, use the `feature-logging-azure-appinsights-telemetry-disabled` key in the `<appSettings>` node in the *web.config* file:

```
<appSettings>
...
    <add key="feature-logging-azure-appinsights-telemetry-
disabled" value="1"/>
...
</appSettings>
```

NOTE
Deactivating telemetry does not affect logging.

Collected Data

The following application specific information is saved for each log entry as part of custom dimensions:

- MachineName
- CorrelationId
- Language
- AutomationScript
- AutomationScriptLibrary
- SQL
- ExtraErrorDetails

Severity Levels

Log entries with severity levels Warning, Information, Debug, or Verbose are saved as **traces**.

Log entries with severity levels Exception or Fatal are saved as **exceptions**.

Information Severity Level Examples

- All calls to the log Server SDK function executed from automation scripts.
- Sucess/Error status for http get/post calls to external APIs executed from automation scripts.
- Authentication errors and information for AD and EBS providers.

Warning Severity Level Example

Missing or missmatched configuration.

Error Severity Level Examples

- Runtime exceptions with full stack trace.
- Errors in automation scripts. These errors include the automation script name, line, and column from the JavaScript code.
- Application custom errors.
- Authentication errors and information for OpenID providers.

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 client. 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 FTOS Portal web.config, section appSettings or 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 also, the automatic signature, please add the following keys:

```
<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.

Set up for the automatic signature with qualified electronic sign

After the installation of the ESign provider package, you should add the following configuration 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
="ESignProvider2AutomaticQESCertName"
 value="certSignTestAutomatic"/> <!-- the mapping for the
certificate provided by FTOS for the Automatic QES signature-->
```

Insert a record in the business entity **FTOS_DDM_ESignQueue**, this record will contain the configurations that will be used for automatic qualified electronic sign.

ProfileName, choose a name for this automatic profile, make sure it is unique if you have multiple configurations:

- ExternalId, this value should be provided CertSign, it will be the externalId of the user that is enrolled to sign with automatic QES
- Seed, this value will be read by the agent from his CertSign acount (he will receive an email with steps to follow)
- WorkstepsBulkNo, this represents the number of worksteps that will be sent in the request to be signed with automatic QES.

Calling the automatic signature with qualified electronic sign

The request for automatic QES will be sent together with the rest of the worksteps. The signing processes will be made in the order provided in the request. It is recommended that this signature to stay at the end because it will be processed async by Job Server.

The workstep with automatic QES should be like:

```
{
      "signatureTag": "#tagAgentQES#",
      "signatureType": FTOSServices.DDM.signatureType.AutomaticQES,
      "automaticProfile": "ProfileNameDefinedInQueue", //defined in
FTOS_DDM_ESignQueue
      "signatureStamp": { //this is for the signature stamp
        "SignerName":"Sign name",
        "Reason": "Credit loan", //this will appear in signature
details
        "Subject": "Bank signature",
        "ShowTimeStamp": true, //show date in signature stamp
        "FontSize": "12"
      }
    }
```

Example

```
var signRequest = {
  "workstepConfigs": [
    {
      "signatureType": "#tagClient#",

"signatureType"
:FTOSServices.DDM.signatureType.QualifiedElectronicSign,
      "recipient": {
        "Country": "RO",
        "Email": "@fintechos.com",
        "ExternalId": "", //an unique id representing the
customer (ex: Accoountid)
        "FirstName": "M",
        "LastName": "C",
        "PhoneMobile": "+407",
        "SocialSecurityNumber": "", //PIN
        "IdPhoto": "", // ftos file attribute value
representing the id picture

      },
      "signatureStamp":{
        "Reason": "Client reason",
        "Subject": "Credit loan",
        "SignerName":  "TestFirstName TestLastName",
        "FontSize": "12"
      }
    },
    {
      "signatureTag": "#tagAgentQES#",
      "signatureType":
FTOSServices.DDM.signatureType.AutomaticQES,
      "automaticProfile": "ProfileNameDefinedInQueue",
      "signatureStamp": { //this is for the signature stamp
        "SignerName":"Sign name",
        "Reason": "Credit loan", //this will appear in
signature details
        "Subject": "Bank signature",
        "ShowTimeStamp": true, //show date in signature
stamp
        "FontSize": "12"
      }
    }
  ],
  "signedDocumentName": "test.pdf",
  "files": [
    {
      "ftosFile": "" //ftos file attribute value
representing the pdf that needs to be signed
    }
  ]
}
```

NOTE

The automatic QES signing has to be processed by Job Server, so you must configure a schedule trigger with a server side script. Please make sure that you won't use the same ExternalId on multiple instances of FintechOS. Also, the scheduled trigger should be configured to run with a frequency of at least 30 seconds. The recommended cron expression should be to start at second 0 or at second 30 and to run from 30s in 30s (or a multiple of 30s = 1min, 2min, etc).

In the server side script you have to call the following method with the parameter ProfileName that you've defined in FTOS_DDM_ESignQueue:

```
//This method returns the list with processed eSignId and the
workstepId that has been finished
//As input you shoul pass the Name that you've configured in  FTOS_
DDM_ESignQueue
```

```
var result = FTOSServices.DDM.ESign2.processPendingAutomaticQESSign
("ProfileNameDefinedInQueue");


// if success the result will look like:
/*
{
  "isSuccess": true,
  "isFinished": true,
  "eSignProcesses": [
    {
      "eSignId": "e2f2bc20-de1f-41f1-851b-5c0279cc4cb7",
      "eSignWorkstepId": "13b2ea4e-f5cb-491a-a32a-268741e7da2a"
    },
    {
      "eSignId": "ac42d532-88a3-4db6-932c-9d0888e2fd0e",
      "eSignWorkstepId": "6830d22a-9309-4a11-9e9a-05ab7ac8807b"
    }
  ]
}
*/
```

FTOS ESign Services API

In order to sign a document you must call the following methods:

1. RequestSign (for the configuration of the automatic signature)

   For client signature with remote method with authorization code sent through sms:

2. AcceptTermsAndConditions
3. Authorize signature
4. Resend code

RequestSign

Firstly, add a reference to the library FTOSServices. To request with qualified electronic signature and automatic, use the following example:

```
var signRequest = {
    "workstepConfigs": [{
        "signatureTag": "#tagClient#",
        "signatureType":
FTOSServices.DDM.signatureType.QualifiedElectronicSign,
        "recipient": {
            "Country": "RO",
            "Email": "@fintechos.com",
            "ExternalId": "", //an unique id representing the
customer (ex: Accoountid)
            "FirstName": "M",
            "LastName": "C",
            "PhoneMobile": "+407",
            "SocialSecurityNumber": "", //PIN
            "IdPhoto": "", // ftos file attribute value
representing the id picture
        },
        "signatureStamp": {
            "Reason": "Client reason",
            "Subject": "Credit loan",
            "SignerName": "TestFirstName TestLastName"
        }
    }, {
        "signatureTag": "#tagBank#",
        "signatureType":
FTOSServices.DDM.signatureType.AutomaticSign,
        "automaticProfile": "Profile1",
        "signatureStamp": { //this is for the signature stamp
            "Reason": "Credit loan", //this will appear in
signature details
            "Subject": "Bank signature",
            "ShowTimeStamp": true //show date in signature stamp
        }
    }],
    "signedDocumentName": "test.pdf",
    "files": [{
        "ftosFile": "" //ftos file attribute value representing the
pdf that needs to be signed
    }]
}
Optionally, you can add to the recipient property the following
information.It will appear in terms and conditions file......
"workstepConfigs": [{
        "signatureTag": "#tagClient#",
        "signatureType":
FTOSServices.DDM.signatureType.QualifiedElectronicSign,
        "recipient": {......
```

```
            "DocumentIssuedBy": "splcid", "DocumentIssuedOn":
"2020-01-20", "DocumentExpiryDate": "2050-01-25", "DocumentNumber":
"123456", "DocumentSeries": "xa", "County": "Braila", "City":
"Braila", "Street": "asd", "StreetNo": "12", "Block": "asd",
"Entrance": "q", "ApartmentNo": "123", "ZipCode": "123453",
        }
```

HINT
If you have a request with multiple signatures, please keep in mind that the signing processes is sequentially and the client signature must have manual input (accept terms and conditions and authorize signing using the code received via sms).

You should save the eSignId in order to track the status of the eSign process using the method GetESignStatus.

AcceptTermsAndConditions

Add a reference to the client script library FTOS_DDM_ESignProvider2.

```
/**
* Accepts terms and conditions for emtitting the certificate
* @param termsId is the id returned by requestSign method
(entityId property)
* @param accepted should be set on true, if the user accepts
the terms and conditions
* @return documentId that will be used for authorize signing
*/
acceptTermsAndConditions(termsId: string, accepted:
boolean): Promise<any>
```

Example:

```
var ddmESign = ebs.importClientScript("FTOS_DDM_
ESignProvider2");
ddmESign.acceptTermsAndConditions(termsId, true).then
(function(result) {
    console.log(result);
    //output should be{isSuccess: true, entityId: "12437-
34873"}
}, function(error) {
    console.log(error);
});
```

Authorize signature

Add a reference to the client script library FTOS_DDM_ESignProvider2.

```
// @param documentId is the id returned by
acceptTermsAndConditions method (entityId property)
// @param code should be the code sent via sms for the
signing process

authorizeSign(documentId: string, code: string): Promise <
any > ;

// Example:

var ddmESign = ebs.importClientScript("FTOS_DDM_
ESignProvider2");
ddmESign.authorizeSign(documentId, code).then(function
(result) {
    console.log(result);
    //output should be {isSuccess: true}
}, function(error) {
    console.log(error);
});
```

Example:

```
var ddmESign = ebs.importClientScript("FTOS_DDM_
ESignProvider2");
ddmESign.authorizeSign(documentId, code).then(function
(result) {
    console.log(result);
    //output should be {isSuccess: true}
}, function(error) {
    console.log(error);
});
```

Resend sms code

As before, add a reference to the client script library FTOS_DDM_
ESignProvider2.

```
/**
```

```
* @param documentId is the id returned by
acceptTermsAndConditions method (entityId property)
*/
resendCode(documentId: string): Promise<any>
```

Example:

```
var ddmESign = ebs.importClientScript("FTOS_DDM_
ESignProvider2");
ddmESign.resendCode(ebs.getCurrentEntityId()).then(function
(result) {
    console.log(result);
    //output should be {isSuccess: true}
}, function(error) {
    console.log(error);
});
```

Async methods, can be used with JobServer:

Update statuses

This method should be called using Job Server scheduler. It will update the status of the in progress eSign processes.

FTOSServices.DDM.ESign2.updateStatusESignProcess();

ProcessAutomaticSign (sign with automatic signature)

This method should be called using Job Server scheduler. It gets the in progress esign processes that must be signed with automatic signatures.

FTOSServices.DDM.ESign2.processPendingAutomaticSign();

Get ESign Status

This method returns the status of the eSign process. If the status is Finished, then you can get the signed document name to use it in your digital journey.

```
/**
```

```
 * @param eSignId is the id returned by requestSign
 */
FTOSServices.DDM.ESign2.getESignStatus(eSignId): any
Example: var eSignStatus =
FTOSServices.DDM.ESign2.getESignStatus(eSignId);
log(toJson(eSignStatus));
/*should print:
{
"isSuccess": true,
"status": "Finished",
"documents": "[\r\n  {\r\n    \"Name\":
\"contract1.pdf\",\r\n    \"RealName\": \"contract1_
42dfa5ba-c1f4-4328-b095-
b6075d0c12ee.pdf\",\r\n    \"IsSuccess\":
true,\r\n    \"Message\": null,\r\n    \"ClientScript\":
null,\r\n    \"Serialized\": null,\r\n    \"ErrorCode\":
0,\r\n    \"UIResult\": null\r\n  }\r\n]"
}
/*
```

Example:

```
var eSignStatus = FTOSServices.DDM.ESign2.getESignStatus
(eSignId);
log(toJson(eSignStatus));
/*should print:
{
"isSuccess": true,
"status": "Finished",
"documents": "[\r\n  {\r\n    \"Name\":
\"contract1.pdf\",\r\n    \"RealName\": \"contract1_
42dfa5ba-c1f4-4328-b095-
b6075d0c12ee.pdf\",\r\n    \"IsSuccess\":
true,\r\n    \"Message\": null,\r\n    \"ClientScript\":
null,\r\n    \"Serialized\": null,\r\n    \"ErrorCode\":
0,\r\n    \"UIResult\": null\r\n  }\r\n]"
}
/*
```

Configure the CData Sync Service

The CData Sync service is required for the FintechOS Data Pipes data replication feature. CData Sync must be installed on the same machine as the FintechOS platform. The service is shared between FintechOS 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 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" on the previous page 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 Innovation Studio *web.config* file.

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

Opent the Innovation 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 Innovation 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.

1 Add a new section in the web.config file for the FTOSApiSMS service

Open the *web.config* file in a text editor and add a new entry in the `<configSections>` node:

```
<configSections>
...
  <section name="ftosApiSmsProvider" type=
"EBS.Core.Data.Services.CommunicationProviders.Sms.FtosApiSmsProvid
erConfig, EBS.Core.Data.Services"/>
</configSections>
```

2 Add the configuration settings for the FTOSApiSMS service

Add the following configuration element in the *web.config* file:

```
<ftosApiSmsProvider
xmlns="urn:EBS.Core.Data.Services.CommunicationProviders.Sms"
        serviceUrl="insertyourURL"
        subscriptionKey="insertyourkey"
        from="SantaClaus"
    />
```

where:

- serviceUrl - the URL to FintechOS SMS gateway;
- subscriptionKey - subscription key for the service;
- from - a text representing the sender of the message.

Customize the SMS messages sent for Multi-Factor Authentication

- web.config - add a new attribute on multiFactorAuthentication/providers/provider section.

  Name: “messageTemplate“. Its value should be one of the FTOS_CMB_ ActionTemplate records.
- identify the correct template - find FTOS_CMB_ActionTemplateContent child of FTOS_ CMB_ActionTemplate (with name equal to the value of messageTemplate attribute) that has a FTOS_CMB_CommunicationChannel (Channel) that has FTOS_DPA_ ChannelProvider (Bus Communication Provider) with name equal to the name of the

channel provider set on the MFA provider. The template also depends on user culture → try to find the template using the user culture (from EbsMetadata.UserSettings with fallback to default system culture);

- if a proper template cannot be identified → error;
- the message (the body of the sms) will be customizable with 2 tokens: {{otp}} (the generated OTP) and {{user_display_name}} (DisplayName of the current user);
- if "messageTemplate" is missing the message will contain just the OTP code (as it is now);

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 the FintechOS Portal web.config file:

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

Security

FintechOS 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 FintechOS web-based platform and the web browser, the connection is secured via SSL certificates.

