IBM Financial Transaction Manager v4.0.4.1 Security Configuration

© Copyright International Business Machines Corporation 2022.

US Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Table of Contents

[1 Overview 6](#_Toc106713065)

[2 Communication - encryption 7](#_Toc106713066)

[2.1 Communication – encryption overview 7](#_Toc106713067)

[2.2 Encrypted and non-encrypted traffic details 7](#_Toc106713068)

[2.3 FTM certificates and pre-requisite 11](#_Toc106713069)

[2.3.1 Overview of FTM certificates and secrets 11](#_Toc106713070)

[2.3.2 Delete Existing Secrets 12](#_Toc106713071)

[2.3.3 Updating keystore/truststore credentials 12](#_Toc106713072)

[2.4 Installing and Configuring Certificate Manager 12](#_Toc106713073)

[2.4.1 Install IBM Cloud Platform Common Services 12](#_Toc106713074)

[2.4.2 Install Certificate Manager 12](#_Toc106713075)

[2.5 Generating FTM certificates 14](#_Toc106713076)

[2.5.1 Db2 Certificates 14](#_Toc106713077)

[2.5.2 Common FTM certificate using certificate manager 18](#_Toc106713078)

[2.5.3 Common FTM Certificate (without certificate manager) 20](#_Toc106713079)

[2.5.4 MQ Config Files 20](#_Toc106713080)

[2.5.5 MQ LDAP Authentication 21](#_Toc106713081)

[2.5.6 ACE Config Files 22](#_Toc106713082)

[2.5.7 Liberty Certificates 24](#_Toc106713083)

[2.5.8 Additional Certificates 24](#_Toc106713084)

[2.5.9 Keep the files at secure location 24](#_Toc106713085)

[2.6 Enable SSL for Db2 Communication 25](#_Toc106713086)

[2.6.1 Configure Db2 for SSL 25](#_Toc106713087)

[2.6.2 Add Db2 public certificate to OCP secret 25](#_Toc106713088)

[2.7 Enable SSL for FTM Components 25](#_Toc106713089)

[2.8 Enable SSL for FTM Immediate Payments Components 25](#_Toc106713090)

[2.9 Enable SSL for FTM Digital Payments/Interac/Check Components 26](#_Toc106713091)

[2.9.1 FTM Digital Payments/Interac/Check JSE Components SSL enablement 26](#_Toc106713092)

[2.9.2 FTM Digital Payments/Interac/Check enable SSL for MQ 28](#_Toc106713093)

[2.9.3 FTM Digital Payments/Interac enable SSL for ACE 28](#_Toc106713094)

[2.9.4 FTM Digital Payments/Interac/Check enable SSL post deployment 29](#_Toc106713095)

[2.10 Enable mTLS 29](#_Toc106713096)

[2.10.1 Enable mTLS during deployment 30](#_Toc106713097)

[2.10.2 Enable mTLS post deployment 30](#_Toc106713098)

[3 Certificate Management 30](#_Toc106713099)

[3.1 Certificate management overview 30](#_Toc106713100)

[3.2 Self-Signed Certificates 31](#_Toc106713101)

[3.2.1 Renewing self-signed certificate 31](#_Toc106713102)

[4 Certificate – Add IBM License Service certificate to FTM keystore 31](#_Toc106713103)

[5 Certificate – Add LDAP certificate to FTM keystore 33](#_Toc106713104)

[6 Network Ports 34](#_Toc106713105)

[6.1 Network Ports - Overview 34](#_Toc106713106)

[6.2 Network Policies Used for FTM 35](#_Toc106713107)

[6.2.1 Db2 [FTM/Immediate Payments/Digital Payments/Interac/Check] 35](#_Toc106713108)

[6.2.2 MQ and ACE [FTM/Immediate Payments/Digital Payments/Interac/Check] 36](#_Toc106713109)

[FTM- OAC [FTM/Immediate Payments/Digital Payments/Interac/Check] 38](#_Toc106713110)

[6.2.3 FTM Digital Payments/Interac/Check – Gateway Server 39](#_Toc106713111)

[6.2.4 FTM Digital Payments/Interac/Check – Business Rules Server 41](#_Toc106713112)

[6.2.5 FTM Digital Payments/Interac/Check – Business Rules Manager 42](#_Toc106713113)

[6.2.6 FTM Digital Payments/Interac/Check – Transaction Server 43](#_Toc106713114)

[6.2.7 FTM Digital Payments/Interac/Check – JEE Components 44](#_Toc106713115)

[6.2.8 FTM Default Deny Policy 45](#_Toc106713116)

[6.2.9 FTM Allow Traffic to Internal IP Ranges 46](#_Toc106713117)

[6.2.10 FTM Allow Traffic to Internal IP Ranges 46](#_Toc106713118)

[7 Liberty transport related security configurations 47](#_Toc106713119)

[7.1 Secure Cookies 47](#_Toc106713120)

[7.2 TLS Communication 48](#_Toc106713121)

[7.2.1 Restricting TLS 48](#_Toc106713122)

[7.2.2 Restrict the supported cipher suites 48](#_Toc106713123)

[7.3 Configure X-Powered-By header property 49](#_Toc106713124)

[7.4 Configure logoutOnHttpSessionExpire property 49](#_Toc106713125)

[8 Liberty Authentication and Authorization 49](#_Toc106713126)

[8.1 Authentication - Basic user Registry 49](#_Toc106713127)

[8.2 Authentication – Custom user registry 50](#_Toc106713128)

[8.3 Authentication – CLIENT\_CERT 52](#_Toc106713129)

[8.4 Authorization – FTM OAC 54](#_Toc106713130)

[8.5 Authorization – FTM Digital Payments/Interac/Check Control Center 54](#_Toc106713131)

[9 FTM Digital Payments/Interac/Check JSE Components – Encrypting configuration files 54](#_Toc106713132)

[9.1 Default deployment with JSE files encrypted 54](#_Toc106713133)

[9.2 IBM Encrypting configuration files - Overview 56](#_Toc106713134)

[9.3 Encrypting properties with placeholders 56](#_Toc106713135)

[9.4 Locating and configuring the utility 58](#_Toc106713136)

[9.5 Running the utility 58](#_Toc106713137)

[10 Appendix 58](#_Toc106713138)

[10.1 Restart all Pods 58](#_Toc106713139)

[10.2 Creating custom SSL certificates (Pre-4.0.4) 59](#_Toc106713140)

[10.2.1 Db2 Certificates 60](#_Toc106713141)

[10.2.2 MQ Certificates 60](#_Toc106713142)

[10.2.3 MQ LDAP Authentication 62](#_Toc106713143)

[10.2.4 ACE Certificates 63](#_Toc106713144)

[10.2.5 Liberty Certificates 65](#_Toc106713145)

[10.2.6 Keep the files at secure location 66](#_Toc106713146)

[11 References 68](#_Toc106713147)

[12 Notices 68](#_Toc106713148)

[12.1 Trademarks 69](#_Toc106713149)

[12.2 Terms and conditions for product documentation 70](#_Toc106713150)

[12.2.1 Applicability 70](#_Toc106713151)

[12.2.2 Personal use 70](#_Toc106713152)

[12.2.3 Commercial use 70](#_Toc106713153)

[12.2.4 Rights 70](#_Toc106713154)

Changes History

|  |  |  |
| --- | --- | --- |
| **Version** | **Date** | **Description** |
| 0.01 | 9 September 2020 | Initial document |
| 0.02 | 7 December 2020 | Added “Configure logoutOnHttpSessionExpire property” |
| 0.03 | 9 February 2021 | Updated for 4.0.3 changes |
| 0.04 | 12 March 2021 | Updated for feedback from team |
| 0.05 | 18 March 2021 | Updated copyright year |
| 0.06 | 19 March 2021 | Updated for CA signed certificates |
| 0.07 | 12 April 2021 | Updated for CLIENT\_CERT authentication for web services. |
| 0.08 | 19 April 2021 | Updated CLIENT\_CERT authentication with basic registry information. |
| 0.09 | 14 June 2021 | Updated for Certificate Manager integration |
| 0.10 | 25 September 2021 | Updated JSE SSL configuration |
| 0.11 | 15 November 2021 | Updated CLIENT\_CERT authentication details for all Liberty components. |
| 0.12 | 15 February 2022 | Updated feedback from team |
| 0.13 | 21 June 2022 | Updated for network policies |

# Overview

This document augments the [IBM FTM documentation in IBM Knowledge Cente](http://www.ibm.com/support/knowledgecenter/SSRH46/welcomeSSRH46.html)*[r](http://www.ibm.com/support/knowledgecenter/SSRH46/welcomeSSRH46.html)* security configuration sections.

Financial Transaction Manager v4.0 consists of four separately downloadable products:

* IBM Financial Transaction Manager for Red Hat OpenShift
  + In this document referred to as FTM.
* IBM Financial Transaction Manager for Immediate Payments for Red Hat OpenShift
  + In this document referred to as FTM Immediate Payments or Immediate Payments.
* IBM Financial Transaction Manager for Digital Payments for Red Hat OpenShift
  + In this document referred to as FTM Digital Payments or Digital Payments.
* IBM Financial Transaction Manager for Interac e-Transfers for Red Hat OpenShift
  + In this document referred to as FTM Interac e-Transfers.
* IBM Financial Transaction Manager for Check Services for Red Hat OpenShift
  + In this document referred to as FTM Check.

When deploying the Financial Transaction Manager (FTM) v4.0 family of products there are several security configurations required to fully secure the deployment. Refer to the IBM FTM documentation in IBM Knowledge Center for general product information and product security information:

* FTM: IBM FTM documentation in IBM Knowledge Center section [OAC Security](https://www.ibm.com/support/knowledgecenter/en/SSRH46_4.0.0/fxhinstoacsecurity.html) within the products section.
* FTM Immediate Payments: IBM FTM documentation in IBM Knowledge Center section [OAC Security](https://www.ibm.com/support/knowledgecenter/en/SSRH46_4.0.0/fxhinstoacsecurity.html) within the products section.
* FTM Digital Payments/Interac/Check: IBM FTM documentation in IBM Knowledge Center section [Security](https://www.ibm.com/support/knowledgecenter/en/SSRH46_4.0.0/chosecurityref.html) within the products section.

The specific steps to perform will depend on your requirements. Refer to the IBM FTM documentation in IBM Knowledge Center for additional product information.

Deployed products are complete in the sense that the dependency products will be deployed automatically. Dependency FTM products are included as follows:

* FTM requires no other FTM products.
* FTM Immediate Payments will include a customized FTM.
* FTM Digital Payments/Interac will include a customized Immediate Payments and FTM.

The general security configuration requirements for each product will be the same regardless if one product is included in another product. For this reason, information in this document will include all three products. By default, some security will already be enabled.

Some configuration files are located on Persistent Volumes (PV). This prevents your existing configuration files from being overridden if the containers are updated.

The FTM Digital Payments/Interac/Check product consists of both Java SE (JSE) components and Java Enterprise Edition (Java EE, JEE) components so there may be configurations for each type of Java component. Note that JSE was formerly J2SE and JEE formerly J2EE so there will be references using these former designations.

Many steps require use of the OpenShift Container Platform command-line interface (CLI). CLI commands start with oc.

General configuration steps are:

1. Decide on communication encryption requirements.

**Note:** for the Control Center access TLS needs to be enabled. For specific encryption configuration information refer to sections 2Communication - encryption and 3Certificate Management.

* 1. Related section 7Liberty transport related security configurations.

1. Internet Protocol (IP) ports are restricted. For further information refer to section 6Network Ports.
2. Authentication and Authorization configurations. Refer to section 8Liberty Authentication and Authorization.
3. Encrypting FTM Digital Payments/Check JSE component configuration files. Refer to section 9FTM Digital Payments/Interac/Check JSE Components – Encrypting configuration files.

# Communication - encryption

## Communication – encryption overview

This section provides the details of the encrypted versus non-encrypted traffic within FTM components and how to enable the encryption within the components where sensitive data is involved. Sensitive data is defined as possibly containing personally identifiable information (PII), for example account numbers and addresses.

**Note:** TLS and SSL are used interchangeably. SSL tends to be a more common term so will be used more often.

## Encrypted and non-encrypted traffic details

FTM application consists of multiple pods. Some communication between the pods is encrypted over SSL/TLS and some is not encrypted.

**Communication from JSE, JEE, App Connect to Db2 is encrypted.**

***Type of Data:*** The communication with JSE, JEE, App Connect flows to Db2 includes sensitive personal account information. This communication can be encrypted by customer during or post deployment (as it requires Db2 to be enabled for encryption before installing FTM).

**Communication between all JEE components is encrypted.**

***Type of Data:*** IIOP for remote EJB calls. This traffic is already encrypted by default.

**Communication between JSE components is encrypted.**

***Type of Data:*** Communication between Gateway Server and Business Rules Server includes sensitive personal account information. This communication can be encrypted by customer during or post deployment.

All other communications do not involve any personal data, but rather IDs of internal objects (not Personally Identifiable II data) via MQ message.

**Communication from JSE components to JEE is encrypted.**

***Type of Data:*** IIOP for remote EJB calls. This traffic is already encrypted by default.

**Communication from JEE components to JSE is encrypted.**

***Type of Data:*** Communication between Gateway Engine and Business Rules Server includes sensitive data. This communication can be encrypted by customer during or post deployment.

All other communications do not involve any sensitive data, but rather IDs of internal objects (not sensitive data) via MQ msg.

**Communication from JSE, JEE to MQ is encrypted.**

***Type of Data:*** The communication with MQ does not involve any sensitive data but IDs of internal objects.

| To  From | db2 (external) \*\* | mq \*\* | gw-server | brm | brs | its | oac \*\* | approvals-engine | business-rules-engine | control-center-ui | distribution-engine | gateway-engine | risk-engine | rtp-engine | services-framework-engine | settlement-engine | webservices-engine | webservices-br | webservices-gw | noc-engine | vetting-engine | webservices-pfs | webservices-pfs-txt |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| db2 (ext) \*\* | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- |  |
| mq \*\* | *Ys* | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- |  |
| gw-server | *Ys* | Y | -- | -- | *Ys* | -- | -- | -- | -- | -- | -- | Y | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- |  |
| brm | *Ys* | -- | -- | -- | -- | **--** | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- |  |
| brs | *Ys* | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- |  |
| its | *Ys* | Y | N | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- |  |
| oac \*\* | *Ys* | Y | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- |  |
| approvals-engine | *Ys* | Y | -- | -- | *Ys* | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- |  |
| business-rules-engine | *Ys* | Y | -- | *Ys* | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- |  |
| control-center-ui | *Ys* | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y | Y |  |
| distribution-engine | *Ys* | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- |  |
| gateway-engine | *Ys* | Y | -- | -- | *Ys* | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- |  |
| risk-engine | *Ys* | Y | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- |  |
| rtp-engine | *Ys* | Y | -- | -- | *Ys* | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- |  |
| services-framewrok-engine | *Ys* | Y | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- |  |
| settlement-engine | *Ys* | Y | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- |  |
| webservices-engine | *Ys* | Y | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- |  |
| webservices-br | *Ys* | Y | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- |  |
| webservices-gw | *Ys* | Y | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- |  |
| noc-engine | *Ys* | Y | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- |  |
| vetting-engine | *Ys* | Y | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- |  |
| webservices-pfs | *Ys* | Y | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- |  |
| webservices-pfs-txn | *Ys* | Y | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- |  |

**Legend:**

Y - Communication is encrypted. No sensitive data.

*Ys* - Communication is encrypted. Contains sensitive data.

N **-** Communication is not encrypted. No sensitive data.

*Ns* **-** Communication is not encrypted. Contains sensitive data.

**-- -** No communication between the containers.

\*\* - FTM and FTM Immediate Payments offerings will have only these components. FTM Digital Payments/Check offering has all the components mentioned in the table.

## FTM certificates and pre-requisite

### 2.3.1 Overview of FTM certificates and secrets

It is recommended to use different set of certificates, even if self-signed, for every deployment. Generally, the self-signed certificates are allowed in development/sandbox deployments. This section provides the steps to create various certificates used by FTM.

Below is the list of OpenShift secrets and their contents used by FTM Pods. The below list will help you to understand which secrets and which files inside it you need to update when using your own CA signed certificates.

There is only one OCP secret **ftm-ssl-cert-secret** that holds the certificates used by all FTM components.

|  |  |  |
| --- | --- | --- |
| **OpenShift Secret** | **Files in the secret** | **FTM Pods using the secret** |
| ftm-db2-ssl-cert-secret | * ca.crt – CA public certificate * tls.key – Db2 certificate private key * tls.crt - Db2 certificate * mydbserver.arm – Db2 certificate. Same as tls.crt. FTM currently expects the Db2 certificate with this specific file name.   The same OCP secret can be used by the Db2U instances. | MQ, ACE, JEE, JSE |
| ftm-ssl-cert-secret | * tls.key – SSL key * tls.crt – SSL certificate   (This is new OCP secret in 4.0.4 that will contain the application key and certificate) | MQ, ACE, JEE, JSE |
| ftm-mq-ssl-cert-secret | * mqwebuser.xml – MQ web console user configuration   (This is existing OCP secret from 4.0.3. But in 4.0.4, it will not have separate MQ key and certificate in it. It will only have Web user config files) | MQ |
| ftm-ace-ssl-cert-secret | * \*-users.txt - (additional user configuration files)   (This is existing OCP secret from 4.0.3. But in 4.0.4, it will not have separate ACE key and certificate in it. It will only have ACE user config files) | ACE |
| ftm-additional-certs-secret (This is new) | * mqserver-ca.crt – CA or root CA certificate. This is required only in case you are using CA to create the FTM certificates (mostly in the customer production environments)   (This is existing OCP secret from 4.0.3 and has the same behavior.) | MQ, ACE, JEE, JSE |

### 2.3.2 Delete Existing Secrets

If you already have the FTM OCP secrets created from previous release, and want to switch to this new approach, please delete the below FTM OCP secret.

**Note:** Please make sure you have backup of the existing OCP secrets and/or the files in the secrets in case you need to revert the changes. The OCP secrets can be backed up using the below command.

# oc get secret <secret-name> -o yaml > <secret-name>.yml

This will create a Yaml file with the secret definition. Please remove the instance specific fields from the yaml and you can use them to restore the secrets later on if needed.

Secrets to delete:

* ftm-mq-ssl-cert-secret
* ftm-ace-ssl-cert-secret
* liberty-ssl-cert-secret

You can delete the secrets from OCP UI or oc CLI.

# oc delete secret <secret-name>

### 2.3.3 Updating keystore/truststore credentials

The FTM j2ee and j2se components use keystore/trust store in jks/pkcs12 format to store the certificates. The certificates generated for FTM are imported into the keystore/truststore for use by these components.

You must set the KEYSTORE\_PASSWORD and TRUSTSTORE\_PASSWORD variables in OCP secret *ftm-application-secret* to set the keystore and truststore password respectively

## Installing and Configuring Certificate Manager

Installation of IBM Cloud Platform Common Services and Certificate Manager is required if you plan to use certificate manager to create the SSL certificates for FTM components.

### 2.4.1 Install IBM Cloud Platform Common Services

The IBM Cloud Platform Common Services installation is a prerequisite for FTM operators. The installation of these prerequisites is documented at

<https://www.ibm.com/docs/en/ftmfm/4.0.3?topic=byifoo-install-cloud-platform-common-services-operator-licensing-operator-operand>

### 2.4.2 Install Certificate Manager

Individual common services are installed by creating an *OperandRequest* specifying which services to be installed. All these common services will be deployed into the namespace ibm-common-services, which cannot be changed.

Create a new *OperandRequest* to enable the Certificate Manager service if it does not already exist.

1. Login to OCP UI.
2. Go to *Home* -> *Search*.
3. Select the namespace as ibm-common-services and search for keyword *operandrequest* by clicking on Resources dropdown.

Graphical user interface, application

Description automatically generated

1. If you see any existing entry, please check if it already has Certificate Manager enabled.

Graphical user interface, text, application, email

Description automatically generated

* 1. Click on the entry and switch to YAML tab.

Text

Description automatically generated

* 1. In the above example, the Certificate Manager service is not enabled. Only Licensing service is enabled.
  2. Add the following entry for Certificate Manager under spec.requests.operands and save the changes.

- name: ibm-cert-manager-operator

Text

Description automatically generated Graphical user interface, text, application

Description automatically generated

1. If there is no entry for *OperandRequest*, you can create a new one with below yaml.

**Note:** Since FTM operator deployment requires Licensing service of IBM Cloud Platform Common Services to be there, you will most probably always have the existing entry as mentioned in the previous step.

apiVersion: operator.ibm.com/v1alpha1

kind: OperandRequest

metadata:

name: common-service

namespace: ibm-common-services

spec:

requests:

- operands:

**- name: ibm-cert-manager-operator**

- name: ibm-licensing-operator

registry: common-service

1. Wait for 3-4 minutes.
2. You should see the IBM Cert Manager operator running in ibm-common-services namespace.

Graphical user interface, application

Description automatically generated

## Generating FTM certificates

### 2.5.1 Db2 Certificates

FTM does not have Db2 database bundled. User needs to have the Db2 pre-deployed before deploying FTM. The Db2 can be external to the OCP cluster or it can be in deployed in the OpenShift using Db2 Operator. For FTM it is always treated as an external database.

FTM supports the TLS encrypted communication with Db2 database. Please configure the Db2 to run over TLS by following the Db2 documentation.

**Note:** FTM supports only one Db2 certificate. So, if you are using two different Db2 instances for FTM database and DupDetect database (in case of FTM for Check deployment), make sure both the instances are configured with the same SSL certificate.

1. Create a directory where Db2 certificate will be placed for importing into the OCP secret.

mkdir -p /var/tmp/db-certs

cd /var/tmp/db-certs

#### Generate Db2 Certificate for traditional Db2 installations

If you are using traditional Db2 database deployed external to the FTM deployment, below are the steps to create Db2 keydb and certificate using GSKit.

**Note:** These steps are from the FTM development team used during development and testing. Please follow the Db2 product documentation for configuring the SSL on production Db2.

**Note:** Creating custom Db2 certificate is an optional step. You don’t need to create the custom Db2 certificates if you are not configuring Db2 to use TLS during FTM deployment. This may be the case in sandbox/development environments.

You should have a GSKit installed to create Db2 certificates. GSKit comes by default with Db2 installation. Here we are using the GSKit from the Db2 installation.

**Note:** The Db2 path is assumed to be /opt/ibm/db2/V11.5.6

1. Login to the Db2 server as instance owner db2inst1 (or any other Db2 instance owner based on your environment).
2. Create a directory where Db2 certificates will be placed.

mkdir -p /var/tmp/db-certs

cd /var/tmp/db-certs

1. Set Db2 command path variable for convenience. Use different value as per your Db2 version and install path. (Change the path value in case Db2 is installed at different location)

export MY\_DB2\_PATH=/opt/ibm/db2/V11.5.6

1. Create a key database.

$MY\_DB2\_PATH/gskit/bin/gsk8capicmd\_64 -keydb -create -db "myserverdb.kdb" -pw "ibmftm" -stash

The password ibmftm is the key database password. It is not used anywhere in the application; it is used just to access the key database when adding/removing the Db2 keys into it.

**Note:** This is just an example password and you should substitute your own password in its place when performing this step.

Make sure below files are created,

ls -al  
total 16  
drwxrwxr-x 2 db2inst1 db2inst1  94 Jan  4 11:08 .  
drwxrwxrwt 1 root     root      22 Jan  4 11:01 ..  
-rw----- 1 db2inst1 db2inst1  88 Jan  4 11:08 myserverdb.crl  
-rw----- 1 db2inst1 db2inst1  88 Jan  4 11:08 myserverdb.kdb  
-rw----- 1 db2inst1 db2inst1  88 Jan  4 11:08 myserverdb.rdb  
-rw----- 1 db2inst1 db2inst1 193 Jan  4 11:08 myserverdb.sth

1. Create a key pair and add into the key database.

$MY\_DB2\_PATH/gskit/bin/gsk8capicmd\_64 -cert -create -db "myserverdb.kdb" -pw "ibmftm" -label "myselfsigned" -dn "CN=ftm-db2.ibm.com,O=ibm, OU=ftm,L=Austin,ST=TX,C=US" -size 2048 -sigalg SHA256\_WITH\_RSA

Here, you need to give the same key database password to *-pw* option. The *-dn* can be given as per your organization structure.

Notice the change in .kdb file size as we added the new key into it.

ls -al  
total 20  
drwxrwxr-x 2 db2inst1 db2inst1   94 Jan  4 11:08 .  
drwxrwxrwt 1 root     root       22 Jan  4 11:01 ..  
-rw----- 1 db2inst1 db2inst1   88 Jan  4 11:08 myserverdb.crl  
-rw----- 1 db2inst1 db2inst1 5088 Jan  4 11:17 myserverdb.kdb  
-rw----- 1 db2inst1 db2inst1   88 Jan  4 11:08 myserverdb.rdb  
-rw----- 1 db2inst1 db2inst1  193 Jan  4 11:08 myserverdb.sth

1. Extract the public certificate part from the key pair.

$MY\_DB2\_PATH/gskit/bin/gsk8capicmd\_64 -cert -extract -db "myserverdb.kdb" -pw "ibmftm" -label "myselfsigned" -target **"mydbserver.arm"** -format ascii -fips

Here, you need to give the same key database password to *-pw* option.

ls -al  
total 24  
drwxrwxr-x 2 db2inst1 db2inst1  116 Jan  4 11:21 .  
drwxrwxrwt 1 root     root       22 Jan  4 11:01 ..  
**-rw-rw-r-- 1 db2inst1 db2inst1 1281 Jan  4 11:21 mydbserver.arm**  
-rw----- 1 db2inst1 db2inst1   88 Jan  4 11:08 myserverdb.crl  
-rw----- 1 db2inst1 db2inst1 5088 Jan  4 11:17 myserverdb.kdb  
-rw----- 1 db2inst1 db2inst1   88 Jan  4 11:08 myserverdb.rdb  
-rw----- 1 db2inst1 db2inst1  193 Jan  4 11:08 myserverdb.sth

1. File mydbserver.arm is the public certificate that needs to be added into the OCP secret.

**Note:** Do not change the filenames as these are referenced in the images.

1. Copy the Db2 certificate mydbserver.arm is to the folder /var/tmp/db-certs on the workstation.

You will have below files under /var/tmp/db-certs.

* mydbserver.arm (Certificate)

#### Certificates for Db2U instances

If you are using Db2U (Db2 on Red Hat OpenShift), below are the steps to either get the existing certificates from Db2U instance or use the pre-created certificates to the Db2U instance.

Create a directory where Db2 certificates will be placed.

mkdir -p /var/tmp/db-certs

cd /var/tmp/db-certs

##### Getting Certificate from pre-deployed SSL enabled Db2U instance

If you are using pre-deployed Db2U (Db2 on Red Hat OpenShift) and SSL is already enabled, you can get the Db2 certificate from location /mnt/blumeta0/db2/ssl\_keystore/rootCA.pem

Copy the Db2 certificate rootCA.pem as mydbserver.arm into the folder /var/tmp/db-certs on the workstation.

You will have below files under /var/tmp/db-certs.

* mydbserver.arm (Certificate)

##### Generating certificates using IBM certificate manager while creating new Db2u instance(s)

Create application certificate using application certificate issuer to be used for Db2U.

Import the following YAML from OCP UI and click **Create**.

apiVersion: certmanager.k8s.io/v1alpha1

kind: Certificate

metadata:

name: ftm-deployment-tls-db2u

namespace: *<FTM-namespace>*

spec:

secretName: ftm-ssl-cert-secret-db2u

issuerRef:

name: **ftm-deployment-tls** **# Should match the name of issuer created earlier**

kind: Issuer

**commonName: "db2u-cert"**

dnsNames:

- *<DNS-Domain>*

where, FTM-namespace – Namespace where FTM is deployed.

DNS-Domain – Domain for which the application certificates will be generated. For example, ibm.com

Once above Yaml is imported, OCP will create a OCP secret named ftm-ssl-cert-secret-db2u and it will have the tls.key and tls.crt to be used by FTM. The secret will also have ca.crt, the CA root certificate in case needed by some components.

Check the certificate using oc command and make sure it has ca.crt, tls.crt and tls.key files.

$ oc get secret ftm-ssl-cert-secret-db2u -o yaml.