The SSL-secured communication between FintechOS 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 can be exposed through REST APIs to enable integration with 3rd party systems / solutions. FintechOS 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 (Microsoft Azure, AWS, IBM Cloud, other).

XSS Prevention

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

In FintechOS, 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: ”<”, “</”, “>”, “&lt;” and “&gt;” (e.g., <text, </text, &lt;text or &gt;text) from being rendered.

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

XSS prevention when upgrading to FintechOS 20.1

When upgrading FintechOS to version 20.1, you should enable the request validation to the latest version; otherwise you will be vulnerable to cross-site scripting attacks. To do so, go to the **web.config** file and set the request validation version to **4.5**:

```
<httpRuntime targetFramework="4.6.2" requestValidationMode="4.5"
... />
```

Authentication

The second pillar of security, authentication is the process of verifying the identity of a user based on a set of credentials.

FintechOS provides the following authentication mechanisms:

FintechOS Authentication

You can log into FintechOS using the account credentials provided to you by your FintechOS administrator.

FintechOS provides you with extensive security measures to protect users access: ensure password security and unauthorize inactive users.

Microsoft Active Directory Authentication

Access within the platform is granted through authentication with FintechOS account credentials (username and password). The built-in integration with Microsoft Active Directory (AD) allows you to access FintechOS using your AD credentials.

Azure Active Directory Authentication

If your organization is using Azure Active Directory (Azure AD) for identity and access management, you can map Azure groups to FintechOS "Security Roles" on page 119 using the OpenID authentication protocol. This allows users to log in to FintechOS using their existing Azure AD credentials.

OpenID Connect Authentication

OpenID Connect is an interoperable authentication protocol based on the OAuth 2.0 specifications which uses straightforward REST/JSON message flows. It enables developers authenticate the users across their apps without having to own and manage password files. OpenID Connect securely identifies the identity of the person that is using an app.

The FintechOS built-in integration with Okta (a certified OpenID Connect provider) provides user authentication and single sign-on (SSO) functionality.

SSO means being able to access all the applications and resources that you need to do business, by signing in only once using a single user account. Once signed in, you can access all of the applications you need without being required to authenticate (for example, type a password) a second time.

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.

Multi-Factor Authentication

Multi-Factor Authentication is a method of authentication that requires the use of more than one verification method and adds a critical second layer of security to user sign-ins and transactions. Multi-Factor Authentication helps safeguard access to data and applications while meeting user demand for a simple sign-in process. It delivers strong authentication via a range of verification options: phone calls, text messages, or mobile app notifications or verification codes and third-party OAuth tokens.

FintechOS Auth Provider

For the authentication process, it is possible to implement if needed within the web.config file a special request upon logging into the FintechOS Portal or Innovation Studio. The full password is never requested in order to log in successfully, but only random characters contained in the password of a user.

Microsoft Active Directory Authentication

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

FintechOS 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 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 authentication with the Microsoft Active Directory authentication, go to the **web.config** file of your WebApp (Portal/Designer) and add/edit the following setting:

```
<appSettings>
...
<add key="EBSDefaultAuthentication" value="AD"/>
...
</appSettings>
```

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

NOTE

- When adding system users in FintechOS 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, 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 WebApp has the privileges to search into the directory entry nodes, otherwise, when trying to log in FintechOS using AD credentials, privileges related errors might occur.

Automatically Adding Users from AD

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

NOTE Automatically creating users from AD will remove the existing business units and security roles from FintechOS 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 using a configuration file, follow these steps:

1. In the **web.config** file fo your WebApp, add the following setting:

   ```
   <appSettings>
   ...
   <add key="EBSADAuthAutoCreateUsers" value="true"/>
   ...
   </appSettings>
   ```

2. In an xml file, create the mapping between the AD groups and the security roles and business units fromFintechOS. Name the file **ADUserConfiguration.xml**.

   Overwrite the Business Unit from FintechOS 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, go to the **web.config** file and add the following key:

```

<appSettings>
...
<add key="ADOverwriteBusinessUnit" value="false"/>
...
</appSettings>
```

To preserve the system user's security roles from FintechOS and merge them with the ones provided in the **ADConfiguration.xml** file, go to the **web.config** file and add the following key:

```
<appSettings>
...
```

```
<add key="ADOverwriteUSerRoles" value="false"/>
...
</appSettings>
```

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 appSettings keys in the web.config file:

- 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 appSettings keys in the web.config file.

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

Setting the users and groups containers:

```
<appSettings>
    <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"/>
    ....
</appSettings>
```

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

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, use the `core-setting-adauth-group-query-mode` key in the `<appSettings>` section of the *web.config* file:

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

- 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.

Azure Active Directory Authentication

If your organization is using Azure Active Directory (Azure AD) for identity and access management, you can map Azure groups to FintechOS "Security Roles" on page 119 using the OpenID authentication protocol. This allows users to log in to FintechOS using their existing Azure AD credentials.

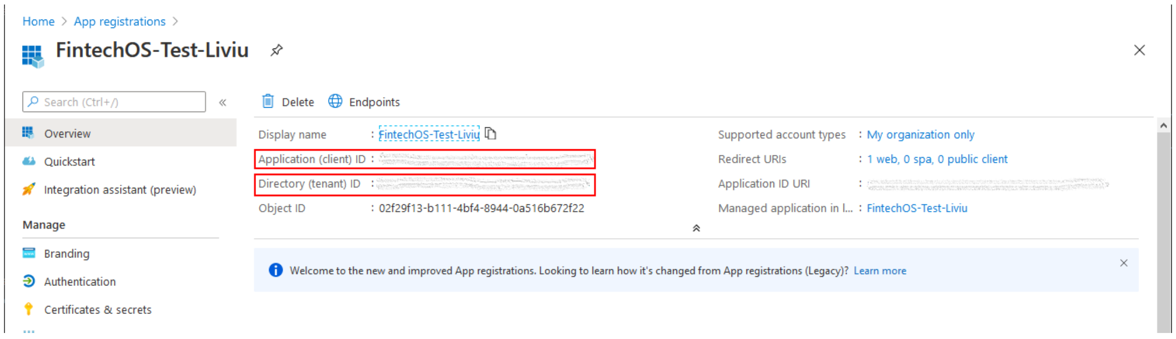
Configure OpenID Settings

Add the following keys to the `<appSettings>` 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.



Display name : FintechOS-Test-Liviu
Application (client) ID : [redacted]
Directory (tenant) ID : [redacted]
Object ID : 02f29f13-b111-4bf4-8944-0a516b672f22
Supported account types : My organization only
Redirect URIs : 1 web, 0 spa, 0 public client
Application ID URI : [redacted]
Managed application in l... : FintechOS-Test-Liviu

Welcome to the new and improved App registrations. Looking to learn how it's changed from App registrations (Legacy)? Learn more

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 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 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. 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 Azure AD user are lost.

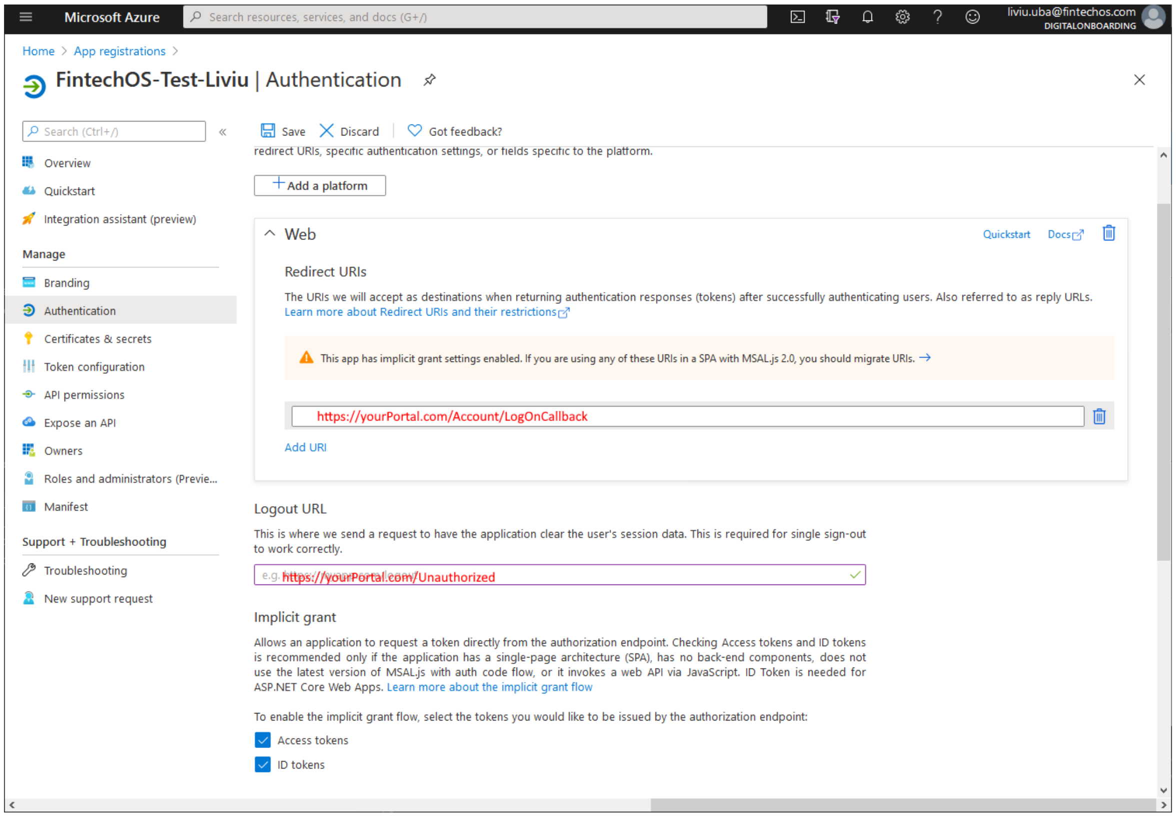
Parameters

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

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`



FintechOS-Test-Liviu | Authentication

redirect URIs, specific authentication settings, or fields specific to the platform.

Web

Redirect URIs

The URIs we will accept as destinations when returning authentication responses (tokens) after successfully authenticating users. Also referred to as reply URLs.
Learn more about Redirect URIs and their restrictions

This app has implicit grant settings enabled. If you are using any of these URIs in a SPA with MSAL.js 2.0, you should migrate URIs. →

https://yourPortal.com/Account/LogOnCallback

Logout URL

This is where we send a request to have the application clear the user's session data. This is required for single sign-out to work correctly.

https://yourPortal.com/Unauthorized

Implicit grant

Allows an application to request a token directly from the authorization endpoint. Checking Access tokens and ID tokens is recommended only if the application has a single-page architecture (SPA), has no back-end components, does not use the latest version of MSAL.js with auth code flow, or it invokes a web API via JavaScript. ID Token is needed for ASP.NET Core Web Apps. Learn more about the implicit grant flow

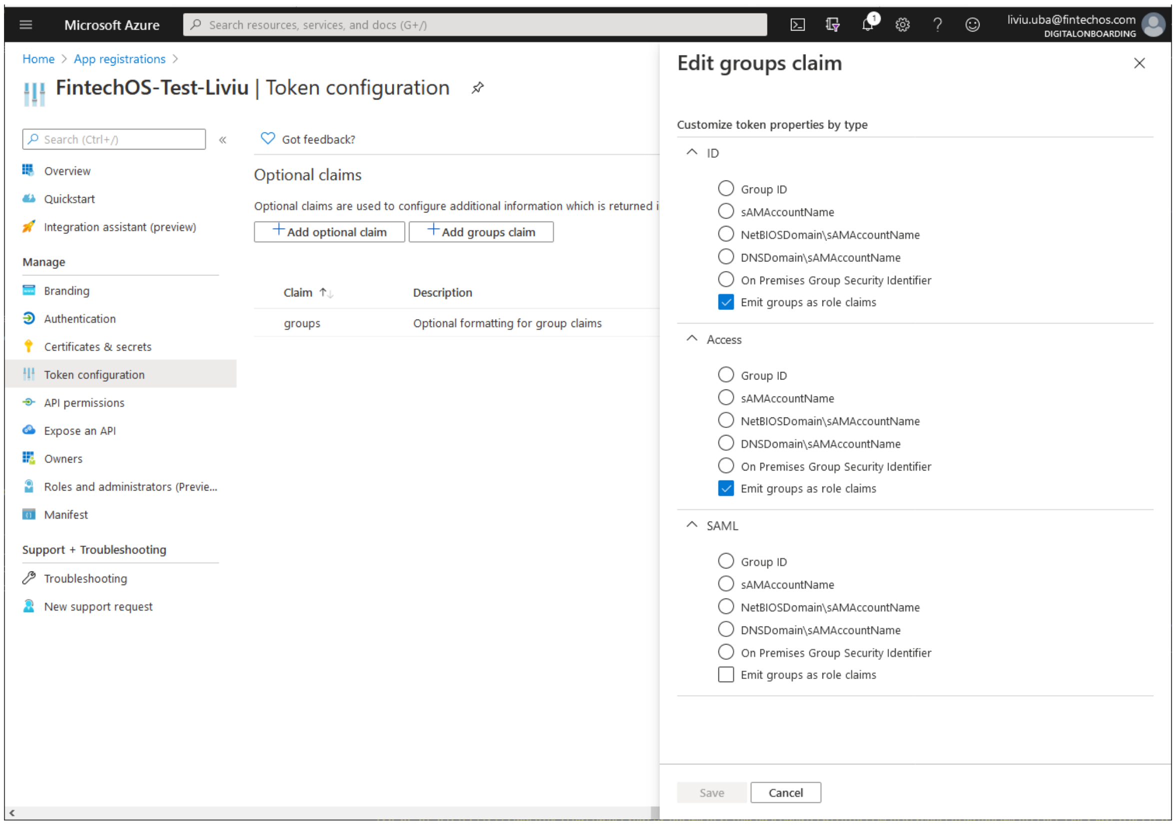
To enable the implicit grant flow, select the tokens you would like to be issued by the authorization endpoint:

- ✓ Access tokens
- ✓ ID tokens

Groups Mapping

When a user is authorized with Azure AD, a corresponding system user is created in FintechOS. 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, the administrator of the Azure AD application must include an optional claim named **groups** in the token configuration.


Microsoft Azure
Search resources, services, and docs (G+/)
liviu.uba@fintechos.com
DIGITALONBOARDING
Home > App registrations >

FintechOS-Test-Liviu | Token configuration

Optional claims

Optional claims are used to configure additional information which is returned i

Claim	Description
groups	Optional formatting for group claims

Edit groups claim

Customize token properties by type

ID
- Group ID
- sAMAccountName
- NetBIOSDomain\sAMAccountName
- DNSDomain\sAMAccountName
- On Premises Group Security Identifier
- [✓] Emit groups as role claims

Access
- Group ID
- sAMAccountName
- NetBIOSDomain\sAMAccountName
- DNSDomain\sAMAccountName
- On Premises Group Security Identifier
- [✓] Emit groups as role claims

SAML
- Group ID
- sAMAccountName
- NetBIOSDomain\sAMAccountName
- DNSDomain\sAMAccountName
- On Premises Group Security Identifier
- [ ] Emit groups as role claims

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 FintechOS Identity Provider

(FintechOS 21.2.2 and later)

The FintechOS Identity Provider built-in integration with OpenID compliant identity providers enables users to log in to the FintechOS Portal using the OpenID single sign-on (SSO).

How to Set up the FintechOS Identity Provider Authentication

Prerequisite:

Make sure that you know the following values from the OpenID identity provider:

- Client ID
- Client Secret
- Discovery Endpoint

In the **web.config** file, go to the <appSettings> section and add the configuration of your FintechOS Identity Provider setup:

```
<!-- 1. Set FintechOS Identity Provider authentication-->
<add key="EBSDefaultAuthentication" value="FTOSOIDC" />

<!-- 2. Replace these values with your FintechOS Identity Provider
configuration: -->
<add key="openid-client-id" value="{ClientId}" />
<add key="openid-client-secret" value="{ClientSecret}" />
<add key="openid-discovery-endpoint" value="
{DiscoveryEndpointUrl}" />
<add key="openid-callback-url" value="CallbackUrl" />
```

```
<!-- 3. FintechOS Identity Provider user role mapping settings: -->
<add key="membership-provider-connection-
username" value="admintest" />
<add key="membership-provider-connection-secret" value="1234567" />
```

For the user role mappings, fill in the following keys as described below:

Key	Description
membership-provider-connection-username	**(No longer required in v21.2.2.4 and later)** Requires an account set up in the FintechOS Identity Provider admin console with realm-management role.
membership-provider-connection-secret	**(No longer required in v21.2.2.4 and later)** Password for the above user account.

How users log in the FintechOS Portal

When accessing the FintechOS Portal, users who have a currently 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 Portal.

FintechOS user account automatic synchronization

When a user logs in to FintechOS Portal using FintechOS Identity Provider single sign-on, the first name, last name, and email address stored in the corresponding FintechOS user account are updated automatically based on the OpenID account settings.

Authentication with Okta

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

FintechOS 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:

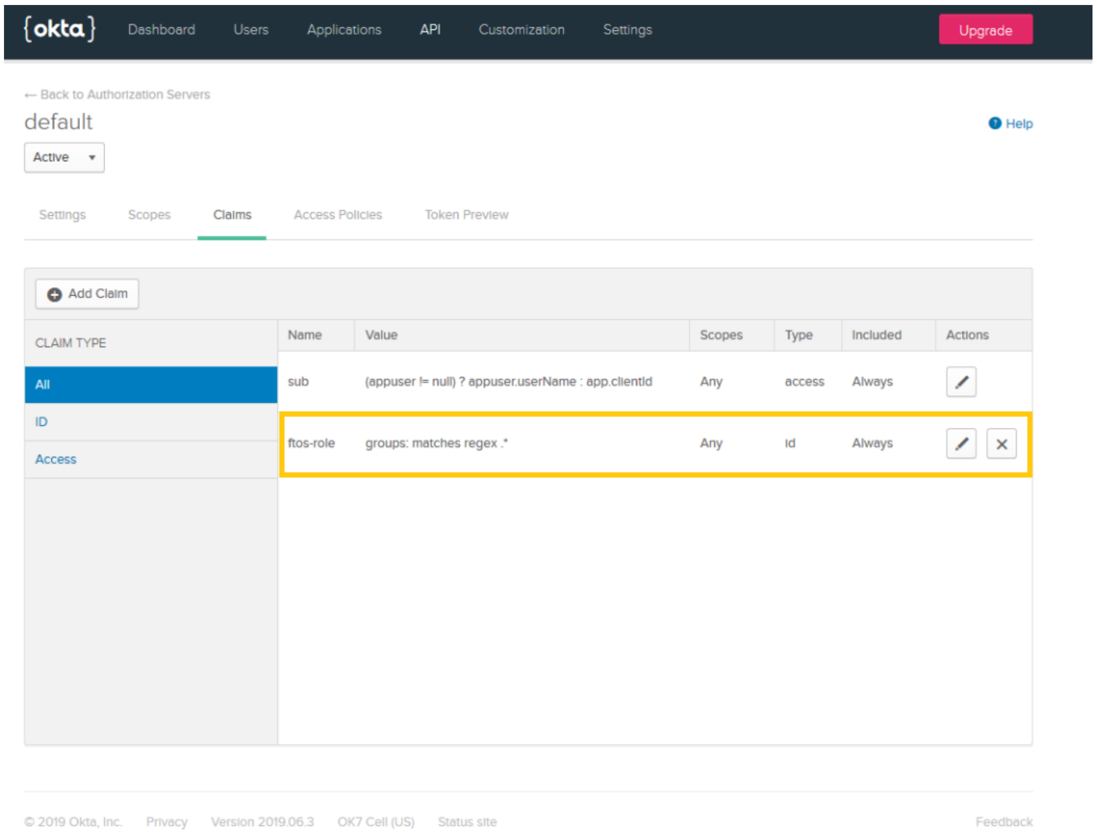
Step 1. Create and configure the Okta app

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 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. To do so, synchronize the user groups created in Okta with FintechOS 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.



Name	Value	Scopes	Type	Included	Actions
sub	(appuser != null) ? appuser.userName : app.clientId	Any	access	Always	
ftos-role	groups: matches regex .*	Any	id	Always	

When a user is authorized with Okta, a corresponding system user will be created in FintechOS . 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)

In the **web.config** file, go to the <appSettings> 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"/>
```

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

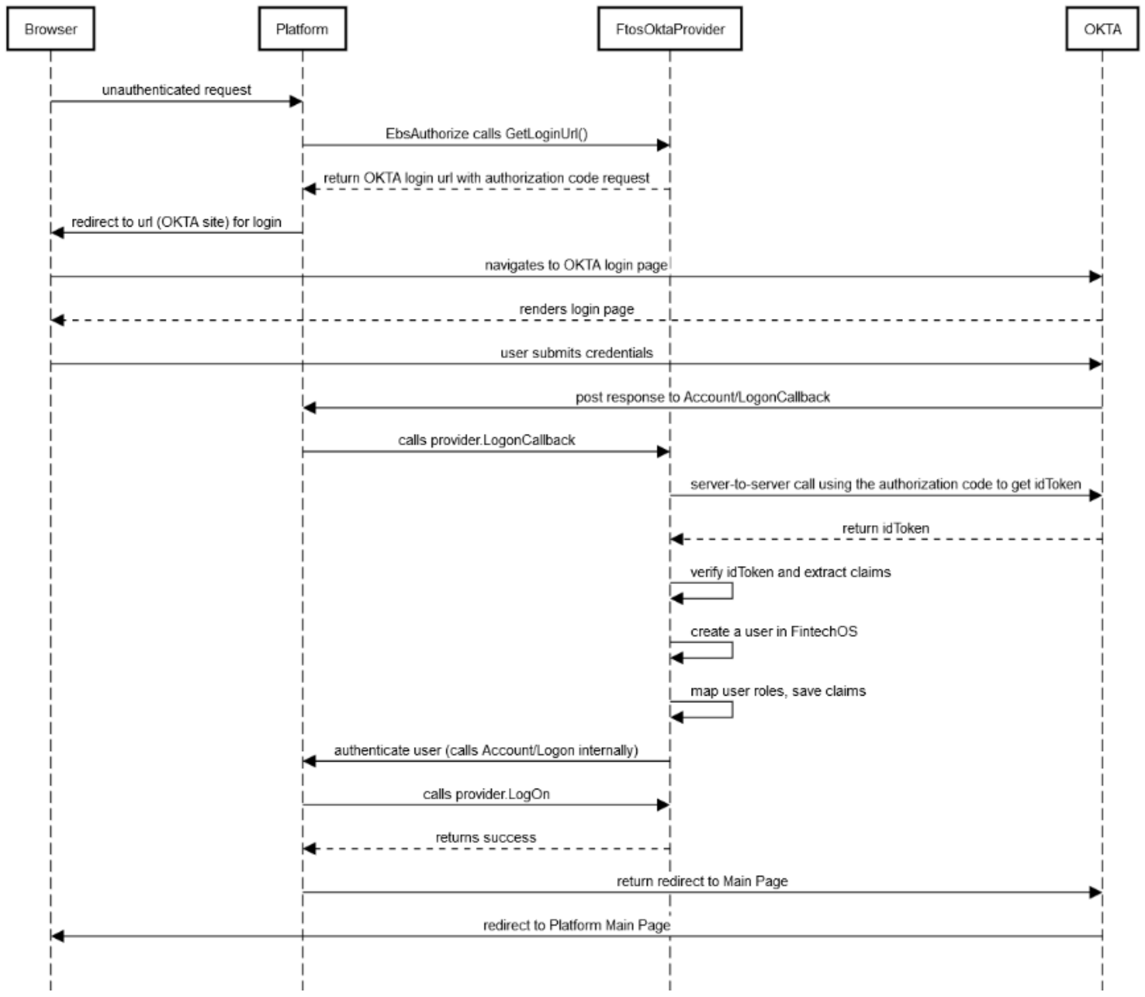
The table below describes the user mapping configuration keys.

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.

Parameter	Value
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.

How it Works

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


Browser
Platform
FtosOktaProvider
OKTA
unauthenticated request
EbsAuthorize calls GetLoginUrl()
return OKTA login url with authorization code request
redirect to url (OKTA site) for login
navigates to OKTA login page
renders login page
user submits credentials
post response to Account/LogonCallback
calls provider.LogonCallback
server-to-server call using the authorization code to get idToken
return idToken
verify idToken and extract claims
create a user in FintechOS
map user roles, save claims
authenticate user (calls Account/Logon internally)
calls provider.LogOn
returns success
return redirect to Main Page
redirect to Platform Main Page

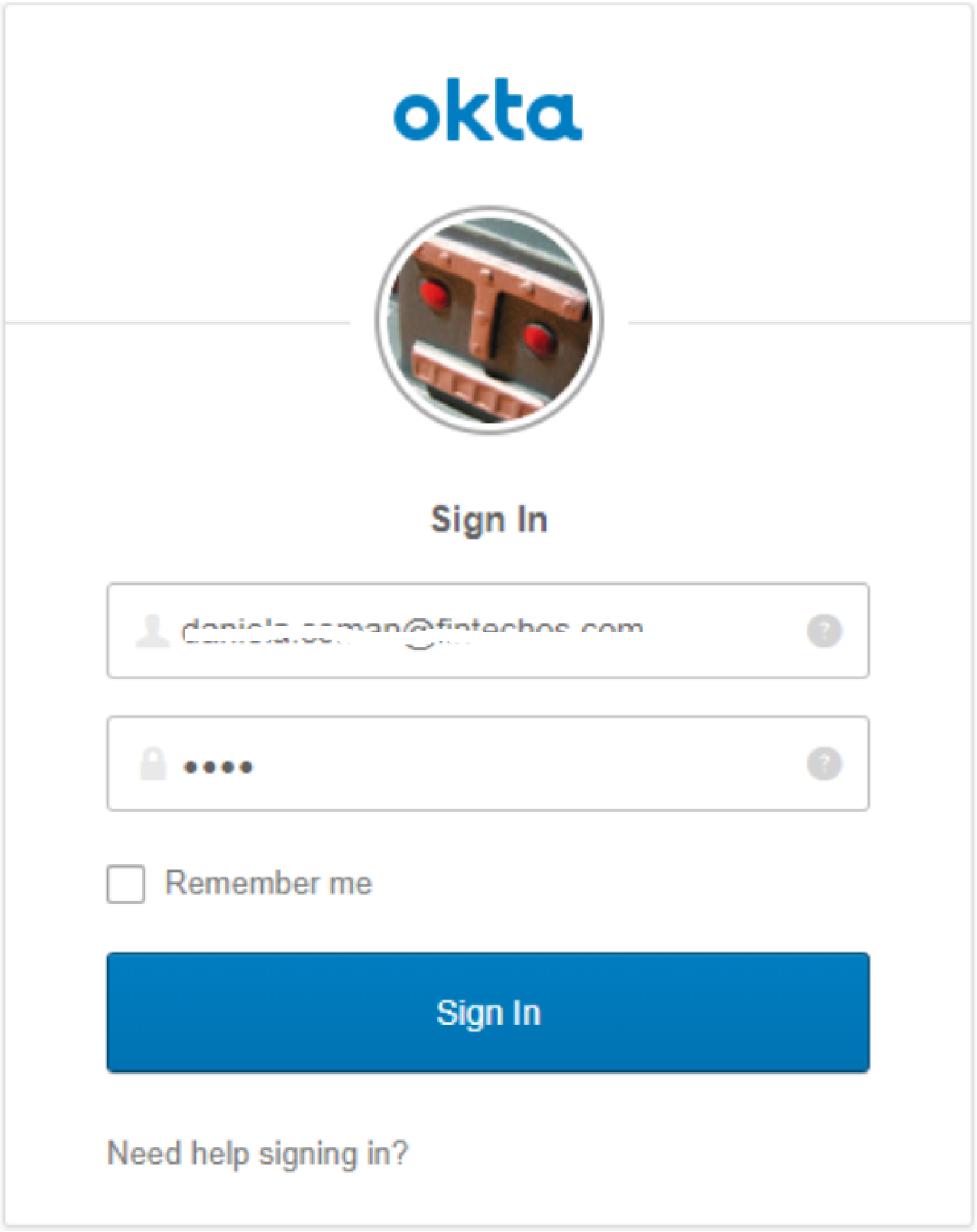
Group mapping in FintechOS

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.



daniela.coman@fintechos.com

••••

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.

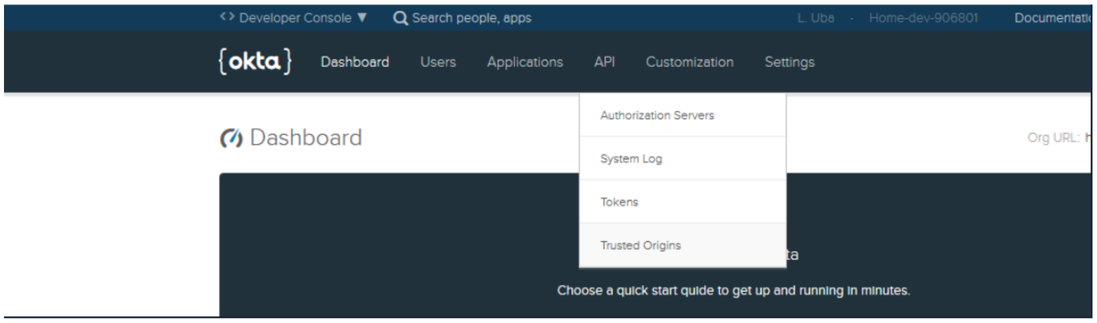
Cause

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

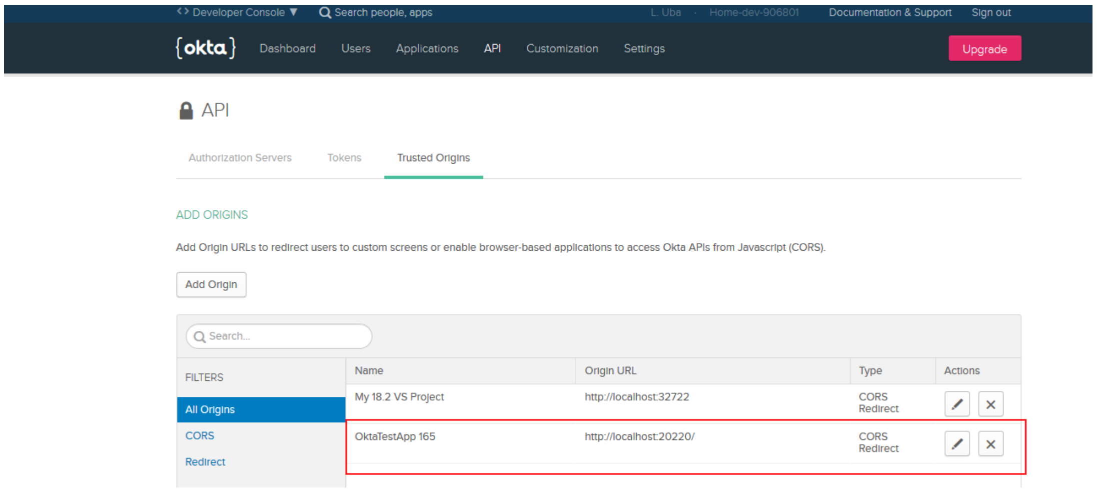
What should I do?

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.



Add Origin URLs to redirect users to custom screens or enable browser-based applications to access Okta APIs from Javascript (CORS).

Name	Origin URL	Type	Actions
My 18.2 VS Project	http://localhost:32722	CORS Redirect	
OktaTestApp 165	http://localhost:20220/	CORS Redirect	

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 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 -->
```