You will need to then export these files from the OCP secret and decode them using base64 command.

##### Using pre-created certificate while creating new Db2U instance(s)

If you already have the pre-created SSL certificates, copy all the certificate files (certificate, private key and CA public certificate) to folder /var/tmp/db-certs. You can use an authorized CA in your environment to generate the certificate.

Create a copy of certificate file tls.crt as mydbserver.arm. This is because FTM refers the Db2 certificate with this specific filename.

You will have below files under /var/tmp/db-certs.

* ca.crt (CA public certificate)
* tls.crt (Certificate)
* tls.key (Private key)
* mydbserver.arm (Copy of tls.crt)

Please follow the FTM deployment guide for the steps to create Db2U instance(s) using the OpenShift secret created in the next step. In case of FTM for Check, use same OpenShift secret for FTM database instance and DupDetect database instance.

#### Add Db2 Certificate into OpenShift Secret

Depending on how Db2 is configured with SSL, you will have one or more files under /var/tmp/db-certs on the workstation.

Create OpenShift secret in the project where FTM will be deployed. This example assumes the project name is ftm-demo.

oc project ftm-demo

oc delete secret **ftm-db2-ssl-cert-secret**

oc create secret generic **ftm-db2-ssl-cert-secret** --from-file=/var/tmp/db-certs

### Common FTM certificate using certificate manager

#### 2.5.2.1 Create the Self-Signed CA Issuers and CA Certs

The Certificate Manager is responsible for creating and managing the certificates used by applications. It can be configured to create the certificates using different issuers. We will be creating the self-signed issuer that will be used to create the CA certificate. Please check the Certificate Manager documentation at

<https://www.ibm.com/docs/en/cloud-paks/cp-management/1.3.0?topic=management-certificate-manager-cert-manager> for using other types of issuers.

**Create certificate issuer:**

Import the following YAML and click Create.

apiVersion: certmanager.k8s.io/v1alpha1

kind: Issuer

metadata:

name: **ftm-self-tls**

namespace: *<FTM-namespace>*

spec:

selfSigned: {}

where,

*FTM-namespace* – Namespace where FTM is deployed.

**Note:**

If you see the below error, wait for few minutes, and try again as it may take time to create the new CRDs that come with Certificate Manager

**An error occurred**

Error "failed calling webhook "webhook.certmanager.k8s.io": the server is currently unable to handle the request" for field "undefined".

**Create CA certificate:**

After you create the self-signed Issuer, create a CA certificate that references the self-signed Issuer and specifies the isCA field. Import the following YAML and click **Create**.

apiVersion: certmanager.k8s.io/v1alpha1

kind: Certificate

metadata:

name: ftm-ca-tls

namespace: *<FTM-namespace>*

spec:

secretName: ftm-ibm-cert-manager-ca-certs

isCA: true

issuerRef:

name: **ftm-self-tls** **# Should match the name of issuer created earlier**

kind: Issuer

dnsNames:

- *<DNS-Domain>*

where,

*FTM-namespace* – Namespace where FTM is deployed.

*DNS-Domain* – Domain for which the CA certificates will be generated. For example, ibm.com



#### Create Certificate Issuer for Application Certificates

**Create certificate issuer that uses the CA created above:**

Import the following YAML and click **Create**.

apiVersion: certmanager.k8s.io/v1alpha1

kind: Issuer

metadata:

name: **ftm-deployment-tls**

namespace: *<FTM-namespace>*

spec:

ca:

secretName: ftm-ibm-cert-manager-ca-certs

where,

*FTM-namespace* – Namespace where FTM is deployed.

Make sure both the issuers are in Ready state. This can be done by checking the Ready status using oc CLI.

$ oc get issuers

NAME READY

ftm-deployment-tls **True**

ftm-self-tls **True**

#### 2.5.2.3 Create Certificate for FTM using the Issuer

**Create FTM application certificate using the issuer created in earlier step:**

Import the following YAML and click **Create**.

apiVersion: certmanager.k8s.io/v1alpha1

kind: Certificate

metadata:

name: ftm-deployment-tls

namespace: *<FTM-namespace>*

spec:

secretName: ftm-ssl-cert-secret

issuerRef:

name: **ftm-deployment-tls** **# Should match the name of issuer created earlier**

kind: Issuer

**commonName: "ftm-cert"**

dnsNames:

- *<DNS-Domain>*

where,

*FTM-namespace* – Namespace where FTM is deployed.

*DNS-Domain* – Domain for which the application certificates will be generated. For example, ibm.com

**Note:**

The value of attribute commonName is important as it is part of the certificate’s *Subject* field. The CA public certificate ca.crt is created with default CommonName (CN) based on the dnsNames attribute. If we don’t specify the Common Name for application certificate, then it will be same as CA public certificate. This causes an issue while adding the CA public certificate and application certificate in the MQ key database due to same *Subject*. Hence, the Common Name for CA public certificate (ca.crt) and application certificate (tls.crt) should be different.

Once above Yaml is imported, OCP will create a OCP secret named ftm-ssl-cert-secret and it will have the tls.key and tls.crt to be used by FTM. The secret will also have ca.crt, the CA root certificate in case needed by some components.

Check the certificate using oc command and make sure it has ca.crt, tls.crt and tls.key files.

$ oc get secret ftm-ssl-cert-secret -o yaml

### Common FTM Certificate (without certificate manager)

This step is to create the common certificate that will be used by all FTM components.

#### Generate FTM Certificate

1. Login to the server where OpenSSL is installed.
2. Create a directory where FTM certificates will be placed.

mkdir -p /var/tmp/ftm-certs

cd /var/tmp/ftm-certs

1. Generate the self-signed certificates which will be used by FTM components

openssl req -newkey rsa:2048 -nodes -keyout **tls.key** -x509 -days 365 -out **tls.crt** -subj "/C=US/ST=TX/L=Austin/O=IBM/OU=FTM/CN=ftmserver"

**Note:** Do not change the filenames as these are referenced in the images.

#### Add FTM certificate into OpenShift Secret

Create OpenShift secret in the project where FTM will be deployed. This example assumes the project name is **ftm-demo**.

oc project ftm-demo

oc delete secret **ftm-ssl-cert-secret**

oc create secret generic **ftm-ssl-cert-secret** --from-file=/var/tmp/ftm-certs

### MQ Config Files

MQ will use the common certificate created in earlier step. This section is to add the additional configuration files in the OCP secrets.

#### MQ Web Console Users Configuration

1. Login to the server where OpenSSL is installed.
2. Create a directory where MQ files will be placed.

mkdir -p /var/tmp/mq-files

cd /var/tmp/mq-files

1. Create a mqwebuser.xml file in /var/tmp/mq-files directory based on below template. You can update the users and passwords as per your choice.

**Note**: Do not change **serverKeyAlias="mqservercert"** in **keyStore** Section**.** It is auto configured by the MQ pod.

<?xml version="1.0" encoding="UTF-8"?>

<server>

<featureManager>

<feature>appSecurity-2.0</feature>

<feature>basicAuthenticationMQ-1.0</feature>

</featureManager>

<enterpriseApplication id="com.ibm.mq.console">

<application-bnd>

<security-role name="MQWebAdmin">

<group name="MQWebAdminGroup" realm="defaultRealm"/>

</security-role>

<security-role name="MQWebAdminRO">

<user name="mqreader" realm="defaultRealm"/>

</security-role>

<security-role name="MQWebUser">

<special-subject type="ALL\_AUTHENTICATED\_USERS"/>

</security-role>

<security-role name="MFTWebAdmin">

<user name="mftadmin" realm="defaultRealm"/>

</security-role>

<security-role name="MFTWebAdminRO">

<user name="mftreader" realm="defaultRealm"/>

</security-role>

</application-bnd>

</enterpriseApplication>

<basicRegistry id="basic" realm="defaultRealm">

<user name="**mqadmin**" password="**mqadmin**"/>

<user name="**mqreader**" password="**mqreader**"/>

<user name="**mftadmin**" password="**mftadmin**"/>

<user name="**mftreader**" password="**mftreader**"/>

<group name="MQWebAdminGroup">

<member name="**mqadmin**"/>

</group>

</basicRegistry>

<variable name="httpsPort" value="9443"/>

<variable name="httpHost" value="\*"/>

<keyStore id="defaultKeyStore" location="key.jks" type="JKS" password="password"/>

<ssl id="thisSSLConfig" clientAuthenticationSupported="true" keyStoreRef="defaultKeyStore" **serverKeyAlias="mqservercert"** sslProtocol="TLSv1.2"/>

<sslDefault sslRef="thisSSLConfig"/>

</server>

Once these steps are done, you will have below files under /var/tmp/mq-files.

* mqwebuser.xml

#### Add MQ Web Console User Config into OpenShift Secret

Create OpenShift secret in the project where FTM will be deployed. This example assumes the project name is ftm-demo.

oc project ftm-demo

oc delete secret **ftm-mq-ssl-cert-secret**

oc create secret generic **ftm-mq-ssl-cert-secret** --from-file=/var/tmp/mq-files

**Note:** Update and confirm following value in the CR file (Operator Instance YAML) during the deployment.

Set mq\_enable\_ssl: true

Set qmgr\_cipherspec: TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA256

Set qmgr\_ciphersuite: 'SSL\_ RSA\_WITH\_AES\_128\_CBC\_SHA256'

Set qmgr\_channel: 'QMLDAP.SVRCONN'

### MQ LDAP Authentication

1. MQ can be configured to authenticate against an external LDAP. It is assumed that you have an LDAP running outside OpenShift.
2. Create a directory where MQ LDAP config file will be placed.

mkdir -p /var/tmp/mq-ldap

cd /var/tmp/mq-ldap

1. Delete the existing config map with name ftm-mq-**<INSTANCE\_NAME>**-mqsc-ini-config

oc project ftm-demo

oc delete configmap ftm-mq-**<INSTANCE\_NAME>**-mqsc-ini-config

1. Create a file ldap\_config.mqscfile in /var/tmp/mq-ldap directory with LDAP Configuration data. Below file is just a template for configuring an LDAP. Please replace highlighted values according to the LDAP server you are connecting to. You can refer IBM MQ documentation for details of each MQSC command.

ALTER QMGR CHLAUTH(ENABLED)

DEFINE CHANNEL**(QMLDAP.SVRCONN)** CHLTYPE(SVRCONN) TRPTYPE(TCP)

ALTER CHANNEL**(QMLDAP.SVRCONN)** CHLTYPE(SVRCONN) MAXMSGL(104857600)

DEFINE AUTHINFO(FTMQMGR.IDPW.LDAP) AUTHTYPE(IDPWLDAP) CONNAME(‘**myldap.server.com(389)**’) SHORTUSR(**‘uid’**) ADOPTCTX(YES) AUTHORMD(SEARCHGRP) BASEDNG(**‘ou=mqusers,dc=example,dc=com’**) BASEDNU(**‘dc=example,dc=com’**) CHCKCLNT(OPTIONAL) CHCKLOCL(NONE) CLASSGRP(**‘groupOfUniqueNames’**) CLASSUSR(**‘inetOrgPerson’**) FINDGRP(**‘uniqueMember’**) GRPFIELD(**‘cn’**) LDAPPWD(**‘password’**) LDAPUSER(**‘cn=read-only-admin,dc=example,dc=com’**) NESTGRP(YES) SECCOMM(NO) USRFIELD(‘uid’)

ALTER QMGR CONNAUTH(FTMQMGR.IDPW.LDAP)

REFRESH SECURITY

SET AUTHREC PROFILE(‘\*\*’) OBJTYPE(QMGR) GROUP(**‘ou=mqusers,dc=example,dc=com’**) AUTHADD(ALL)

SET AUTHREC PROFILE(‘\*\*’) OBJTYPE(queue) GROUP(**‘ou=mqusers,dc=example,dc=com’**) AUTHADD(ALL)

SET AUTHREC PROFILE(‘@class’) OBJTYPE(queue) GROUP(**‘ou=mqusers,dc=example,dc=com’**) AUTHADD(CRT)

SET AUTHREC PROFILE(‘\*\*’) OBJTYPE(topic) GROUP(**‘ou=mqusers,dc=example,dc=com’**) AUTHADD(ALL)

SET AUTHREC PROFILE(‘@class’) OBJTYPE(topic) GROUP(**‘ou=mqusers,dc=example,dc=com’**) AUTHADD(CRT)

1. Create OpenShift ConfigMap in the project where FTM will be deployed. This example assumes the project name is ftm-demo.

oc project ftm-demo

oc delete configmap ftm-mq-**<INSTANCE\_NAME>**-mqsc-ini-config

oc create configmap ftm-mq-**<INSTANCE\_NAME>**-mqsc-ini-config --from-file=/var/tmp/mq-ldap

1. Make sure you have QMGR\_CHANNEL value in CR file is consistent with this file. Once the MQ pod is re-created, it will run these MQSC commands and configure LDAP authentication.  
   Ex: **QMLDAP.SVRCONN**

### ACE Config Files

ACE will use the common certificate created in earlier step. This section is to add the additional configuration files in the OCP secrets.

#### ACE Web Console User Configuration

1. Login to the server where OpenSSL is installed.
2. Create a directory where ACE certificates will be placed.

mkdir -p /var/tmp/ace-files

cd /var/tmp/ace-files

1. ACE console can be configured with users and can provide roles to the users. You can create one or more types of users based on your choice.

Use the same server as in the previous step where you have created the ACE certificates.

1. Creating **Admin** users

Create a text file called admin-users.txt. These users will have READ, WRITE and EXECUTE access on the Integration Server. The file has the following format:

* + Lines starting with a "#" are ignored
  + Each line should specify the <user> <password>, separated by a single space
  + Each user will have "READ", "WRITE" and "EXECUTE" access on the integration server

**Ex: admin-users.txt**

admin1 password1

admin2 password2

1. Creating **Operator** users

Create a text file called operator-users.txt. It contains a list of users to be created as operator users. These users will have READ and EXECUTE access on the Integration Server. The file has the following format:

* Lines starting with a "#" are ignored
* Each line should specify the <user> <password>, separated by a single space
* Each user will have "READ" and "EXECUTE" access on the integration server

**Ex: operator-users.txt**

operator1 password1

operator2 password2

1. Creating **Editor** users

Create a text file called editor-users.txt. These users will have READ and WRITE access on the Integration Server. The file has the following format:

* Lines starting with a "#" are ignored
* Each line should specify the <user> <password>, separated by a single space
* Each user will have "READ" and "WRITE" access on the integration server

**Ex: editor-users.txt**

editor1 password1

editor2 password2

1. Creating **Audit** users

Create a text file called audit-users.txt. These users will have READ access on the Integration Server. The file has the following format:

* Lines starting with a "#" are ignored
* Each line should specify the <user> <password>, separated by a single space
* Each user will have "READ" access on the integration server

**Ex: audit-users.txt**

audit1 password1

audit2 password2

1. Creating **Viewer** users

Create a text file called viewer-users.txt. These users will have READ access on the Integration Server. The file has the following format:

* Lines starting with a "#" are ignored
* Each line should specify the <user> <password>, separated by a single space
* Each user will have "READ" access on the integration server

**Ex: viewer-users.txt**

viewer1 password1

viewer2 password2

Once these steps are done, you will have below files under /var/tmp/ace-files.

* aceserver.key
* aceserver.pem
* \*-users.txt (additional user configuration files)

#### Add ACE Web Console User Config into OpenShift Secret

Create OpenShift secret in the project where FTM will be deployed. This example assumes the project name is ftm-demo.

oc project ftm-demo

oc delete secret **ftm-ace-ssl-cert-secret**

oc create secret generic **ftm-ace-ssl-cert-secret** --from-file=/var/tmp/ace-files

**Note:** Since the .txt files contain passwords in clear text, please delete them by keeping the user passwords at a safe place to be retrieved later when needed.

### Liberty Certificates

Liberty will use the common certificate created in earlier step in case of both certificate manager as well as custom SSL certificates.

### Additional Certificates

Apart from the application specific certificates, customer might also have additional certificates to be configured in the FTM application truststore for integrations with external applications. This is also required for CLIENT\_CERT authentication where the client certificate needs to be added into the Liberty truststore.

Another use case is integration with IBM Licensing Service for usage reporting. For that, the certificate of Licensing Service needs to be imported into the keystore used by FTM OAC. Please refer [*Certificate – Add IBM License Service certificate*](#_Certificate_–_Add) to FTM keystore for details.

**Note:** The additional certificates should be in the PEM format and with file extension as .crt.

In case customer has any root or intermediate CA certificates, append all the certificates in mqserver-ca.crt and add it to this secret. The mqserver-ca.crt will be imported into MQ and ACE as CA or root signer certificates.

#### Get the Additional Certificates

If you have any additional certificates, please follow these steps to add them into the OCP secret.

1. Login to the server where OpenSSL is installed.
2. Create a directory where additional certificates will be placed.

# mkdir -p /var/tmp/addtnl-certs

# cd /var/tmp/addtnl-certs

1. Copy the certificate files to this directory.

#### Add Files into OpenShift Secret

Create OpenShift secret in the project where FTM will be deployed. This example assumes the project name is ftm-demo. (Delete existing secret if exists)

# oc project ftm-demo

# oc delete secret **ftm-additional-certs-secret**

# oc create secret generic **ftm-additional-certs-secret** --from-file=/var/tmp/addtnl-certs

### 2.5.9 Keep the files at secure location

All the files created in above steps should be kept at some secure location before adding them into the OpenShift secrets. You will need to update some of these files (for example truststore) to add new or additional certificates.

## Enable SSL for Db2 Communication

The production Db2 database is external. For SSL, the customer needs to enable Db2 to run over SSL. For enabling Db2 over SSL some general steps are provided but refer to the IBM Db2 11.5 documentation in IBM Knowledge Center for complete details.

**Note:** These steps are based on the SSL configuration of Db2 in Development environment. If you are using production Db2 server, please check with the DB Admins on enabling the SSL for Db2 instance that will be used for FTM.

### 2.6.1 Configure Db2 for SSL

Please check with the DB Admins on enabling the SSL for Db2 instance that will be used for FTM in production environment. Section [Db2 Certificates (Optional)](#_DB2_Certificates_(Optional)) has the steps that FTM development team uses to create the SSL certificates in downloadable Db2 image.

### 2.6.2 Add Db2 public certificate to OCP secret

Pre-4.0.4 approach: If you have configured the SSL certificates using [Pre-4.0.4 approach](#_Creating_custom_SSL), refer the section [Liberty Certificates](#_Liberty_Certificates) for the steps to add Db2 certificate mydbserver.arm in Liberty truststore.

4.0.4-onwards approach: If you have configured the SSL certificates using [4.0.4-onwards approach](#_Creating_custom_SSL_1), refer the section for adding the [Db2 certificate](#_Add_Db2_Certificate) in OCP secrets.

Restart all the pods so that new certificate will be used.

## Enable SSL for FTM Components

For using SSL with Db2 the Db2 database needs to be configured for SSL. Refer to section [Enable SSL for Db2 Communication](#_Configure_DB2_for)

[FTM Digital Payments/Check enable SSL for MQ](#_FTM_Digital_Payments) This will enable the SSL based communication for all incoming traffic to MQ.

[FTM Digital Payments/Interac enable SSL for ACE](#_FTM_Digital_Payments_1) This will enable the SSL based communication for all incoming traffic to ACE. This is controlled by the Operator input parameter mqace.ace\_enable\_ssl (this parameter only affects ACE). The parameter is set to false by default.

When deploying FTM Digital Payments/Interac/ACE, provide value true to mqace.ace\_enable\_ssl Operator input and the SSL will be enabled for all the traffic to ACE.

## Enable SSL for FTM Immediate Payments Components

For using SSL with Db2 the Db2 database needs to be configured for SSL. Refer to section [Enable SSL for Db2 Communication](#_Configure_DB2_for)

[FTM Digital Payments/Interac/Check enable SSL for MQ](#_FTM_Digital_Payments) This will enable the SSL based communication for all incoming traffic to MQ.

## Enable SSL for FTM Digital Payments/Interac/Check Components

For using SSL with Db2 the Db2 database needs to be configured for SSL. Refer to section [Enable SSL for Db2 Communication](#_Configure_DB2_for)

### FTM Digital Payments/Interac/Check JSE Components SSL enablement

Operator input parameter jse\_services.jse\_enable\_ssl identifies whether FTM JSE components (Gateway Server, Business Rules Server, Business Rules Manager, Transaction Server) require an SSL connection for incoming communications. The default value is true (secure).

FTM will use the same certificates that are already configured with JEE and JSE components for encrypting secure communications with these components.

When JSE components are enabled for SSL, all JEE components that initiate connections to JSE servers require a reference to an SSL configuration in their Liberty server configuration. Each component’s reference is a property value stored in the FTM database’s CORE\_PROPERTIES table. Since Operator input parameter jse\_services.jse\_enable\_ssl defaults to SSL enabled, the sample reference implementations provided in Data Setup Utility (DSU) import workbooks set these properties’ values to reference the Liberty SSL configuration defaultSSLConfig.

If the operator value is set to not enable SSL communications, then the references to Liberty SSL configurations must be removed.

Customers are expected to modify the JEE SSL configuration references in either of the following ways:

* Using the Control Center to update properties pages for the various components. Relevant pages are located under navigation path Administration > Components > {component\_name} > {page\_name}.
* Importing configuration data via the Data Setup Utility.

The table below outlines affected components and their properties. Column “UI Navigation Path” describes the “{component\_name} > {page\_name}” portion of the Control Center’s navigation path. Column “Property Key” is used for DSU worksheet definitions.

| **Affected Deployments** | **Component** | **Property** | **Property Key** | **UI Navigation Path** |
| --- | --- | --- | --- | --- |
| All | Approvals | Business Rules SSL Configuration Name | apv.rules.br.ssl.config | Approvals > Properties |
| Payement Feature Services | RMI SSL Configuration Name | iyb.properties.rmi.sslconfig | Payment Feature Services > Properties |
| Payement Feature Services | (Business Rules Web Services) Business Rules SSL Configuration Name | iyb.rules.br.ssl.config | Payment Feature Services > Properties |
| Business Rules | Business Rules SSL Configuration Name | izw.properties.general.br.ssl.config | Validation Rules > Properties |
| Gateway Engine | Business Rules SSL Configuration Name | izx.properties.general.br.ssl.config | Gateway > General |
| Check | Auto Adjust | Business Rules SSL Configuration Name | autoadjust.properties  .general.br.ssl.config | Adjustments > General |
| Amount Keying | SSL Config Alias | cky.pds.ssl.config | Amount Keying > Properties |
| TCR | Business Rules SSL Configuration Name | izk.properties.general.br.ssl.config | TCR > General |
| Payment Repair | SSL Configuration Alias | izq.pds.ssl.config | Payment Repair > Properties |

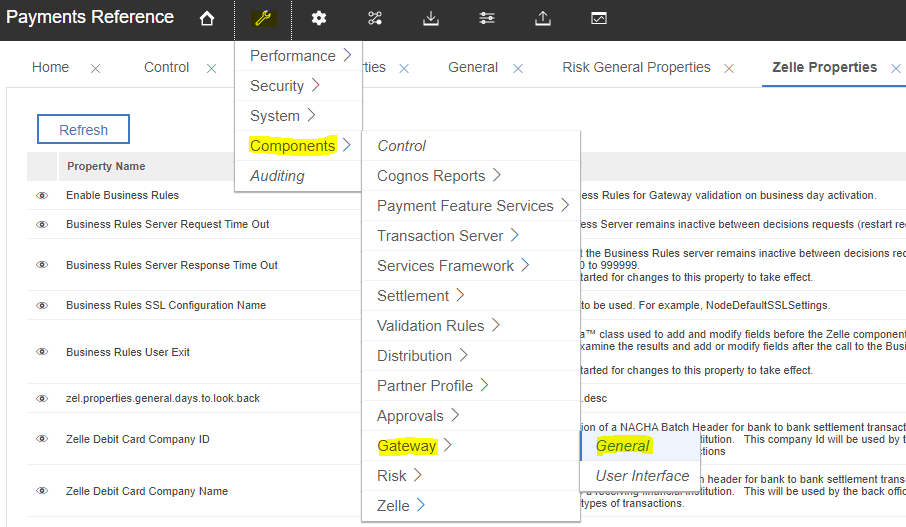
Sections below provide examples of using the Control Center to modify the settings.

**Updating the configuration for Socket SSL communication with Business Rules Server and RMI SSL communication with Gateway Server, Business Rules Manager, Transaction Server:**

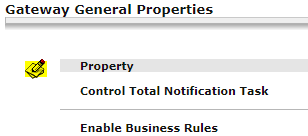
Use the properties pages for each component to configure it to use SSL. Add the SSL configuration alias to use for the Business Rules SSL configuration to the Business Rules SSL configuration name property for the component for each client application like JEE engines, control center etc. Set the value of the property to an empty string if an SSL connection is not used. An example of an SSL configuration alias is **defaultSSLConfig**.

Setting this property for Approvals Engine:

1. Navigate through *Administration > Components > Gateway > General*.



1. Click on highlighted icon to change to Edit mode.



1. Update the below property.



1. Follow the similar steps for other components’ properties.

### FTM Digital Payments/Interac/Check enable SSL for MQ

This will enable the SSL based communication for all incoming traffic to MQ. This is controlled by the Operator input parameter mqace.mq\_enable\_ssl (this parameter only affects MQ). The parameter is set to false by default.

When deploying FTM Digital Payments/Interac/Check, provide following values to the operator parameters.

mq\_enable\_ssl: true

qmgr\_cipherspec: 'TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA256'

qmgr\_ciphersuite: 'SSL\_RSA\_WITH\_AES\_128\_CBC\_SHA256'

qmgr\_channel: 'QMLDAP.SVRCONN'

### FTM Digital Payments/Interac enable SSL for ACE

**Note:** FTM Check does not have any ACE component. So this section is not applicable to Check deployment.

This will enable the SSL based communication for all incoming traffic to ACE. This is controlled by the Operator input parameter mqace.ace\_enable\_ssl (this parameter only affects ACE). The parameter is set to false by default.

When deploying FTM Digital Payments/Interac, provide value true to mqace.ace\_enable\_ssl Operator input and the SSL will be enabled for all the traffic to ACE.

### FTM Digital Payments/Interac/Check enable SSL post deployment

To enable the SSL for Db2, JSE, MQ and ACE communication when FTM Digital Payments/Interac/Check is already deployed, refer the steps below.

**Note:** Please make sure all required certificates are created and those are configured in the OpenShift secret as mentioned in the section [Creating custom SSL certificates (Pre-4.0.4)](#_Creating_custom_SSL). Or [Creating custom SSL certificates (4.0.4-onwards)](#_Creating_custom_SSL_1)

1. Login to OCP UI
2. Click on **Installed Operators**.
3. Click on **IBM FTM-DP Operator** instance and go to **YAML** tab.
4. To enable SSL for Db2, change the database.db\_enable\_ssl attribute to true.

**Note:** This affects all the FTM products.

1. To enable SSL for JSE components (BRS specifically), change the jse\_services.jse\_enable\_ssl attribute to true.

**Note:** This affects all the FTM products that use JSE components.

1. To enable SSL for MQ, change the mqace.mq\_enable\_ssl attribute to true.

**Note:** This affects all the FTM products that uses MQ.

1. To enable SSL for ACE, change the mqace.ace\_enable\_ssl attribute to true.

**Note:** This affects all the FTM products that uses ACE.

1. Update the General properties of JEE components to use the SSL config. Follow the steps from section **Error! Reference source not found.Error! Reference source not found.**.
2. After this change is done, restart all the pods. See [Restart all Pods](#_Restart_all_Pods) section.

## Enable mTLS

FTM supports mTLS (mutual TLS) from 4.0.4 onwards. With mTLS enabled, all the internal communications between FTM components (that support TLS) will use mTLS.

**Note:** The MQ ([FTM Digital Payments/Interac/Check enable SSL for MQ](#_FTM_Digital_Payments)) and BRS ([FTM Digital Payments/Interac/Check JSE Components SSL enablement](#_FTM_Digital_Payments_2)) must be SSL enabled before enabling the mTLS.

mTLS can be enabled during or post deployment.

### Enable mTLS during deployment

To enable the mTLS during FTM deployment, set mtls.mtls\_enable attribute to true in the Operator CR as shown below.

spec:  
  mtls:  
    mtls\_enable: **true**  
    liberty\_mtls\_cert\_label: "liberty"

### Enable mTLS post deployment

To enable the mTLS when FTM is already deployed, refer the steps below.