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

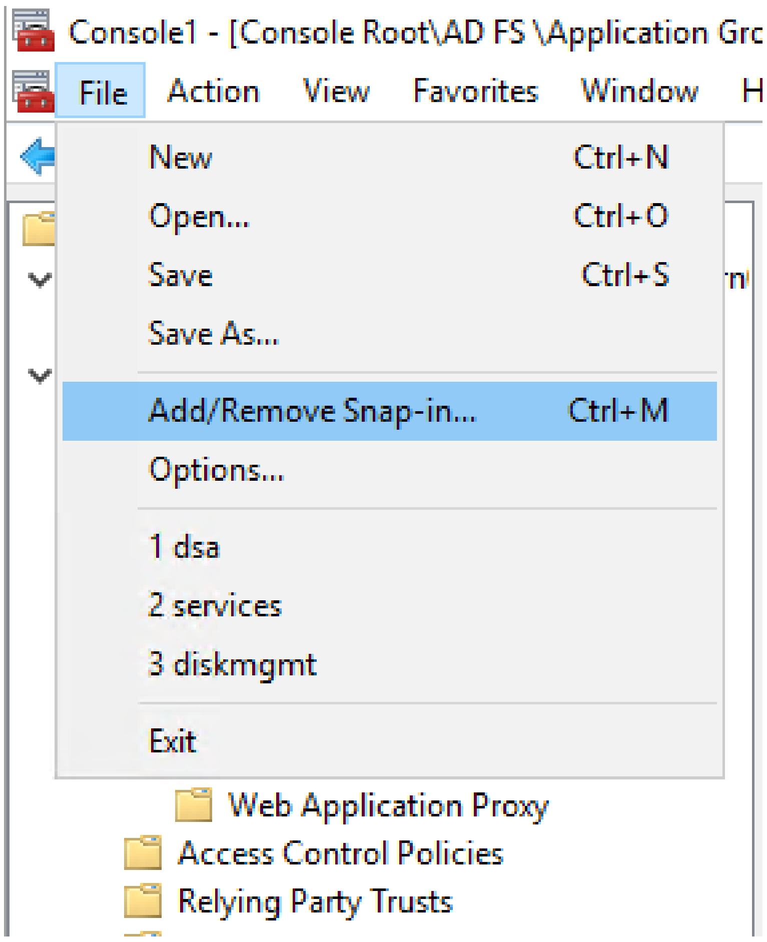
Parameters:

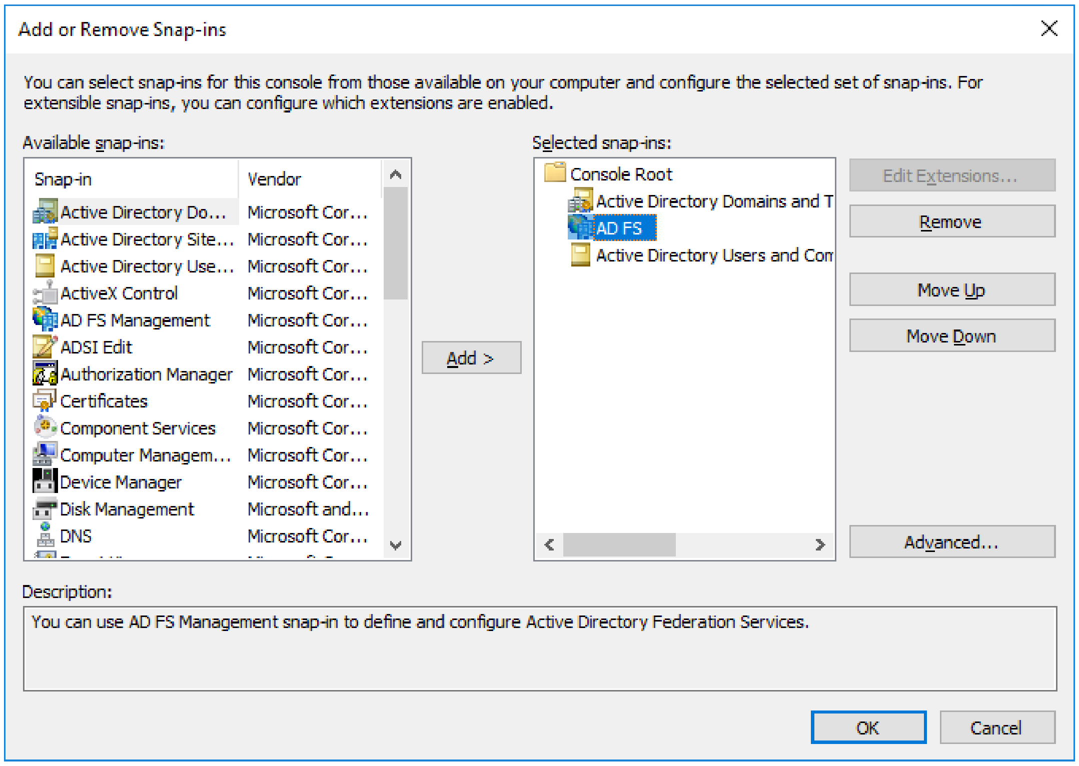
Parameter	Value
{portalRoot}	root url for FintechOS portal
{adfs server url}	ADFS server url

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.





Add or Remove Snap-ins

You can select snap-ins for this console from those available on your computer and configure the selected set of snap-ins. For extensible snap-ins, you can configure which extensions are enabled.

Available snap-ins:

Snap-in	Vendor
Active Directory Do...	Microsoft Cor...
Active Directory Site...	Microsoft Cor...
Active Directory Use...	Microsoft Cor...
ActiveX Control	Microsoft Cor...
AD FS Management	Microsoft Cor...
ADSI Edit	Microsoft Cor...
Authorization Manager	Microsoft Cor...
Certificates	Microsoft Cor...
Component Services	Microsoft Cor...
Computer Managem...	Microsoft Cor...
Device Manager	Microsoft Cor...
Disk Management	Microsoft and...
DNS	Microsoft Cor...

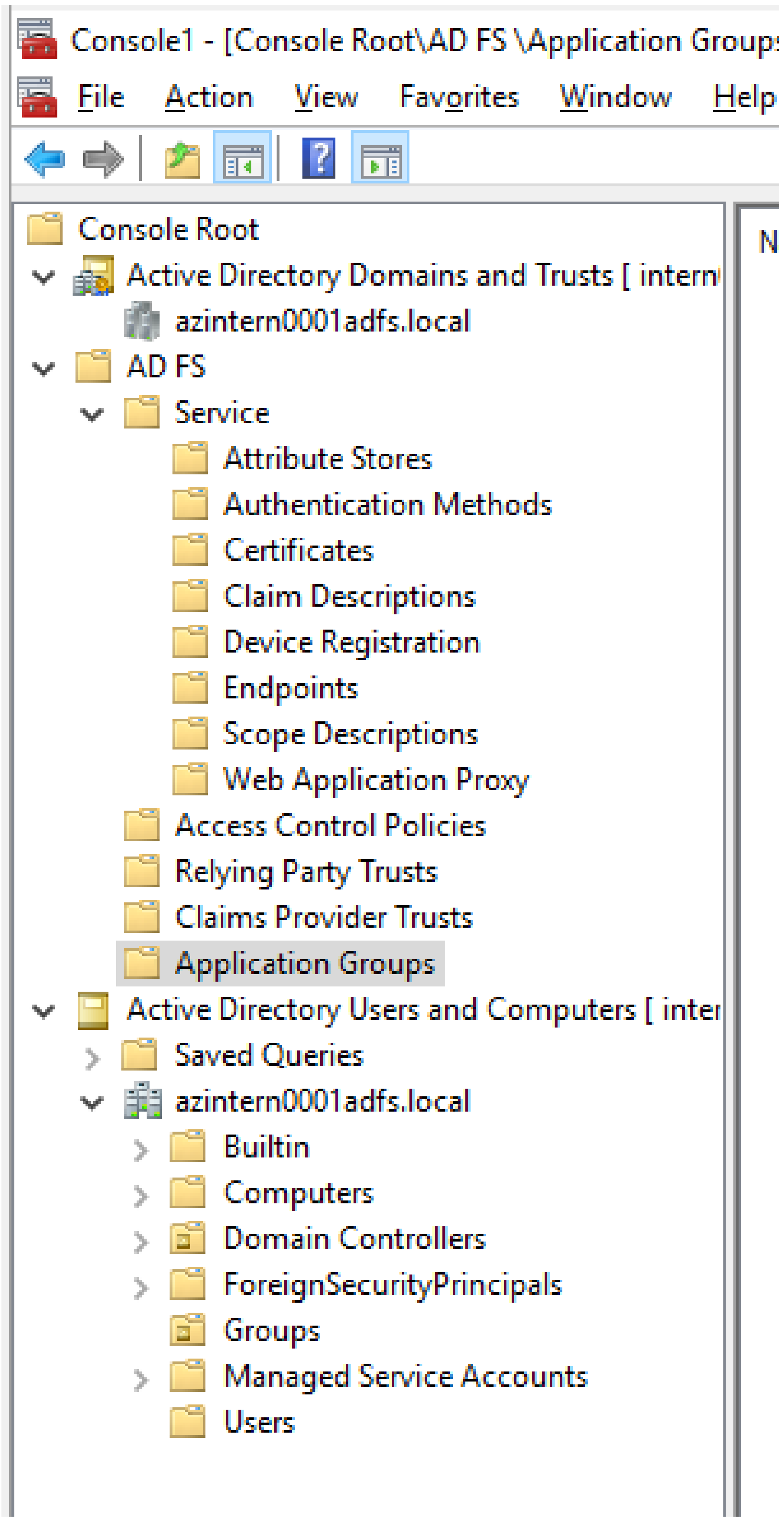
Selected snap-ins:

- Console Root
  - Active Directory Domains and T
  - AD FS
  - Active Directory Users and Com

Description:

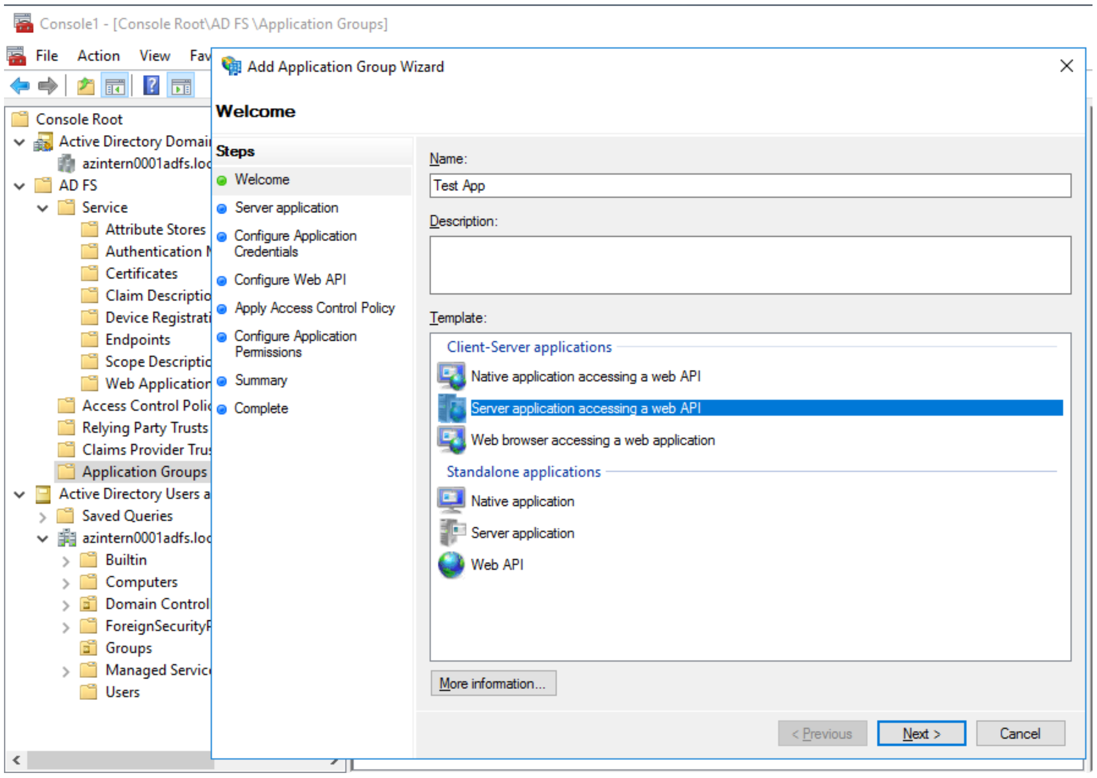
You can use AD FS Management snap-in to define and configure Active Directory Federation Services.

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



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.



Name: Test App

Description:

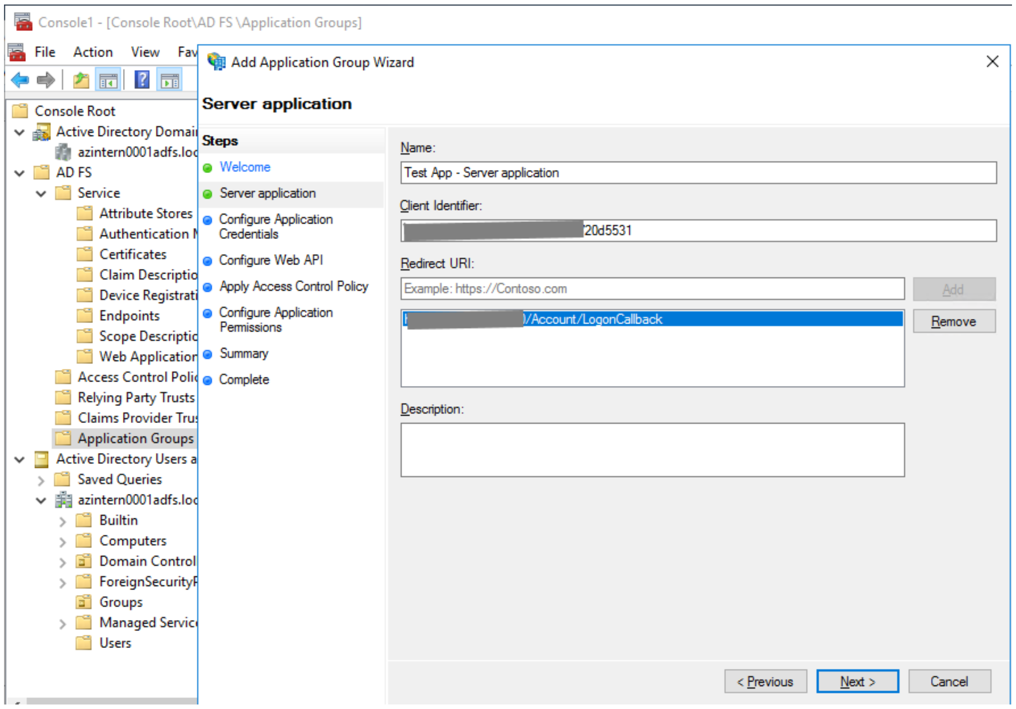
Template:

Client-Server applications
- Native application accessing a web API
- Server application accessing a web API
- Web browser accessing a web application

Standalone applications
- Native application
- Server application
- 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.



Server application

Name: Test App - Server application

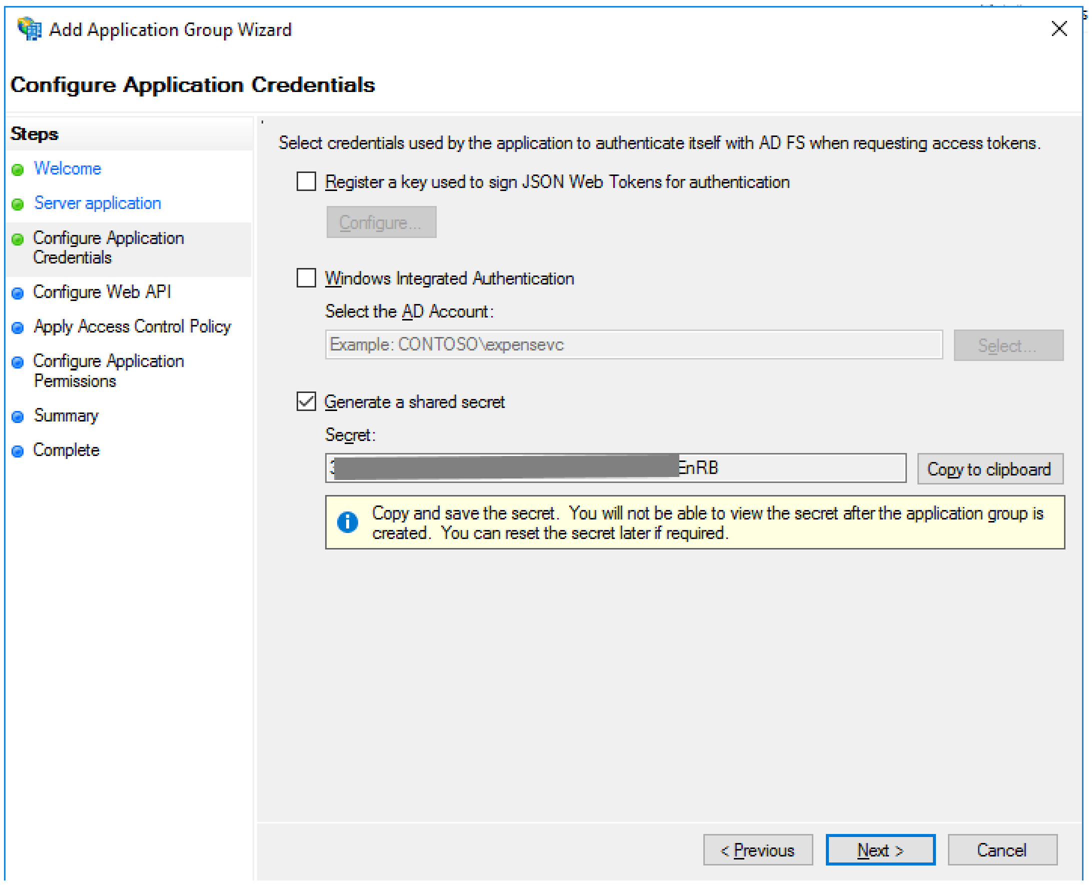
Client Identifier: [redacted]20d5531

Redirect URI: Example: https://Contoso.com

[redacted]/Account/LogonCallback

Description:

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

☐ Windows Integrated Authentication

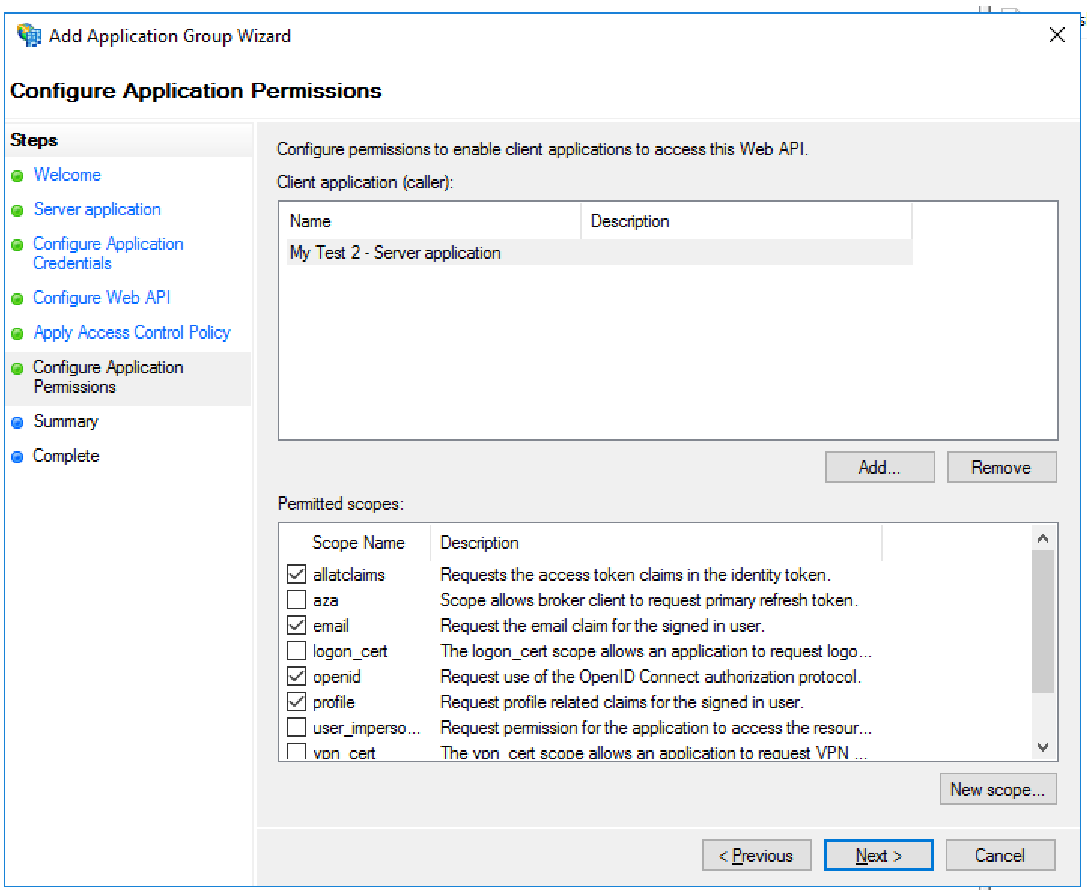
Select the AD Account: Example: CONTOSO\expensevc

☑ Generate a shared secret

Secret: 3[redacted]EnRB

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.

6. Configure the Web API identifier


Add Application Group Wizard
Steps
Welcome
Server application
Configure Application Credentials
Configure Web API
Apply Access Control Policy
Configure Application Permissions
Summary
Complete
Add...
Remove
New scope...
< Previous
Next >
Cancel

Configure Application Permissions

Configure permissions to enable client applications to access this Web API.

Client application (caller):

Name	Description
My Test 2 - Server application	

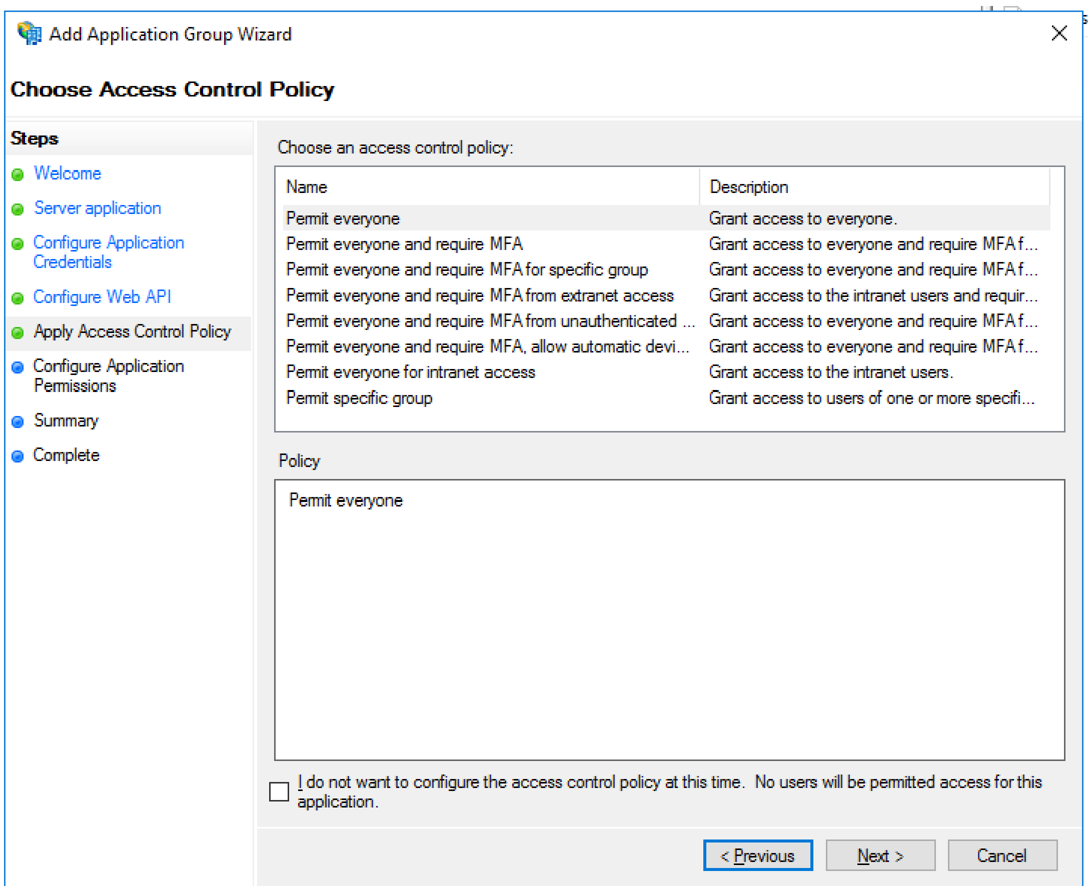
Permitted scopes:

	Scope Name	Description
☑	allatclaims	Requests the access token claims in the identity token.
☐	aza	Scope allows broker client to request primary refresh token.
☑	email	Request the email claim for the signed in user.
☐	logon_cert	The logon_cert scope allows an application to request logo...
☑	openid	Request use of the OpenID Connect authorization protocol.
☑	profile	Request profile related claims for the signed in user.
☐	user_imperso...	Request permission for the application to access the resour...
☐	vpn_cert	The vpn_cert scope allows an application to request VPN ...