**Note:** Please make sure all required certificates are created and those are configured in the OpenShift secret as mentioned in the section [Creating custom SSL certificates (Pre-4.0.4)](#_Creating_custom_SSL). Or [Creating custom SSL certificates (4.0.4-onwards)](#_Creating_custom_SSL_1)L

1. Login to OCP UI
2. Click on **Installed Operators**.
3. Click on **IBM FTM-DP Operator** instance and go to **YAML** tab.
4. To enable mTLS change the mtls.mtls\_enable attribute to true.

spec:  
  mtls:  
    mtls\_enable: **true**  
    liberty\_mtls\_cert\_label: "liberty"

**Note:** This affects all the FTM products.

1. After this change is done, restart all the pods. See [Restart all Pods](#_Restart_all_Pods) section.

# Certificate Management

## Certificate management overview

Contents in this section provide information on:

* About self-signed certificates.
* Creating new self-signed certificates.
* Updating the existing keystore and truststore with new certificates.
* Changing keystore and truststore passwords.

## Self-Signed Certificates

FTM components use the keystore (key.p12) and truststore (truststore.jks). These files and mydebserver.arm (which is a public certificate of downloadable Db2 image) are kept in the OpenShift secret **liberty-ssl-cert-secret**. Creating these certificates and secret is documented in section [Creating custom SSL certificates](#_Creating_custom_SSL).

**Keystore** (key.p12)**:** Contains the Liberty key and Liberty certificate, and is configured on all FTM components which are SSL enabled. All JEE components need this to be configured as both keystore and truststore in server.xml.

**Truststore** (truststore.jks)**:** Contains the Liberty Db2 certificate along with Liberty certificate for SSL based communication with Db2 and is configured on all the FTM components that connect to SSL enabled Db2.

Any additional certificates should be imported into this truststore.

### Renewing self-signed certificate

When the SSL certificates configured at deployment time are about to expire or are compromised, these certificates need to be renewed. Follow the section [Creating custom SSL certificates (4.0.4-onwards)](#_Creating_custom_SSL_1) to create new self-signed certificates. This involves creating the new certificates, adding them to keystore/truststore, and creating all required OpenShift secrets.

If you are using IBM Certificate Manager, the certificates in ftm-ssl-cert-secret OCP secret will be updated automatically by Certificate Manager.

Once certificates are updated, you will need to restart all the pods by following the steps at [Restart all Pods](#_Restart_all_Pods).

**Note:** Same approach should be applied in case you are using CA signed certificates.

# Certificate – Add IBM License Service certificate to FTM keystore

IBM License Service runs in a pod in another namespace and is accessed from the FTM Operator and Administrator Console (OAC) container using the service exposed by the License Service.

For the OAC to access the license server over SSL successfully, the certificate of license server needs to be added into keystore used by the FTM components.

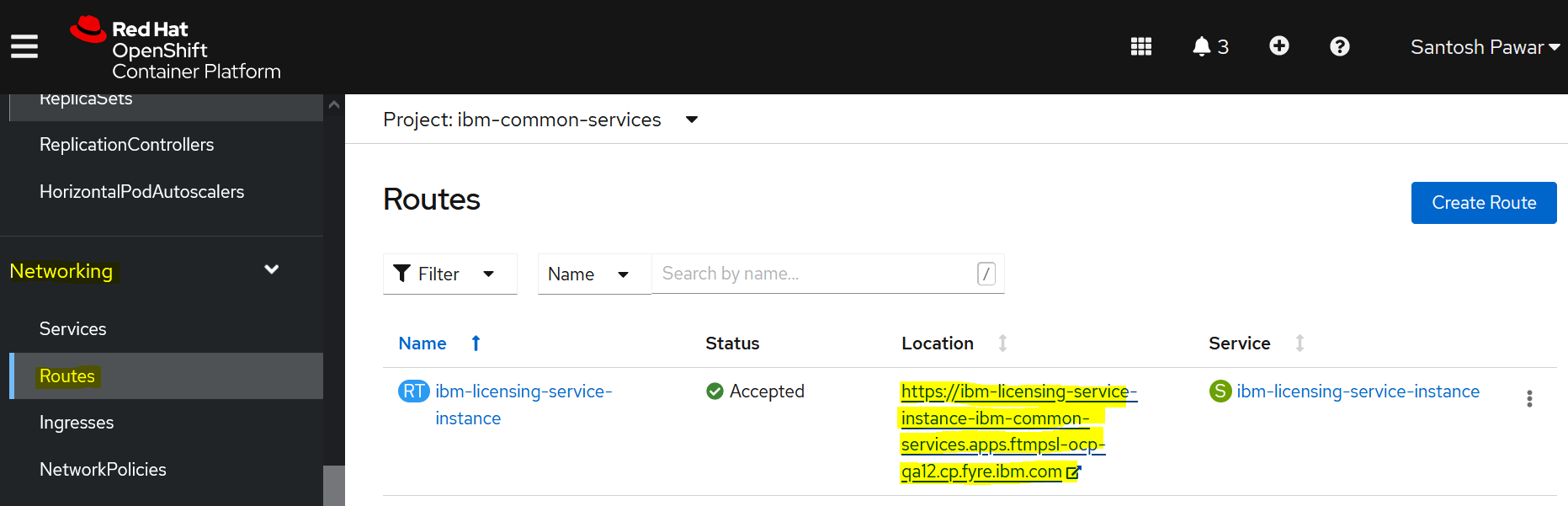
Follow the below steps to add the license server certificate in the FTM keystore.

**Note:** You will need to perform these steps every time the certificate on License server changes. You will need to obtain the license from the license server. You can use any browser supported in your environment or any other method to get the public certificate from license server. The following example is provided by using the Google Chrome browser and its Developer tools Security tab.

**License Service Certificate:**

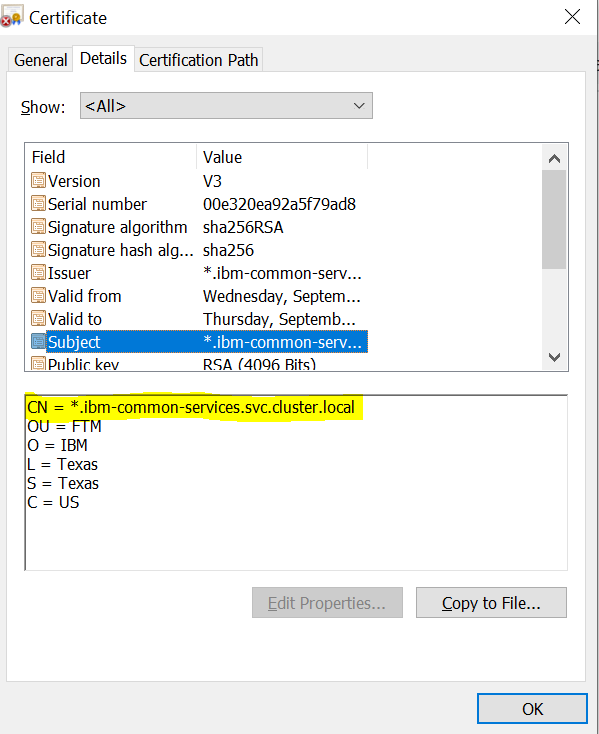
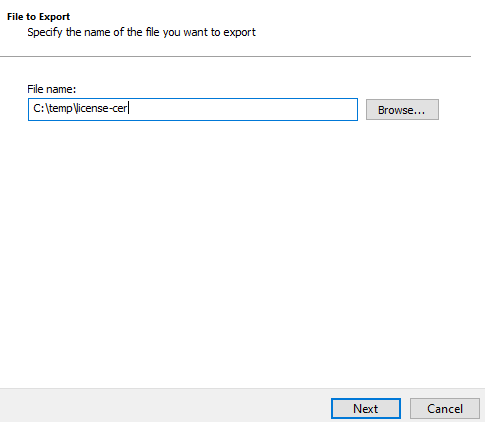
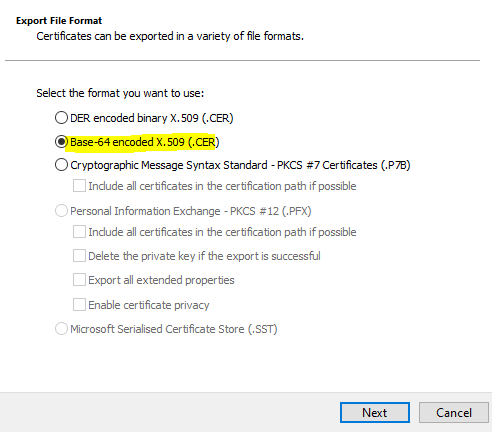
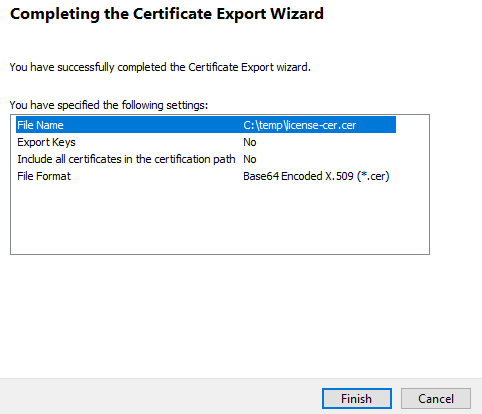
The Licensing Service uses the certificate from OCP secret **ibm-licensing-certs**. This secret is managed by Licensing Service. Please refer the Licensing Service documentation at <https://github.com/IBM/ibm-licensing-operator/blob/release-1.8/docs/Content/Configuration.md#using-custom-certificates>. Make sure the CN field in the certificate resolves the License Service URL that FTM uses to connect to the service. You can either specify the full hostname or the partial domain name matching the hostname in the URL. This URL is fetched by FTM OAC from the ConfigMap named **ibm-licensing-upload-config**. The URL stored in the ConfigMap is normally **https://ibm-licensing-service-instance.ibm-common-services.svc.cluster.local:8080**. This ConfigMap gets created automatically in the namespace where FTM is deployed when Licensing Service is configured for FTM. Please refer the Licensing Service documentation at <https://github.com/IBM/ibm-licensing-operator/blob/release-1.8/docs/Content/Configuration.md#using-custom-certificates> for the details on how to create custom certificate to make sure you provide the correct CN for the certificate.

1. Login to the OpenShift UI and click on the *Networking* -> *Routes*.
2. Select the *ibm-common-services* project from the drop down.
3. Click on the link under *Location* column for *ibm-licensing-service-instance* route.



1. Access the license server URL from browser in Google Chrome browser for example and download the certificate in Base64 format.

**Note: Certificate CN should match the License Service URL:**

1. You will get the public certificate in file C:\temp\license-cer.cer file.
2. Refer the section Additional Certificates for steps to add this certificate into OCP secret.
3. Restart the OAC pods so that new certificate will be used.

# Certificate – Add LDAP certificate to FTM keystore

If you have configured the FTM product to use the LDAP, and if LDAP is configured to be accessed over SSL, then the certificate of LDAP needs to be imported into the Liberty truststore.

Follow the below steps to add the LDAP server certificate in the FTM keystore.

**Note:** You will need to perform these steps every time the certificate on LDAP server changes.

If you have the LDAP server certificate at C:\temp\ldap-cer.cer

1. Get the keystore and truststore files from your secure location that you had created at deployment time. There will be key.p12, truststore.jks and mydbserver.arm files part of that set.
2. Next steps assume that you have copied these files to /tmp/liberty-ssl-certs folder on a Linux machine that has OpenSSL installed and has CLI to connect to OpenShift cluster.
3. Copy the certificate to the same Linux machine at /tmp/ldap-cer.cer.
4. Add LDAP server certificate to keystore. See the section [Import any additional custom certificates](#_Import_any_additional) to import the /tmp/ldap-cer.cer into the Liberty truststore.
5. Delete and recreate the OpenShift secret that keeps these files. See the section [Create OpenShift Secret for Keystores and Db2 certificate](#_Create_OpenShift_Secret) for recreating the Liberty ssl secret.
6. Restart all the pods so that new certificate will be used.

# Network Ports

## Network Ports - Overview

By default, all pods in a project are accessible from other pods and network endpoints. FTM deploys network policy to isolate pods in the project where FTM is deployed. FTM also uses network policy to allow only selective traffic between pods, as required. FTM deploys following network policies

1. Db2 container network policy (only for development node): Allow traffic on ports 50000 and 50001 from pods within the project. Network policy also allow ingress traffic for users external to OpenShift to connect to Db2
2. FTM OAC network policy: Allows traffic from pods only within the same project. Network policy also allow secure ingress traffic to access FTM OAC console
3. MQ/App Connect network policy: Allows traffic from pods only within the same project. Network policy also allow secure ingress traffic to on ports 1414 for MQ and 4414 for ACE console

The following policies are only applicable to FTM Digital Payments/Interac/Check

1. JEE Engines network policy: Allows traffic from pods only within the same project on specific ports only
2. Business Rules Manager and Business Rules Server network policy: Allows traffic from pods only within the same project on specific ports only
3. Gateway Engine network policy: Allows traffic from Control Center and Transaction Server pods only within the same project on specific ports only
4. Transaction Server network policy: Allows traffic from Control Center, Gateway Server, Business Rules Manager and JEE engine pods only within the same project on specific ports only

Network policy allows external traffic to Db2 for required pods only. Additionally, the following ports configured in the network policy to allow Prometheus monitoring of the Db2, if it is configured.

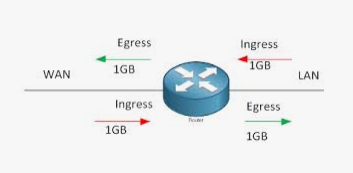
* MQ/App Connect container: Port 9157
* Business Rules Server (BRS) container: Port 9082
* (Business Rules Manager (BRM) container: Port 9081
* Gateway Server (GW) container: Port 9083
* Transaction Server (ITS) container: Port 9084

## Network Policies Used for FTM

By default, all pods in a project are accessible from other pods and network endpoints. To isolate one or more pods in a project, you can create Network Policy objects in that project to indicate the allowed incoming connections.

Ingress – This denotes incoming/inbound traffic coming to pod.