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.



Add Application Group Wizard

Choose Access Control Policy

Steps
- Welcome
- Server application
- Configure Application Credentials
- Configure Web API
- Apply Access Control Policy
- Configure Application Permissions
- Summary
- Complete

Choose an access control policy:

Name	Description
Permit everyone	Grant access to everyone.
Permit everyone and require MFA	Grant access to everyone and require MFA f...
Permit everyone and require MFA for specific group	Grant access to everyone and require MFA f...
Permit everyone and require MFA from extranet access	Grant access to the intranet users and requir...
Permit everyone and require MFA from unauthenticated ...	Grant access to everyone and require MFA f...
Permit everyone and require MFA, allow automatic devi...	Grant access to everyone and require MFA f...
Permit everyone for intranet access	Grant access to the intranet users.
Permit specific group	Grant access to users of one or more specifi...

Policy

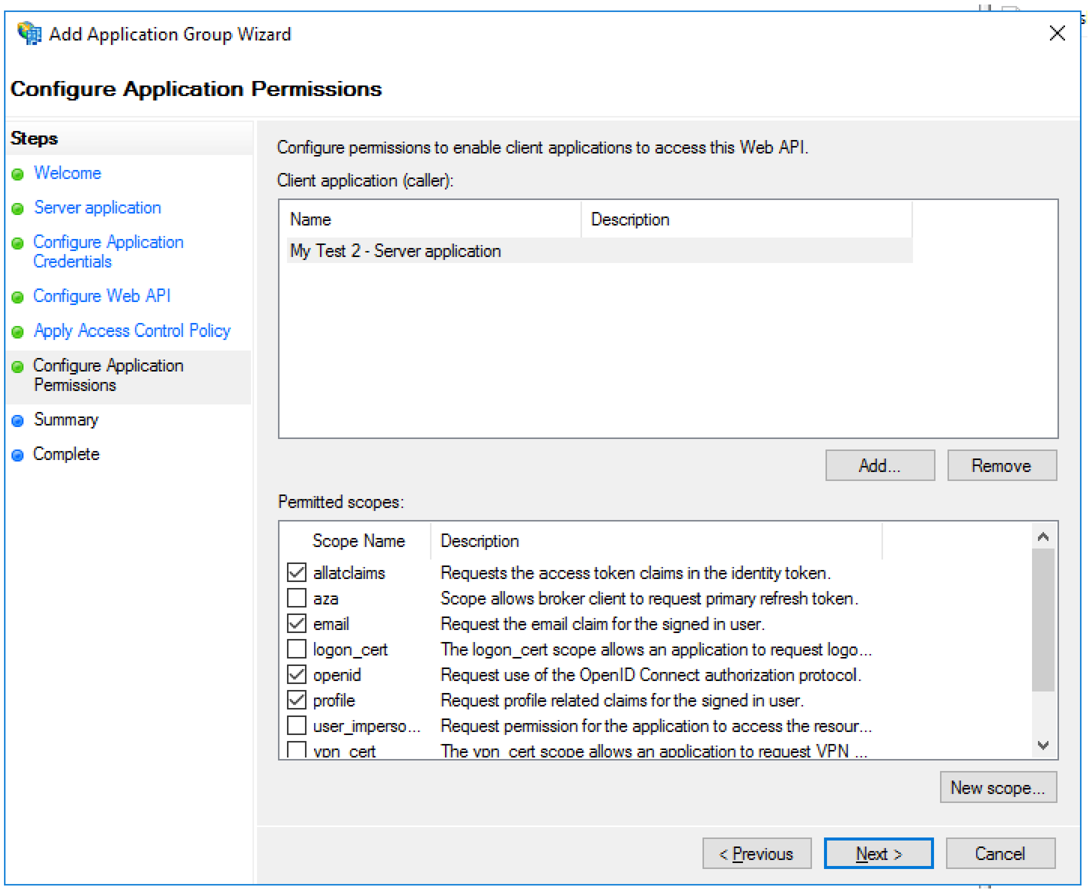
Permit everyone

☐ I do not want to configure the access control policy at this time. No users will be permitted access for this application.

< Previous | Next > | Cancel

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

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


Add Application Group Wizard
Steps
Welcome
Server application
Configure Application Credentials
Configure Web API
Apply Access Control Policy
Configure Application Permissions
Summary
Complete
Add...
Remove
New scope...
< Previous
Next >
Cancel

Configure Application Permissions

Configure permissions to enable client applications to access this Web API.

Client application (caller):

Name	Description
My Test 2 - Server application	

Permitted scopes:

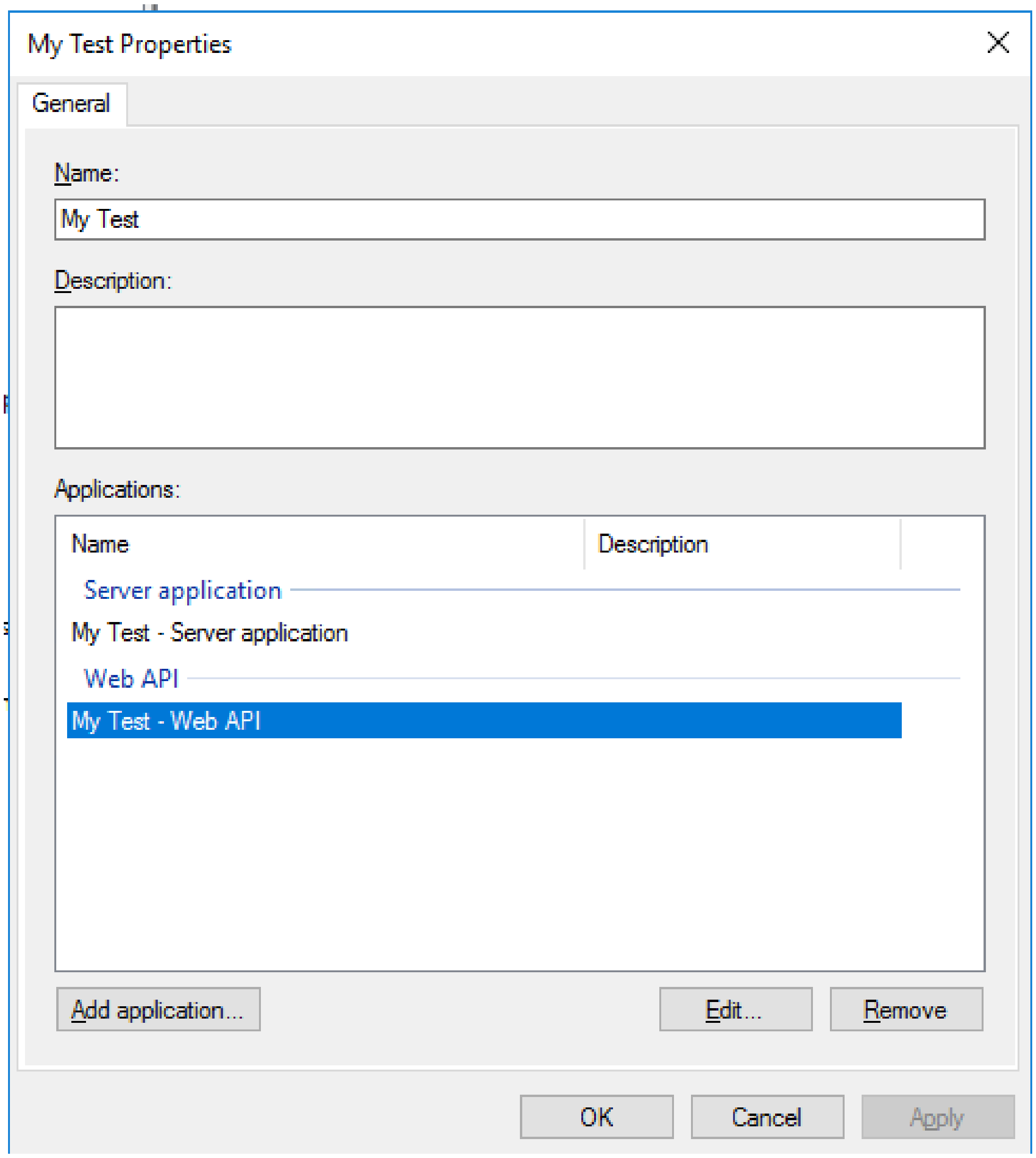
	Scope Name	Description
☑	allatclaims	Requests the access token claims in the identity token.
☐	aza	Scope allows broker client to request primary refresh token.
☑	email	Request the email claim for the signed in user.
☐	logon_cert	The logon_cert scope allows an application to request logo...
☑	openid	Request use of the OpenID Connect authorization protocol.
☑	profile	Request profile related claims for the signed in user.
☐	user_imperso...	Request permission for the application to access the resour...
☐	vpn_cert	The vpn_cert scope allows an application to request VPN ...

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.


My Test Properties

General

Name:

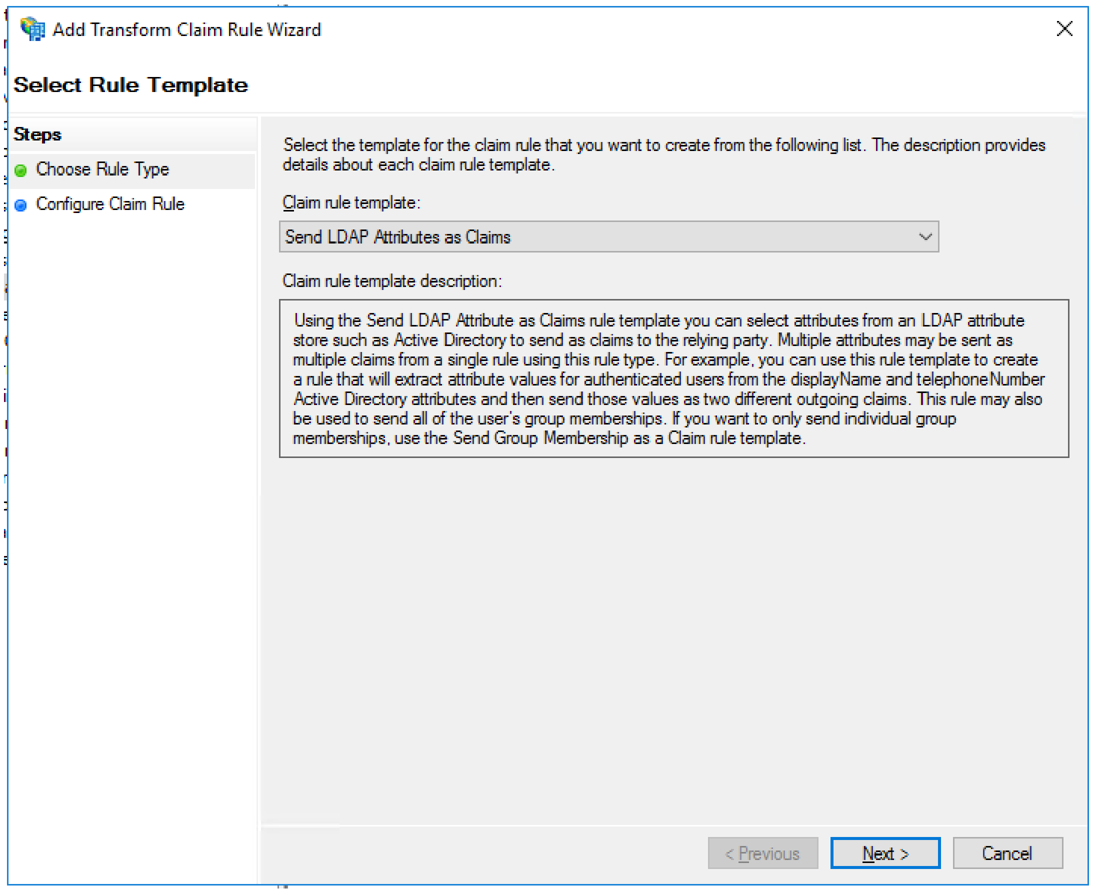
My Test

Description:

Applications:

Name	Description
Server application	
My Test - Server application	
Web API	
My Test - Web API	

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



Add Transform Claim Rule Wizard

Select Rule Template

Steps
- Choose Rule Type
- Configure Claim Rule

Select the template for the claim rule that you want to create from the following list. The description provides details about each claim rule template.

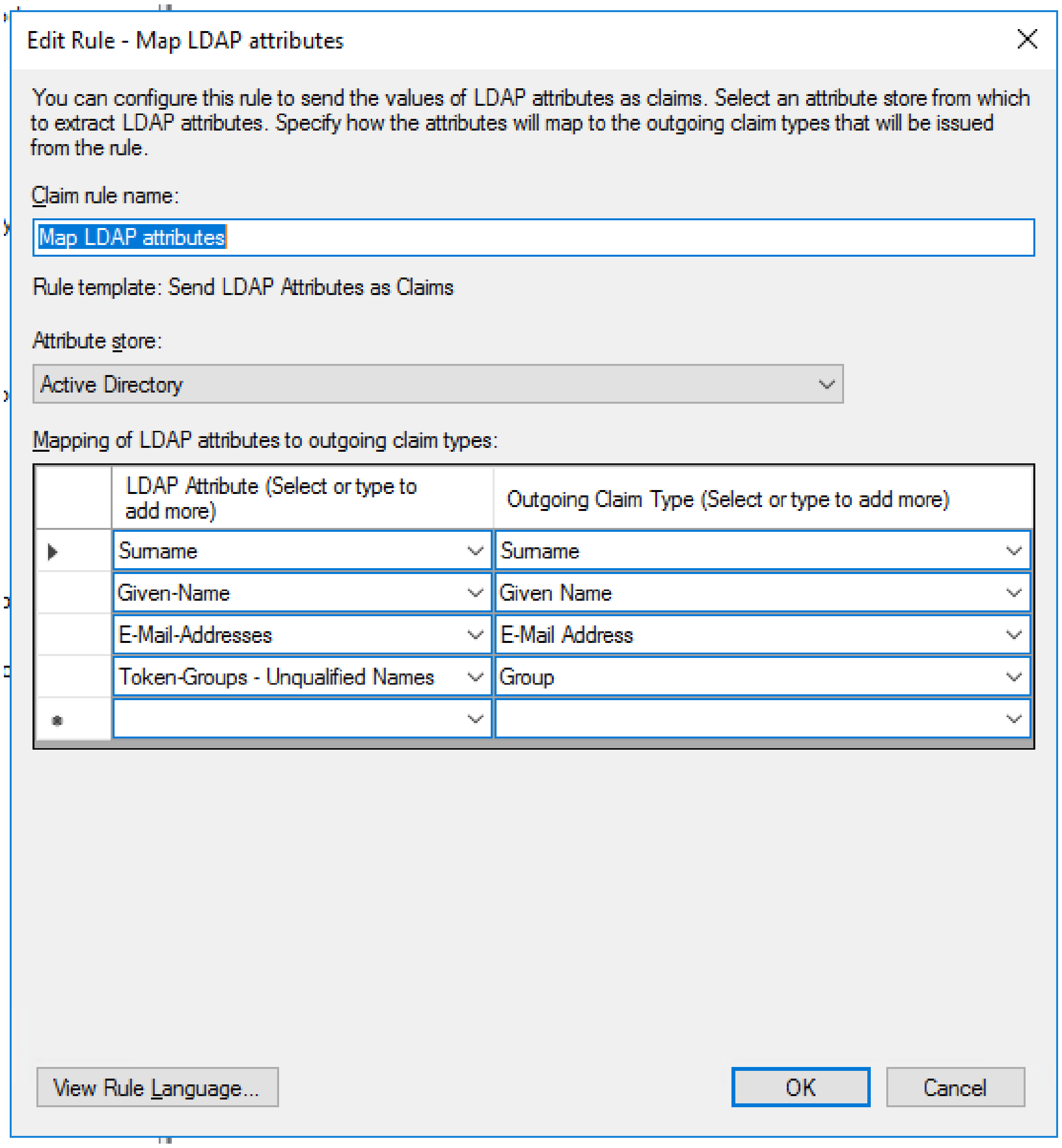
Claim rule template: Send LDAP Attributes as Claims

Claim rule template description:

Using the Send LDAP Attribute as Claims rule template you can select attributes from an LDAP attribute store such as Active Directory to send as claims to the relying party. Multiple attributes may be sent as multiple claims from a single rule using this rule type. For example, you can use this rule template to create a rule that will extract attribute values for authenticated users from the displayName and telephoneNumber Active Directory attributes and then send those values as two different outgoing claims. This rule may also be used to send all of the user's group memberships. If you want to only send individual group memberships, use the Send Group Membership as a Claim rule template.

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:



Edit Rule - Map LDAP attributes

You can configure this rule to send the values of LDAP attributes as claims. Select an attribute store from which to extract LDAP attributes. Specify how the attributes will map to the outgoing claim types that will be issued from the rule.

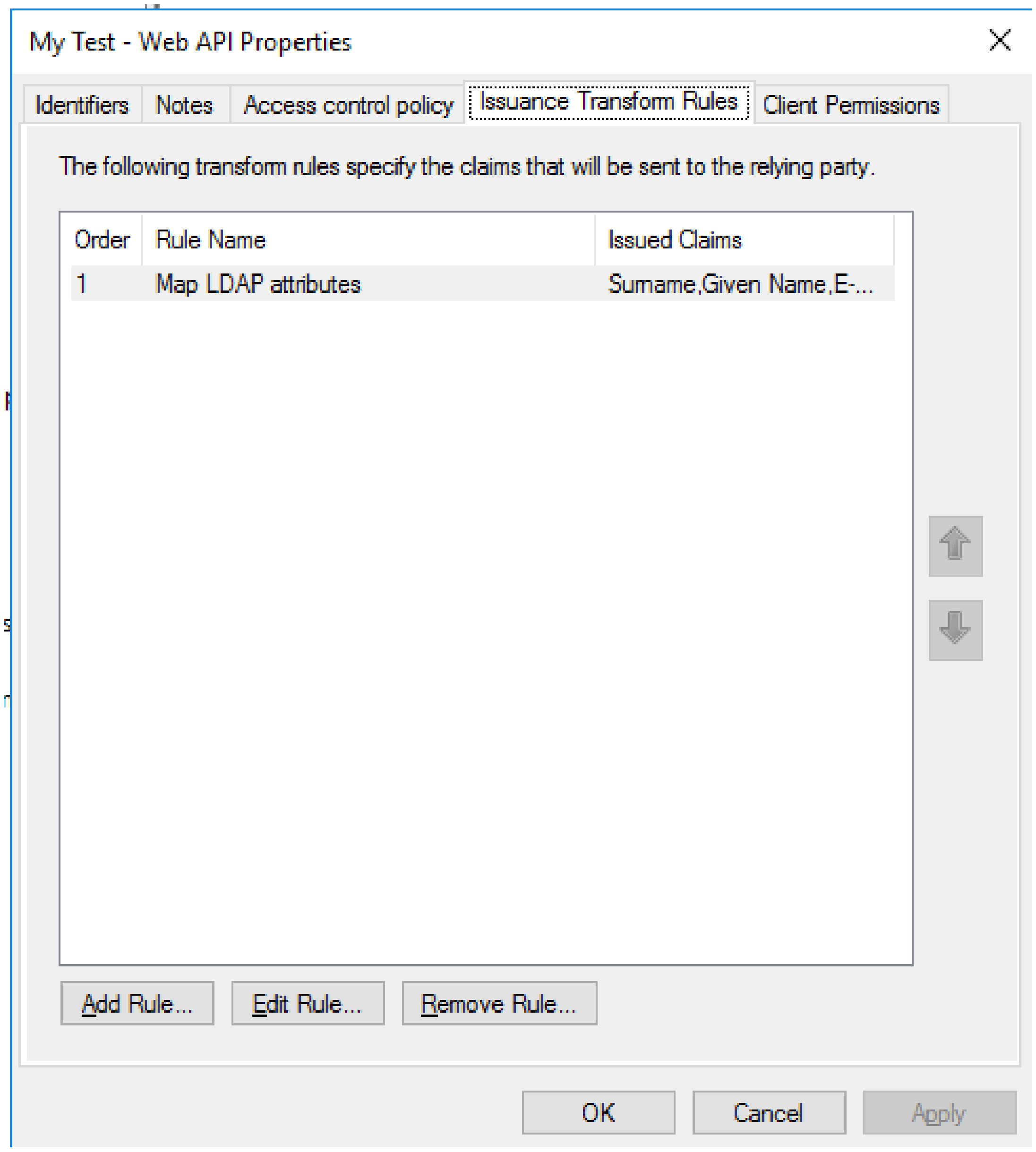
Claim rule name: Map LDAP attributes

Rule template: Send LDAP Attributes as Claims

Attribute store: Active Directory

Mapping of LDAP attributes to outgoing claim types:

	LDAP Attribute (Select or type to add more)	Outgoing Claim Type (Select or type to add more)
▶	Surname	Surname
	Given-Name	Given Name
	E-Mail-Addresses	E-Mail Address
	Token-Groups - Unqualified Names	Group
*		



The following transform rules specify the claims that will be sent to the relying party.

Order	Rule Name	Issued Claims
1	Map LDAP attributes	Surname,Given Name,E-...

Group mapping in FintechOS

Once the system user has been created in the Innovation 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

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 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 -->
```

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

Parameters:

Parameter	Value
${portalRoot}	root url for FintechOS portal

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>
```

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 the web.config, under `<appSettings>` to enable the feature:

- ebsauth-partial-password to true (default this is false),
- within the keys that have the following structure:

```
<appSettings>
...
<add key="core-setting-ebsauth-partial-password" value="true"/>
...
</appSettings>
```

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.

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 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 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 app following these two steps:

- Users will navigate to the FintechOS 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 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.Web.MVC.Security.SmsMultiFactorAuthenticationProvi
der, EBS.Core.Web.MVC" />
                <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 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 FintechOS 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.Web.MVC.Security.SmsMultiFactorAuthenticationProv
ider, EBS.Core.Web.MVC" />
                <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 app following these two steps:

- Users will navigate to the FintechOS 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 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;

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 FintechOS installation package resides, go to the **schedule.config** file and add the following section:

2.
```
<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>
```

3. On the server where the FintechOS 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>-->
```

4.

```
<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 key to the `<appSettings>` node in the Innovation Studio *web.config* file:

```
<appSettings>
    ...
    <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;descri
ption&apos;:&apos;Client certificate for Api1 a&apos; }"/>
```

```
    ...
</appSettings>
```

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.

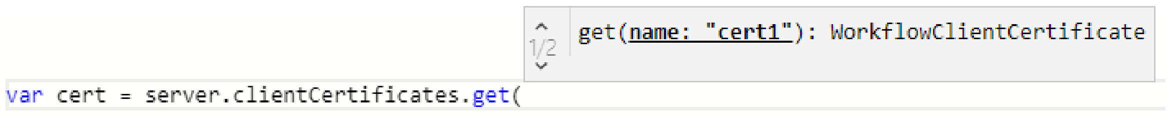
Property	Description
`checkValidity`	• true - Even if the thumbprint is found, the API returns the certificate only if the root issuer in the certificate build chain is part of the trusted root certification authorities. • false - For development or testing purposes.

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.


```
get(name: "cert1"): WorkflowClientCertificate
var cert = server.clientCertificates.get(
```

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 the *web.config* file. 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 key to the `<appSettings>` node in the *web.config* file.

```
<appSettings>
   ...
   <!-- 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':
'xyzxyzxyzxyzxyzxyzxyzxyzxyzxyzxyzxyzxyzxyzxyzxyz',
        'description': 'Client certificate for wso2-devel',
        'checkValidity': false
    }
}"/>

</appSettings>
```

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 113.

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, 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, 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 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 *Innovation Studio User Guide*.

Data Ownership

In FintechOS, 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 *Innovation Studio User Guide*, section Security.

Password Security

By default, FintechOS can log into the Innovation 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 has various options in place to ensure password security:

- prevent users to log 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 password complexity either from the **web.config** file (see section Global Password Complexity Settings) or by using server scripting (see section Customize Password Complexity Rules using Server Scripting).

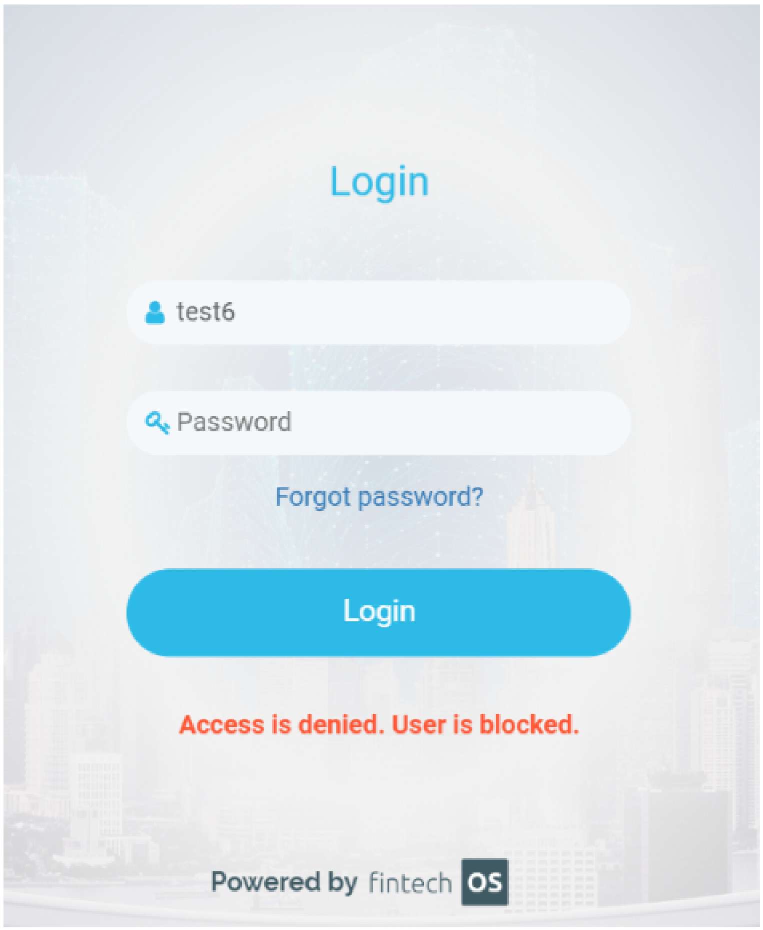
When users will choose to reset their password, an email is sent to the email address associated with their FintechOS account. FintechOS offers a default email template that is used for password reset. It’s easy to customize the default email template, or by using server scripting .

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.

Locked account

If users enter a wrong password multiple times, reaching the maximum number of retries (that is, 5), their account will be locked.



Access is denied. User is blocked.

To unlock their account they should contact their FintechOS admin to unlock their account. After the account is unlocked, they will be able to log in using the last password (if they remember it) or recover the password if they forgot it.

Password expired

If the password is expired, a message displays on the login page notifying the user that the password. It also provides the user with the option to reset the password.

NOTE This feature is available only for EBS Authentication Provider.

Activate Forgot Password Feature

In FintechOS, the Forgot Password feature allows users to reset their password and enables FintechOS developers to set the password complexity and to customize the password reset email template.

NOTE The forgot password feature is disabled by default and the validity of the password reset token is by default 15 minutes.