Egress – This denotes traffic originated from pod and going outside.



spec.podSelector.matchLabels – This denotes pods to which Network policy will be applied (e.g. pod A )

spec.ingress – This section specifies Policy rule which denotes from which pods (e.g. pod B, C) inbound traffic is allowed from to policy impacted pods (e.g. pod A ), for example pod A allows incoming traffic from pod B and pod C.

spec.egress – This section denotes, outgoing traffic from Policy impacted pods. Note that spec.egress is not supported below OCP V4.10 (Reference: https://docs.openshift.com/container-platform/4.9/networking/network\_policy/about-network-policy.html)

### Db2 [FTM/Immediate Payments/Digital Payments/Interac/Check]

[Policy is applied to Db2 pod, which accepts incoming traffic from all pods on port 50000 and 50001]

kind: NetworkPolicy

apiVersion: networking.k8s.io/v1

metadata:

name: '{{ db\_network\_policy\_name }}'

labels:

app.kubernetes.io/instance: ftm-base

app.kubernetes.io/managed-by: IBM

app.kubernetes.io/name: ftmbase-operator

namespace: '{{ meta.namespace }}'

spec:

podSelector:

matchLabels:

app: "{{ db\_container\_name }}"

ingress:

- ports:

- protocol: TCP

port: 50000 # Db2 non secure port

- protocol: TCP

port: 50001 # Db2 secure port

egress:

- {}

policyTypes:

- Ingress

- Egress

### MQ and ACE [FTM/Immediate Payments/Digital Payments/Interac/Check]

[Policy is applied to MQ pod, which accepts incoming traffic from all pods **within same Project only** on port 1414, 9443 and 9157]

kind: NetworkPolicy

apiVersion: networking.k8s.io/v1

metadata:

name: '{{ mq\_network\_policy\_name }}'

labels:

app.kubernetes.io/instance: ftm-base

app.kubernetes.io/managed-by: IBM

app.kubernetes.io/name: ftmbase-operator

namespace: '{{ meta.namespace }}'

spec:

podSelector:

matchLabels:

app: '{{ mq\_container\_name }}'

ingress:

- from:

- namespaceSelector:

matchLabels:

name: '{{ meta.namespace }}'

- podSelector: {}

- ports:

- protocol: TCP

port: 1414 # MQ port

- protocol: TCP

port: 9443 # MQ HTTPS Port

- protocol: TCP

port: 9157 # Prometheus / metrics endpoint for MQ

egress:

- {}

policyTypes:

- Egress

- Ingress

[Policy is applied to ACE pod, which accepts incoming traffic from all pods **within same Project only** on port 7600, 7800, 7843 and 9483]

kind: NetworkPolicy

apiVersion: networking.k8s.io/v1

metadata:

name: '{{ ace\_network\_policy\_name }}'

labels:

app.kubernetes.io/instance: ftm-base

app.kubernetes.io/managed-by: IBM

app.kubernetes.io/name: ftmbase-operator

namespace: '{{ meta.namespace }}'

spec:

podSelector:

matchLabels:

app: 'ace-{{ meta.name }}'

ingress:

- from:

- namespaceSelector:

matchLabels:

name: '{{ meta.namespace }}'

- podSelector: {}

- ports:

- protocol: TCP

port: 7600 # ACE HTTPS Port

- protocol: TCP

port: 7800 # ACE HTTP Message flows

- protocol: TCP

port: 7843 # ACE Message flows

- protocol: TCP

port: 9483 # Prometheus / metrics endpoint for ACE

egress:

- {}

policyTypes:

- Egress

- Ingress

### FTM- OAC [FTM/Immediate Payments/Digital Payments/Interac/Check]

[Policy is applied to JEE-OAC pod, which accepts incoming traffic from all pods **within same Project only** on ports 9443 and 9080]

kind: NetworkPolicy

apiVersion: networking.k8s.io/v1

metadata:

name: '{{ oac\_network\_policy\_name }}'

labels:

app.kubernetes.io/instance: ftm-base

app.kubernetes.io/managed-by: IBM

app.kubernetes.io/name: ftmbase-operator

namespace: '{{ meta.namespace }}'

spec:

podSelector:

matchLabels:

app: '{{ oac\_container\_name }}'

ingress:

- from:

- namespaceSelector:

matchLabels:

name: '{{ meta.namespace }}'

- podSelector: {}

- ports:

- protocol: TCP

port: 9443 # HTTPS port

- protocol: TCP

port: 9080 # HTTP port

egress:

- {}

policyTypes:

- Egress

- Ingress

### FTM Digital Payments/Interac/Check – Gateway Server

Policy is applied to JSE-Gateway pod, which accepts incoming traffic from JSE Transaction Server and JEE control Center UI and all other JEE pods on ports 1068, 1099, 2005, and 9083. Refer to the actual network policy in the product for any differences.

kind: NetworkPolicy

apiVersion: networking.k8s.io/v1

metadata:

name: '{{ j2se\_gw\_network\_policy\_name }}'

labels:

app.kubernetes.io/instance: ftm-dp

app.kubernetes.io/managed-by: IBM

app.kubernetes.io/name: ftmdp-operator

namespace: '{{ meta.namespace }}'

spec:

podSelector:

matchLabels:

app: '{{ j2se\_gw\_container\_name }}'

ingress:

- from:

- podSelector:

matchLabels:

app: '{{ j2se\_its\_container\_name }}'

- podSelector:

matchLabels:

app: '{{ j2ee\_control\_center\_ui\_container\_name }}'

- podSelector:

matchLabels:

name: 'ftm-j2ee'

- ports:

- protocol: TCP

port: 1068 # ITS connects to GW server on this port

- protocol: TCP

port: 1099 # RMI communication

- protocol: TCP

port: 2005 # Ingress port for BRS container running in same pod

- protocol: TCP

port: 9083 # Prometheus /metrics endpoint

egress:

- {}

policyTypes:

- Ingress

- Egress

### FTM Digital Payments/Interac/Check – Business Rules Server

Policy is applied to JSE-BRS pod, which accepts incoming traffic from all pods on ports 2005 and 9082 within same Project only. Refer to the actual network policy in the product for any differences.

kind: NetworkPolicy

apiVersion: networking.k8s.io/v1

metadata:

name: '{{ j2se\_brs\_network\_policy\_name }}'

labels:

app.kubernetes.io/instance: ftm-dp

app.kubernetes.io/managed-by: IBM

app.kubernetes.io/name: ftmdp-operator

namespace: '{{ meta.namespace }}'

spec:

podSelector:

matchLabels:

app: '{{ j2se\_brs\_container\_name }}'

ingress:

- from:

- namespaceSelector:

matchLabels:

name: '{{ meta.namespace }}'

- podSelector: {}

- ports:

- protocol: TCP

port: 2005 # Ingress port for BRS

- protocol: TCP

port: 9082 # Prometheus /metrics endpoint

egress:

- {}

policyTypes:

- Ingress

- Egress

### FTM Digital Payments/Interac/Check – Business Rules Manager

Policy is applied to JSE-BRM pod, which accepts incoming traffic from all PODs on ports 1099 and 9081 within same Project only. Refer to the actual network policy in the product for any differences.

kind: NetworkPolicy

apiVersion: networking.k8s.io/v1

metadata:

name: '{{ j2se\_brm\_network\_policy\_name }}'

labels:

app.kubernetes.io/instance: ftm-dp

app.kubernetes.io/managed-by: IBM

app.kubernetes.io/name: ftmdp-operator

namespace: '{{ meta.namespace }}'

spec:

podSelector:

matchLabels:

app: '{{ j2se\_brm\_container\_name }}'

ingress:

- from:

- namespaceSelector:

matchLabels:

name: '{{ meta.namespace }}'

- podSelector: {}

- ports:

- protocol: TCP

port: 1099 # RMI communication

- protocol: TCP

port: 9081 # Prometheus /metrics endpoint

egress:

- {}

policyTypes:

- Ingress

- Egress

### FTM Digital Payments/Interac/Check – Transaction Server

Policy is applied to JSE-ITS pod, which accepts incoming traffic from JSE Gateway, BRM, JEE Control Center UI and all other JEE pods on ports 1099, 2002, and 9084. Refer to the actual network policy in the product for any differences.

kind: NetworkPolicy

apiVersion: networking.k8s.io/v1

metadata:

name: '{{ j2se\_its\_network\_policy\_name }}'

labels:

app.kubernetes.io/instance: ftm-dp

app.kubernetes.io/managed-by: IBM

app.kubernetes.io/name: ftmdp-operator

namespace: '{{ meta.namespace }}'

spec:

podSelector:

matchLabels:

app: '{{ j2se\_its\_container\_name }}'

ingress:

- podSelector:

matchLabels:

app: '{{ j2se\_gw\_container\_name }}'

- podSelector:

matchLabels:

app: '{{ j2se\_brm\_container\_name }}'

- podSelector:

matchLabels:

app: '{{ j2ee\_control\_center\_ui\_container\_name }}'

- podSelector:

matchLabels:

name: 'ftm-j2ee'

- ports:

- protocol: TCP

port: 1099 # RMI communication

- protocol: TCP

port: 2002 # Ingress port for ITS

- protocol: TCP

port: 9084 # Prometheus /metrics endpoint

egress:

- {}

policyTypes:

- Ingress

- Egress

### FTM Digital Payments/Interac/Check – JEE Components

[Policy is applied to all JEE pods, which accepts incoming traffic from all pods on ports 9080, 9443, 9402, 2809 and 2005 within Project only. Refer to the actual network policy in the product for any differences.

kind: NetworkPolicy

apiVersion: networking.k8s.io/v1

metadata:

name: '{{ j2ee\_network\_policy\_name }}'

labels:

app.kubernetes.io/instance: ftm-dp

app.kubernetes.io/managed-by: IBM

app.kubernetes.io/name: ftmdp-operator

namespace: '{{ meta.namespace }}'

spec:

podSelector:

matchLabels:

name: 'ftm-j2ee'

ingress:

- from:

- namespaceSelector:

matchLabels:

name: '{{ meta.namespace }}'

- podSelector: {}

- ports:

- protocol: TCP

port: 9080 # HTTP communication

- protocol: TCP

port: 9443 # HTTPS communication

- protocol: TCP

port: 9402 # Secure IIOP communication port

- protocol: TCP

port: 2809 # IIOP communication port between JEE engines

- protocol: TCP

port: 2005 # Ingress port for BRS. BRS container runs in GW engine pod

egress:

- {}

policyTypes:

- Ingress

- Egress

### FTM Default Deny Policy

This policy restricts all egress traffic from all pods managed-by IBM in namespace in which the network policy is applied.

kind: NetworkPolicy

apiVersion: networking.k8s.io/v1

metadata:

name: ftm-default-deny

namespace: '{{ meta.namespace }}'

spec:

podSelector:

matchLabels:

app.kubernetes.io/managed-by: IBM

policyTypes:

- Egress

egress: []

### FTM Allow Traffic to Internal IP Ranges

This policy allows egress traffic from all pods managed-by IBM in namespace in which the network policy is applied to Internal IP Ranges 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16 to enable DNS.

kind: NetworkPolicy

apiVersion: networking.k8s.io/v1

metadata:

name: ftm-allow-internal-ip-ranges

namespace: '{{ meta.namespace }}'

spec:

podSelector:

matchLabels:

app.kubernetes.io/managed-by: IBM

egress:

- to:

- ipBlock:

cidr: 10.0.0.0/8

- to:

- ipBlock:

cidr: 172.16.0.0/12

- to:

- ipBlock:

cidr: 192.168.0.0/16

### FTM Allow Traffic to Internal IP Ranges

This policy allows egress traffic from all pods managed-by IBM in namespace in which the network policy is applied to white listed CIDR blocks passed from CR.

kind: NetworkPolicy

apiVersion: networking.k8s.io/v1

metadata:

name: ftm-allow-custom-cidrs

namespace: '{{ meta.namespace }}'

spec:

podSelector:

matchLabels:

app.kubernetes.io/managed-by: IBM

egress:

{% for cidr in cidr\_whitelist %}

- to:

- ipBlock:

cidr: '{{ cidr }}'

{% endfor %}

# Liberty transport related security configurations

This section contains recommendations for additional security configurations in Liberty. Some recommendations are already pre-configured. You will need to evaluate if the additional configuration changes need to be made. Refer to the WebSphere Application Server Liberty Knowledge Center

## Secure Cookies

The HTTP session cookie may contain the tokens and sensitive user information. Hence, it is necessary to secure the cookies used in the product by configuring for HttpOnly and Secure fields. Refer to the IBM WebSphere Application Server Liberty documentation in IBM Knowledge Center for further information.

**FTM**

The server.xml comes pre-configured with following stanza.

<httpSession

cookieName="JSESSIONID"

cookieSecure="true"

cookieHttpOnly="true"

cookiePath="/" />

**FTM Digital Payments/Interac/Check**

The server.xml comes pre-configured with following stanza.

<httpSession

cookieName="FXHSESSIONID"

cookieSecure="true"

cookieHttpOnly="true"

cookiePath="/" />

## TLS Communication

TLS and SSL may be used interchangeably.

Enabling HTTPS

Liberty needs to be enabled for HTTPS support by adding the following feature

<featureManager>

<feature>transportSecurity-1.0</feature>

</featureManager>

The server.xml comes pre-configured with this feature.

### Restricting TLS

Restrict the TLS / SSL version support to TLS v1.2

The server.xml comes pre-configured to restrict the TLS / SSL version support to TLS v1.2.

Example server.xml stanza

<ssl id="defaultSSLConfig" trustDefaultCerts="true" clientAuthenticationSupported="true" sslProtocol="TLSv1.2" />

* Custom keystore password is defined in server.xml

<keyStore id="defaultKeyStore" password="yourPassword" />

### Restrict the supported cipher suites

It is recommended to not support weaker encryption cipher suites.

The server.xml for FTM OAC, FTM Digital Payments/Interac/Check Control Center and FTM Digital Payments WebServices comes pre-configured with some weaker cipher suites removed. Example server.xml stanza:

<ssl id="defaultSSLConfig"

trustDefaultCerts="true"

clientAuthenticationSupported="true"

sslProtocol="TLSv1.2"

enabledCiphers={List of cipher suites}

## Configure X-Powered-By header property

To avoid revealing details about the technology used by the server, disable the X-Powered-By header by configuring the following property in the server.xml:

<webContainer disableXPoweredBy="true" />

FTM, FTM for Digital Payments/Interac/Check comes pre-configured with this stanza.