To activate the forgot password feature, on the server where the FintechOS installation package resides, go to the **web.config** file and set the following setting:

```
<appSettings>
    <add key="feature.reset-password" value="1" />
    ...
</appSettings>
```

In order to send email instructions to users who have requested password reset from the login page, also make sure to include the following settings in the **web.config** file:

```
<add key="SMTP:Port" value="" />
<add key="SMTP:Host" value="" />
<add key="SMTP:EnableSSL" value="" />
<add key="SMTP:User" value="" />
<add key="SMTP:Password" value="" />
<add key="DefaultFromEmail" value="" />
```

In the **web.config** file, set the validity of the password reset token by configuring the following key: **PasswordResetExpiration**. The default value of this key is 15 minutes.

Token expiration after 5 minutes:

```
<appSettings>
    <add key="PasswordResetExpiration" value="00:05:00"/>
    ...
</appSettings>
```

Configure Password Change

FintechOS 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 installation package resides, go to the **web.config** file and add the following setting:

```
<add key="core-setting-ebsauth-password-min-age" value="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.

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

To enable password expiry feature (Maximum password age), on the server where the FintechOS installation package resides, go to the **web.config** file and add the following setting:

```
<add key="core-setting-ebsauth-password-max-age" value="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 from **web.config** 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.

Configuring password change based on password history

FintechOS 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 installation package resides, go to the **web.config** file and add the following setting:

```
<add key="core-setting-ebsauth-password-history-depth" value="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).

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 installation package resides, go to the **web.config** file and add the following setting:

```
<add key="core-setting-ebsauth-password-about-to-expire-days-until-
expiration" value="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.

To customize the notification message , in the **web.config** file, add the following setting:

```
<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.'}]"/>
```

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.

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, in the **web.config** field, add the following setting:

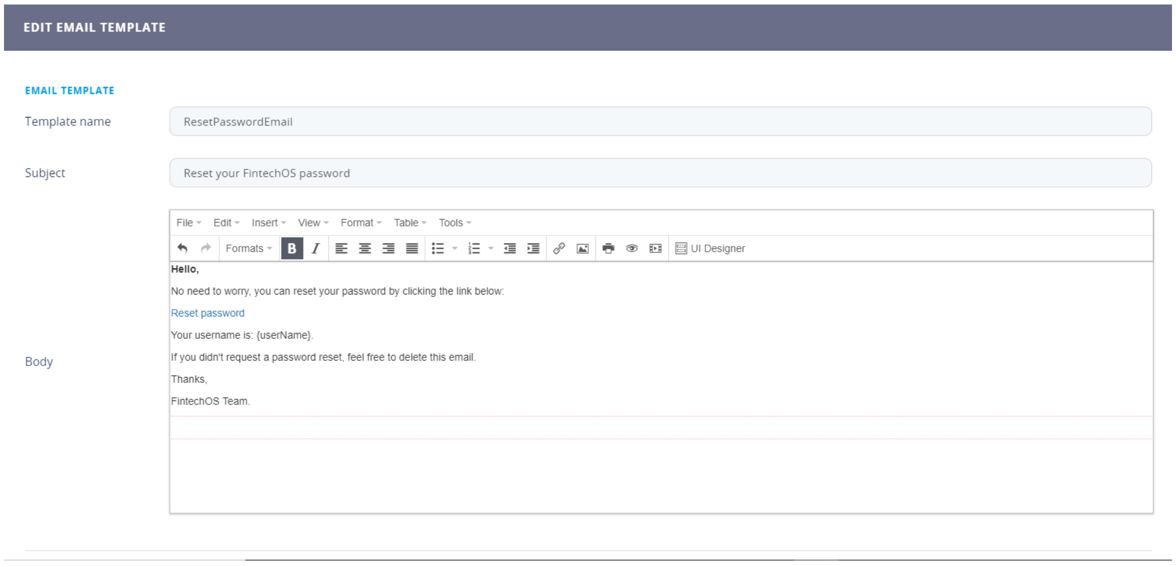
```
<add key="core-setting-ebsauth-password-expired-excepted-
role" value="securityRole"/>
```

The users with the security role specified in the value will never have to reset the password due to the password expiry rule.

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.



Template name	ResetPasswordEmail
Subject	Reset your FintechOS password
Body	**Hello,** No need to worry, you can reset your password by clicking the link below: Reset password Your username is: {userName}. If you didn't request a password reset, feel free to delete this email. Thanks, FintechOS Team.

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 key to the web.config file

On the server where the FintechOS installation package resides, go to the web.config file and provide the name of the automation script name which customizes the email template, by adding the following key:

```
<appSettings>
        <add
key="ResetPasswordEmailTemplateWorkflowName" value="WorkflowName"/>
        ...
</appSettings>
```

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.

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 *Innovation 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 key to the web.config file

On the server where the FintechOS installation package resides, go to the web.config file and provide the name of the automation script name which configures the password complexity, by adding the following key:

```
<appSettings>
        <add
key="ResetPasswordRulesWorkflowName" value="WorkflowName"/>
        ...
</appSettings>
```

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.

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 *Innovation 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");
```

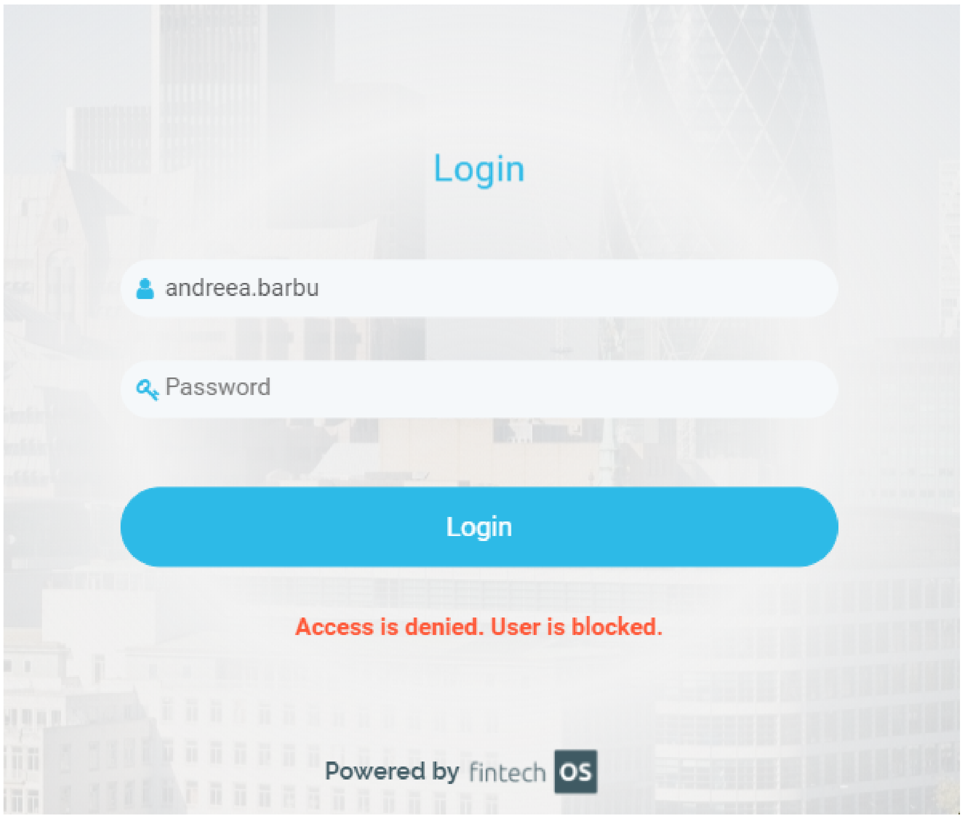
Context contains two keys in the **Values** property:

- 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.



Access is denied. User is blocked.

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 - Web.config setup

In order to create the setup, head over to installation files of the FintechOS environment where you wish to make the necessarily modifications, and in the web.config file add the following configurations.

To configure the the maximum amount of retries (the default value is zero):

<add key="core-setting-ebsauth-account-lockout-duration" value="5"/>

<add key="feature.reset-password" value="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 136 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.

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 provides you with two SDK functions which enable you to identify any inactive FintechOS users and disable their access as an extra security measure for protecting your FintechOS resources against unauthorized access:

- `inactiveUsers(int daysOfInactivity)` - get the list of users who have not been active in FintechOS 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:

```
<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, use the following web.config keys:

```
<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"/>
```

Key	Description
`feature-upload-malware-detection`	Set to 1 to enable anti-malware scanning or 0 to disable it.
`feature-upload-malware-use-remote`	Set to 1 to use a remote scan engine (see below) or 0 to use the local anti-malware engine. The local anti-malware engine is available only for on-premise deployments. For cloud deployments, you can use only the remote scan engine. Default: 0. **NOTE** Currently, the Kaspersky Scan Engine is the only remote scan engine supported. Additional scan engines may be added in the future.
`feature-upload-malware-endpoint`	(Remote scan engine only) API endpoint of the remote scan engine.
`feature-upload-malware-apikey`	(Remote scan engine only) Subscription key for the remote scan engine.
`feature-upload-malware-timeout`	(Remote scan engine only) Duration in minutes to wait for a response from the remote scan engine before rejecting the file. Default: 30.

Data Audit

The forth pillar of FintechOS 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 has an extensive audit functionality that can be enabled for any entity, allowing change tracking at entity level.

Using the Innovation 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 Logging

FintechOS 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 to the **web.config** file and set the feature-universal-logging setting, as desired. By default, it is set to **0**, that is, the feature is enabled.:

```
<configuration>
      <appSettings>
                ...
                <add key="feature-universal-
logging" value="0|1|true|false"/>
      </appSettings>
</configuration>
```

FintechOS API Logging

FintechOS logs the calls over the FintechOS API (REST AND WCF) and DataService CRUD operations.

The logs are saved by default in a separate database schema named EbsLogs. Database administrators can restrict read access for this schema and grant insert only rights for the SQL login used by the FintechOS platform.

Source names:

- OpenApi (REST endpoint)
- ApiService (WCF endpoint)
- DataService (MVC endpoint)

EbsLogs.ApiLog Schema

Field	Type	Description
Id	bigint	identity, primary key
LogId	uniqueidentifier	alternate unique key
Tenant	nvarchar(150)	tenant name, default value: ebs_default
UserName	nvarchar(200)	authenticated user name
Source	nvarchar(150)	controller name : OpenApi, ApiService or DataService
Method	nvarchar(150)	action name
Request	nvarchar(max)	request parameter as JSON
Response	nvarchar(max)	response as JSON
Message	nvarchar(max)	response message
Exception	nvarchar(max)	response error
Success	bit	success/error
CreatedAtUtc	datetime	call moment UTC
Duration	bigint	call duration milliseconds
CorrelationId	nvarchar(100)	correlation id
RequestId	nvarchar(100)	request id
ApiInfo	nvarchar(max)	call authentication information

How to Configure the FintechOSAPI Logging

To configure this feature, go to the **web.config** file and use a custom configuration section, as provided below:

```
<configuration>
     <configSections>
             <section name="ftosApiLogging"
```

```
type
="EBS.Core.Utils.ApiLoggingConfiguration.ApiLoggingConfigSection,
EBS.Core.Utils"/>
     </configSections>
     <ftosApiLogging enabled="true|false">
                <sources>
                    <source
name="OpenApi|ApiService|DataService" exclude="true|false">
                        <methods>
                            <method name="*">
                                <input exclude="true|false">
                                </input>
                            </method>

<method name="GetById" exclude="true|false">
                                <input exclude="true/false">
                                    <properties>
                                        <property
name="A" exclude="true|false"/>
                                    </properties>
                                </input>
                                <output exclude="true|false">
                                    <properties>
                                        <property
name="B" exclude="true|false"/>
                                    </properties>
                                </output>
                            </method>
                        </methods>
                    </source>
                </sources>
       </ftosApiLogging>
 </configuration>
```

The configuration allows filtering the out from logging elements at different levels of granularity: source, method (action), input (request), output (result), input property, output property.

The user can configure all other methods of a source by specifying "*" for method name. Any explicitly defined method will override all settings from "*".

NOTE When a property is excluded, it will not be serialized in the log.

Restrict Access to Innovation Studio Based on the StudioUser Role

(v21.2.2.4 and later)

Apart from the security roles that limit the access to specific features within Innovation Studio, an additional layer of security is provided by preventing users from logging in to Innovation Studio based on the StudioUser role.

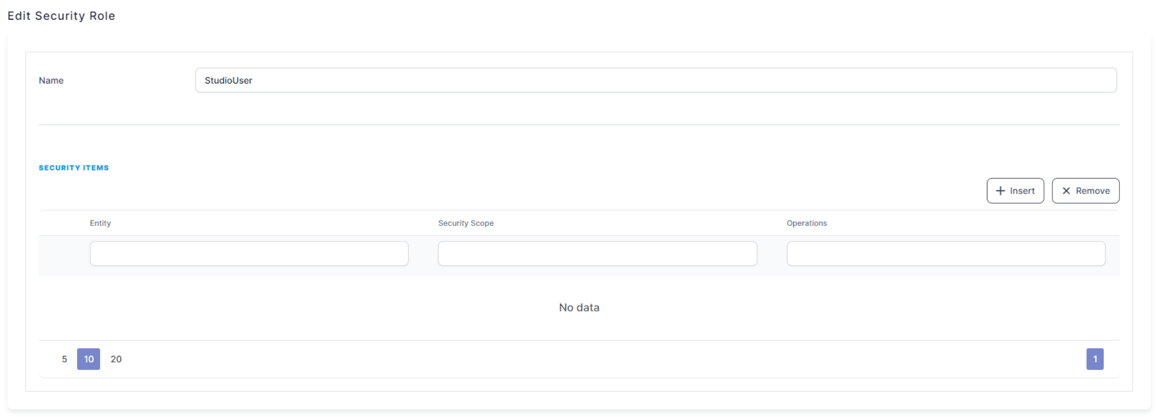
For this, in the *web.config* file, go to the `<appSettings>` section and add the following key

```
<add key="feature-restrict-access-in-studio-to-studio-user-
role" value="1" />
```

Setting	Description
0	All users can log to Innovation Studio regardless of the assigned security roles. This is the default behavior.
1	Only users with the StudioUser role can log in to Innovation Studio

This key is optional and, if not added, the behavior is equivalent to it being set to 0.

Keep in mind that the StudioUser role must be created and assigned to users manually.


Edit Security Role

Name: StudioUser

SECURITY ITEMS