## Configure logoutOnHttpSessionExpire property

When a user’s Control Center browser HTTP session timeout occurs, the user will be logged out. There is a possibility that someone may intercept and override the timeout. To avoid this interception override, configure in the server.xml:

<webAppSecurity logoutOnHttpSessionExpire="true"></webAppSecurity>

Refer to the IBM WebSphere Application Server Liberty documentation in IBM Knowledge Center for further information.

FTM for Digital Payments/Interac/Check comes pre-configured with this stanza.

# Liberty Authentication and Authorization

The FTM Operation and Administrative Console (OAC) and the FTM Digital Payments/Interac/Check Control Center allow users to connect from a browser. Authentication for both FTM and FTM Digital Payments/Check are handled via the user registry configured in Liberty. FTM uses Roles defined in Liberty and FTM Digital Payments user’s permission Groups that are defined via the Control Center.

All FTM UI components (FTM OAC, Control Center UI) that are accessible for end user from the browser, and the WebServices Engine from a customer application.

FTM and FTM Digital Payments/Interac/Check Liberty comes pre-configured with basic user registry for authentication. Liberty allows customers to configure their own custom user registry for authentication.

## Authentication - Basic user Registry

Authentication for both FTM and FTM Digital Payments/Interac/Check are handled via the user registry configured in Liberty. By default, the Liberty components are configured with basic registry. Following is an example stanza that goes in the Liberty configDropins folder when there is no customization by user.

**FTM example server.xml stanza**

<basicRegistry id="basic" realm="BasicRealm">

<user name="fxhadmin" password="${FXH\_PASSWORD}"></user>

<group name="FTMAdmin">

<member name="fxhadmin"></member></group>

<group name="FTMCfg">

<member name="fxhadmin"></member></group>

<group name="FTMEdit">

<member name="fxhadmin"></member></group>

<group name="FTMUser">

<member name="fxhadmin"></member></group>

</basicRegistry>

<administrator-role>

<user>fxhadmin</user>

</administrator-role>

**FTM Digital Payments/Interac/Check example server.xml stanza**

<basicRegistry id="basic" realm="BasicRealm">

<user name="fxhadmin" password="<FXH\_PASSWORD>"></user>

</basicRegistry>

<administrator-role>

<user>fxhadmin</user>

</administrator-role>

## Authentication – Custom user registry

Customers can provide their own user registry configuration before starting the FTM UI components. A custom user-registry XML file can be created and copied to the pre-created Persistent Volume location or to the OpenShift ConfigMaps. Please see the “Application Customization” section in the FTM Knowledge Center for detailed steps.

Normally, customers would have LDAP user registry rather than basic user registry. For more details on how to configure the user registry (basic, LDAP etc) please refer to “Customizing the User Registry for Java Platform, Enterprise Edition” in the IBM WebSphere Application Server Liberty documentation in IBM Knowledge Center.

UI components

In case of custom keystore required by LDAP user registry, those can be added to the same **pv-user-exit** PV to a location referenced in the XML (for example, <PV-root-Path>/server-xml-customization/user-registry/**LdapSSLKeyStore.jks** and <PV-root-Path>/server-xml-customization/user-registry/**LdapSSLTrustStore.jks)**. Since, this PV is mounted at /opt/IBM/FTM/sharedLib, the files will be available at /opt/IBM/FTM/sharedLib/server-xml-customization/user-registry/oac/**LdapSSLKeyStore.jks** and/opt/IBM/FTM/sharedLib/server-xml-customization/user-registry/oac/**LdapSSLTrustStore.jks.** The LDAP user registry configuration XML in this case would be similar to the following:

<server>

<featureManager>

<feature>ldapRegistry-3.0</feature>

<featureManager>

<ldapRegistry id="ldap"

host="ldapserver.mycity.mycompany.com" port="389" ignoreCase="true"

baseDN="o=mycompany,c=us"

bindDN="cn=admin,dc=FTM,dc=dp"

bindPassword="nehapass"

ldapType="Custom"

sslEnabled="true"

sslRef="LDAPSSLSettings">

<idsFilters

userFilter="(&amp;(uid=%v)(objectclass=ePerson))"

groupFilter="(&amp;(cn=%v)(|(objectclass=groupOfNames)

(objectclass=groupOfUniqueNames)

(objectclass=groupOfURLs)))"

userIdMap="\*:uid"

groupIdMap="\*:cn"

groupMemberIdMap="ibm-allGroups:member;ibm-allGroups:uniqueMember;

groupOfNames:member;groupOfUniqueNames:uniqueMember">

</idsFilters>

</ldapRegistry>

<ssl id="LDAPSSLSettings" keyStoreRef="LDAPKeyStore"

trustStoreRef="LDAPTrustStore"/>

<keyStore id="LDAPKeyStore"

location="/opt/IBM/FTM/sharedLib/server-xml-customization/user-registry/**LdapSSLKeyStore.jks**"

type="JKS" password="{xor}CDo9Hgw="/>

<keyStore id="LDAPTrustStore"

location="/opt/IBM/FTM/sharedLib/server-xml-customization/user-registry/**LdapSSLTrustStore.jks**"

type="JKS" password="{xor}CDo9Hgw=" />

</server>

Here, ldapType could be any supported LDAP, such as below-

* Custom
* IBM Lotus Domino
* IBM SecureWay Directory Server
* IBM Tivoli Directory Server
* Microsoft Active Directory
* Netscape Directory Server
* Novell eDirectory
* Sun Java System Directory Server

For SSL communication with an LDAP server to succeed, the Signer certificate for the LDAP server must be added to the truststore that is referenced by the **sslRef** attribute of the <ldapRegistry> element

## Authentication – CLIENT\_CERT

FTM Liberty based components support authentication via CLIENT\_CERT authentication method. A custom web-app-auth-method XML file can be created and copied to the pre-created Persistent Volume location or to the OpenShift ConfigMaps. Please see the “Application Customization” section in the FTM Knowledge Center for detailed steps. Without any customer modifications, the components will use default authentication (BASIC for WebServices and FROM for FTM OAC, Control Center) using basicRegistry.

The CLIENT\_CERT authentication is currently supported for following Liberty components:

* FTM OAC
* FTM Digital Payments/Interac/Check Control Center
* WebServices BusinessRules
* WebServices Gateway
* WebServices PFS
* Interac Outbound API
* Interac Inbound Notifications
* Interac WebServices Payment Emulator
* Interac WebServices Customer Emulator
* Interac WebServices Fraud Emulator

To enable CLIENT\_CERT authentication, below stanza should be included in the web-app-auth-method.xml file that goes in the Liberty configDropins folder:

<server>

<variable name=*"WEB\_APP\_AUTH\_METHOD"* defaultValue=*"CLIENT\_CERT"*/>

</server>

If the customer is using a custom registry like LDAP for user registry, the appropriate certificate filter should be defined. The certificate filter tells the Liberty server which element(s) from the certificate should be used to locate the user in the LDAP. Liberty will form an LDAP search query based on the design of the certificate filter configuration. For more information on how to define a certificate filter for LDAP user registry, refer to “LDAP certificate map mode” section in “Configuring LDAP user registries in Liberty” section in IBM WebSphere Application Server Liberty documentation.

If using default basic registry, the basic-registry.xml file gets added under configDropins folder of each Liberty server. This file defines the Basic registry contents having fxhadmin user defined in it. However, when any custom configuration (like CLIENT\_CERT dropin) is detected, the basic-registry.xml file is not added into the configDropins folder. Customer should define the registry (Basic or LDAP) explicitly whenever server.xml customization is configured. CLIENT\_CERT authentication by default will use the basicRegistry realm to authenticate the user. The certificate map mode by default will use the PRINCIPAL\_CN from the client’s certificate to authenticate the username.

For default basic registry, below stanza should be included in the default-basic-registry.xml (filename should be different from the default basic-resigtry.xml) file that goes in the Liberty configDropins folder:

<server>

<basicRegistry id=*"basic"* realm=*"BasicRealm"*>

<user name=*"fxhadmin"* password=*"${FXH\_PASSWORD}"* />

</basicRegistry>

<administrator-role>

<user>fxhadmin</user>

</administrator-role>

</server>

If CLIENT\_CERT authentication is used for FTM Control Center component, the “Logout URL” System Property must be configured by the customer in the FTM Control Center UI (Administration -> System -> Properties screen). This will enable FTM to properly redirect the user to customer’s landing page upon a user-initiated logout or session expiration. If the “Logout URL” property is not configured, FTM will redirect the logged out/session-expired user back to FTM home page as an authenticated user based on the underlying certificate transmitted by the secure connection.

**Note:** The certificate provided by the client to establish a secure connection between the client and server will be used by liberty server for CLIENT\_CERT authentication. This certificate should be trusted by the liberty server. Please make sure all custom client certificates are added to the liberty trust store as mentioned in [section 2.3.5.4](#_Import_any_additional).

## Authorization – FTM OAC

The authorization of the application to the users is defined using application bindings in Liberty server.xml. Here, the users/groups are mapped to the application roles. The application provides the appropriate access to the users or users in the groups as per this binding.

FTM comes pre-configured with the following application roles and user id assignment. You will need to customize this for your own user ids and roles.

Below is the application binding details for OAC. Role defined in OAC application is FTM.Role.

<application id="OAC" location="FTM.ear" name="FTM" >

<classloader apiTypeVisibility="spec, ibm-api, api, spi, third-party" commonLibraryRef="DB2JCCLib, UserExit">

<privateLibrary apiTypeVisibility="spec, ibm-api, api, spi, third-party">

<fileset dir="/opt/ibm/wlp/dependencies" includes="\*.jar"/>

</privateLibrary>

</classloader>

<application-bnd id="OAC-BND">

<security-role id="OAC-SR" name="FTM.Role.Logon">

<special-subject type="ALL\_AUTHENTICATED\_USERS"/>

</security-role>

</application-bnd>

</application>

## Authorization – FTM Digital Payments/Interac/Check Control Center

For the FTM Digital Payments/Check Control Center UI, the initial user authentication will happen from configured user registry. Authorization is done after a user is authenticated. Authorization determines which pages, and which actions on those pages, a user is allowed to use. Additional users and their authorizations are configured from Control Center UI. Refer to the IBM FTM documentation in IBM Knowledge Center for details.

# FTM Digital Payments/Interac/Check JSE Components – Encrypting configuration files

## Default deployment with JSE files encrypted

The default deployment of FTM automatically encrypts the JSE configuration files for the sensitive parameters.

The following parameters from JSE configuration files are encrypted during this default encryption process.

* **ftm-j2se-brm:** dbConnectPwd dbKeyStorePassword dbTrustStorePassword remoteKeyStorePassword remoteTrustStorePassword remoteKeyStorePassword remoteTrustStorePassword

Cleartext properties file for non-sensitive params - /opt/ibm/ftm/shared/v404/pfs/BusinessRules/manager/profiles/pdmconfig\_\*.properties

Encrypted properties file for sensitive params - /opt/ibm/ftm/shared/v404/pfs/BusinessRules/manager/profiles/pdmconfig.properties.partial.enc

* **ftm-j2se-brs:** dbConnectPwd dbKeyStorePassword dbTrustStorePassword keyStorePassword trustStorePassword remoteKeyStorePassword remoteTrustStorePassword

Cleartext properties file for non-sensitive params - /opt/ibm/ftm/shared/v404/pfs/BusinessRules/server/profiles/config\_\*.properties

Encrypted properties file for sensitive params - /opt/ibm/ftm/shared/v404/pfs/BusinessRules/server/profiles/configBR.properties.partial.enc

* **ftm-j2se-its:** dbConnectPwd dbKeyStorePassword dbTrustStorePassword mqPassword mqKeyStorePassword mqTrustStorePassword remoteKeyStorePassword remoteTrustStorePassword mqKeyStorePassword mqTrustStorePassword

Cleartext properties file for non-sensitive params - /opt/ibm/ftm/shared/v404/pfs/ITS/profiles/config\_\*.properties

Encrypted properties file for sensitive params - /opt/ibm/ftm/shared/v404/pfs/ITS/profiles/configITS.properties.partial.enc

Encrypted XML files for sensitive params –

/opt/ibm/ftm/shared/v404/pfs/ITS/xml/ExtractionProfile.xml.enc

/opt/ibm/ftm/shared/v404/pfs/ITS/xml/ListenerProfileReference.xml.enc

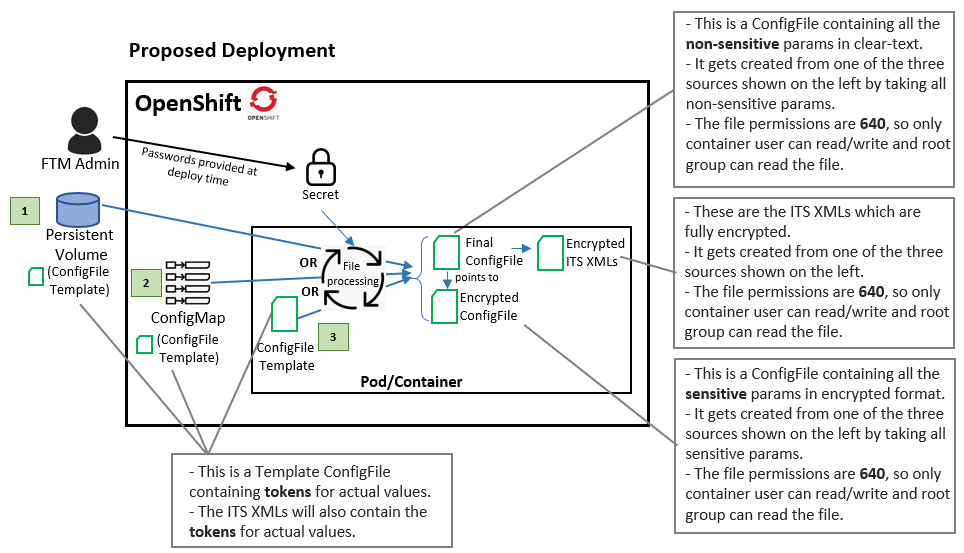
/opt/ibm/ftm/shared/v404/pfs/ITS/xml/SchedulerReference.xml.enc

* **ftm-j2se-gateway:** dbConnectPwd dbKeyStorePassword dbTrustStorePassword mqPassword remoteKeyStorePassword remoteTrustStorePassword

Cleartext properties file for non-sensitive params - /opt/ibm/ftm/shared/v404/pfs/Gateway/properties/izxconfig\_\*.properties

Encrypted properties file for sensitive params - /opt/ibm/ftm/shared/v404/pfs/Gateway/properties/izxconfig-gateway-[inbound|outbound].properties.partial.enc

Below is the design diagram showing how the JSE files are encrypted during the container startup step.



In order to encrypt the JSE configuration files, the minimum requirement is to have the passphrase used for encryption. Hence, it is mandatory to define J2SE\_PROPERTIES\_ENCR\_PARAMS parameter in the application secret. It should, at the minimum, have the passphrase defined in the format "-pass=<passphrase>". For example, "-pass=ibmftm".

## IBM Encrypting configuration files - Overview

To help protect sensitive configuration information, several Payment Feature Services components support the use of properties and configuration files that can be encrypted. Payment Feature Services provides an encryption utility to help the user create these encrypted files.

Most of the Payment Feature Services components support both DES and triple DES encryption. The same encryption algorithm does not need to be used for all the properties and configuration files for a single component. The components use the file extension of the properties file to determine which encryption algorithm was used to encode the file.

Refer to the IBM FTM documentation in IBM Knowledge Center section “[Encrypting configuration files in Payment Feature Services](https://www.ibm.com/support/knowledgecenter/en/SSRH46_4.0.0/gpypfsencryptutil.html)” for more details on file extensions, and components and files supported by the encryption utility.

The following information is specific to using encrypted property files within the containers.

## Encrypting properties with placeholders

The JSE config files use the placeholders that get replaced during container start up.

These placeholders cannot be replaced when they are in an encrypted file. For properties with placeholders that you want to include in the encrypted file use the actual values in the file to be encrypted.

For example, replace

dbConnectPwd = **%DBPASS%**

with

dbConnectPwd = **mydbpass**

There are some properties whose value should be based on the pod name. These values use placeholder %podname%. For example,

serverName = GatewayServer\_**%podname%**

Do not add these properties in the encrypted file even by replacing %podname% with actual value as the %podname% should be replaced with actual pod name that will be different for each pod.

You can encrypt the entire Transaction Server XML files because the Transaction Server XML files do not contain the %podname% placeholder.

**From 4.0.4.1, these placeholders are no longer present in the configuration file. These placeholders %% are replaced with ${} which will be replaced during component startup.**

Update OpenShift application secret if additional encryption parameters used

When an encrypted properties file is being encrypted by using a passphrase (with *-pass*), an encryption password is required to decrypt the file.

-pass=passphrase

If the properties file was encrypted by using custom mode and padding, then they are required for decrypting the file.

-mode=mode

-pad=padding

These values need to be provided to the JSE start up script of that component. In FTM product 4.x, these optional start up script parameters are provided in the OpenShift secret. Once you encrypt the properties file with passphrase and optional mode and padding, update the OpenShift application secret with below string.

-pass=<passphrase> -mode=<mode-value-if-provided> -pad=<padding-value-if-provided>

Refer to the IBM FTM documentation in IBM Knowledge Center section “[Create an application secret for your FTM installation on OpenShift](https://www.ibm.com/support/knowledgecenter/SSRH46_4.0.0/fosdplyprerqinstcsappsecr.html)” on adding and updating the application secret in OpenShift. You will need to add the above value in application secret field name J2SE\_PROPERTIES\_ENCR\_PARAMS.

## Locating and configuring the utility

The encryption utility is provided in Artifacts container for the offering you have deployed (Digital Payments or Interac e-transfer or Check). Please follow the instructions in section “Getting the Files from Artifacts Container” from the product documentation.

Below zip file contains the encryption utility encode.sh

* /home/ftmuser/ftm/artifacts/ftm-db2-<product>-<ftm-release>-latest.zip

Below zip files contain the J2SE configuration files and other supporting files.

* /home/ftmuser/ftm/artifacts/j2se-config.zip (All J2SE config files)
* /home/ftmuser/ftm/artifacts/j2se-brm-jars.zip (if encrypting BRM config file)
* /home/ftmuser/ftm/artifacts/j2se-brs-jars.zip (if encrypting BRS config file)
* /home/ftmuser/ftm/artifacts/j2se-gateway-jars.zip (if encrypting Gateway Server config file)
* /home/ftmuser/ftm/artifacts/j2se-its-jars.zip (if encrypting ITS config file)

## Running the utility

Refer to the IBM FTM documentation in IBM Knowledge Center section “Encrypting configuration files in Payment Feature Services” for more details on file extensions, and components and files supported by the encryption utility.

# Appendix

## Restart all Pods

1. It is easier to restart all the pods using oc command.

**Note:** This step assumes that the namespace has only FTM pods running in it.

1. Connect to OpenShift environment with the user that has required permissions to the project where FTM application is deployed.

$ oc login --server=<OCP-URL>

1. Switch to the project where FTM pods are deployed. For example, psl-ftm-base-qa-team.

# oc get pods

**<output>**

NAME READY STATUS RESTARTS AGE

ftm-db2-psl-ftm-dp-qa-team-85c498b59f-xdbxz 1/1 Running 0 13h

…

$ oc project psl-ftm-base-qa-team

1. Delete each FTM pod in below sequence. Wait for 4-5 minutes after deleting the MQACE pod as it takes time to restart the pod with all the integration servers.
   * db2 (only if you are using FTM provided Db2 container)
   * mqace
   * oac
   * All jse
   * All jee

**Caution:** When Db2 container is restarted, it resets the database back to the original state. You will lose all the data and configurations applied to the database. You will also need to perform [FTM Digital Payments/Interac/Check JSE Components SSL enablement](#_FTM_Digital_Payments_2) if BRS is enabled for SSL.

Replace the pod names with the actual pod names from your command output.

$ oc delete pod <pod-name>

…

1. Wait for few minutes and make sure all pods are back in running state using below command.

$ oc get pods

## Creating custom SSL certificates (Pre-4.0.4)

**Note:** This approach will be deprecated soon hence not recommended to continue with. It is advised to configure the certificates as per [2.4 Creating custom SSL certificates (4.0.4-onwards)](#_Creating_custom_SSL_1) section.

It is recommended to use different set of certificates, even if self-signed, for every deployment. Generally, the self-signed certificates are allowed in development/sandbox deployments. This section provides the steps to create various certificates used by FTM. This is a pre-4.0.4 way of creating certificates where different types FTM components use different certificates. The current approach is still supported in 4.0.4 for backward compatibility.

Below is the list of OpenShift secrets and their contents used by FTM Pods. The below list will help you to understand which secrets and which files inside it you need to update when using your own CA signed certificates.

|  |  |  |
| --- | --- | --- |
| **OpenShift Secret** | **Files in the secret** | **FTM Pods using the secret** |
| ftm-db2-ssl-cert-secret | * mydbserver.arm – SSL certificate for Db2 * myserverdb.crl – Cert revocation list * myserverdb.kdb – Keystore containing Db2 certificate * myserverdb.rdb – Certificate request information * myserverdb.sth – Stash file | Db2 (when FTM shipped Db2 image is used) |
| ftm-mq-ssl-cert-secret | * mqserver.key – SSL Key for MQ * mqserver.crt – SSL certificate for MQ * mqwebuser.xml – MQ web console user configuration * mqserver-ca.crt – CA or root CA certificate | MQ |
| ftm-ace-ssl-cert-secret | * aceserver.key – SSL key for ACE * aceserver.pem – SSL certificate for ACE * \*-users.txt - (additional user configuration files) | ACE |
| liberty-ssl-cert-secret | * mydbserver.arm – SSL certificate for Db2 * key.p12 – Keystore containing key and certificate for Liberty * truststore.jks - Truststore containing certificate for Liberty and Db2 | MQ, ACE, JEE, JSE |

### Db2 Certificates

Please refer section [2.4.2 Db2 Certificates](#_Db2_Certificates)

### MQ Certificates

FTM containerized solution requires the TLS certificates to be created for MQ components.

#### Generate MQ Certificate

Below are the steps to create MQ certificate.

1. Login to the server where OpenSSL is installed.
2. Create a directory where MQ certificates will be placed.

# mkdir -p /var/tmp/mq-certs

# cd /var/tmp/mq-certs

1. Generate the self-signed certificates which will be used by MQ Server and MQ Web Console

# openssl req -newkey rsa:2048 -nodes -keyout **mqserver.key** -x509 -days 365 -out **mqserver.crt** -subj "/C=US/ST=TX/L=Austin/O=IBM/OU=FTM/CN=mqserver"

**Note:** Do not change the filenames as these are referenced in the images.

#### MQ Web Console Users Configuration

1. Create a mqwebuser.xml file in /var/tmp/mq-certs directory based on below template. You can update the users and passwords as per your choice.

**Note**: Do not change **serverKeyAlias="mqservercert"** in **keyStore** Section**.** It is auto configured by the MQ pod.

<?xml version="1.0" encoding="UTF-8"?>

<server>

<featureManager>

<feature>appSecurity-2.0</feature>

<feature>basicAuthenticationMQ-1.0</feature>

</featureManager>

<enterpriseApplication id="com.ibm.mq.console">

<application-bnd>

<security-role name="MQWebAdmin">

<group name="MQWebAdminGroup" realm="defaultRealm"/>

</security-role>

<security-role name="MQWebAdminRO">

<user name="mqreader" realm="defaultRealm"/>

</security-role>

<security-role name="MQWebUser">

<special-subject type="ALL\_AUTHENTICATED\_USERS"/>

</security-role>

<security-role name="MFTWebAdmin">

<user name="mftadmin" realm="defaultRealm"/>

</security-role>

<security-role name="MFTWebAdminRO">

<user name="mftreader" realm="defaultRealm"/>

</security-role>

</application-bnd>

</enterpriseApplication>

<basicRegistry id="basic" realm="defaultRealm">

<user name="**mqadmin**" password="**mqadmin**"/>

<user name="**mqreader**" password="**mqreader**"/>

<user name="**mftadmin**" password="**mftadmin**"/>

<user name="**mftreader**" password="**mftreader**"/>

<group name="MQWebAdminGroup">

<member name="**mqadmin**"/>

</group>

</basicRegistry>

<variable name="httpsPort" value="9443"/>

<variable name="httpHost" value="\*"/>

<keyStore id="defaultKeyStore" location="key.jks" type="JKS" password="password"/>

<ssl id="thisSSLConfig" clientAuthenticationSupported="true" keyStoreRef="defaultKeyStore" **serverKeyAlias="mqservercert"** sslProtocol="TLSv1.2"/>

<sslDefault sslRef="thisSSLConfig"/>

</server>

Once these steps are done, you will have below files under /var/tmp/mq-certs.

* mqserver.key
* mqserver.crt
* mqwebuser.xml

In case customer has any root or intermediate CA certificates, append all the certificates in

* mqserver-ca.crt

mqserver-ca.crt will be imported into MQ and ACE as CA or root signer certificates.

#### Add MQ certificate and Web Console User Config into OpenShift Secret

Create OpenShift secret in the project where FTM will be deployed. This example assumes the project name is ftm-demo.

# oc project ftm-demo

# oc create secret generic ftm-mq-ssl-cert-secret --from-file=/var/tmp/mq-certs

**Note:** Update and confirm following value in the CR file (Operator Instance YAML)

Set mq\_enable\_ssl: true

Set qmgr\_cipherspec: TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA256

Set qmgr\_ciphersuite: 'SSL\_ RSA\_WITH\_AES\_128\_CBC\_SHA256'

Set qmgr\_channel: 'QMLDAP.SVRCONN'

### MQ LDAP Authentication

1. MQ can be configured to authenticate against an external LDAP. It is assumed that you have an LDAP running outside OpenShift.
2. Create a directory where MQ LDAP config file will be placed.

# mkdir -p /var/tmp/mq-ldap

# cd /var/tmp/mq-ldap

1. Delete the existing config map with name ftm-mq-**<INSTANCE\_NAME>**-mqsc-ini-config

# oc project ftm-demo

# oc delete configmap ftm-mq-**<INSTANCE\_NAME>**-mqsc-ini-config

1. Create a file ldap\_config.mqscfile in /var/tmp/mq-ldap directory with LDAP Configuration data. Below file is just a template for configuring an LDAP. Please replace highlighted values according to the LDAP server you are connecting to. You can refer IBM MQ documentation for details of each MQSC command.

ALTER QMGR CHLAUTH(ENABLED)

DEFINE CHANNEL**(QMLDAP.SVRCONN)** CHLTYPE(SVRCONN) TRPTYPE(TCP)

ALTER CHANNEL**(QMLDAP.SVRCONN)** CHLTYPE(SVRCONN) MAXMSGL(104857600)

DEFINE AUTHINFO(FTMQMGR.IDPW.LDAP) AUTHTYPE(IDPWLDAP) CONNAME(‘**myldap.server.com(389)**’) SHORTUSR(**‘uid’**) ADOPTCTX(YES) AUTHORMD(SEARCHGRP) BASEDNG(**‘ou=mqusers,dc=example,dc=com’**) BASEDNU(**‘dc=example,dc=com’**) CHCKCLNT(OPTIONAL) CHCKLOCL(NONE) CLASSGRP(**‘groupOfUniqueNames’**) CLASSUSR(**‘inetOrgPerson’**) FINDGRP(**‘uniqueMember’**) GRPFIELD(**‘cn’**) LDAPPWD(**‘password’**) LDAPUSER(**‘cn=read-only-admin,dc=example,dc=com’**) NESTGRP(YES) SECCOMM(NO) USRFIELD(‘uid’)

ALTER QMGR CONNAUTH(FTMQMGR.IDPW.LDAP)

REFRESH SECURITY

SET AUTHREC PROFILE(‘\*\*’) OBJTYPE(QMGR) GROUP(**‘ou=mqusers,dc=example,dc=com’**) AUTHADD(ALL)

SET AUTHREC PROFILE(‘\*\*’) OBJTYPE(queue) GROUP(**‘ou=mqusers,dc=example,dc=com’**) AUTHADD(ALL)

SET AUTHREC PROFILE(‘@class’) OBJTYPE(queue) GROUP(**‘ou=mqusers,dc=example,dc=com’**) AUTHADD(CRT)

SET AUTHREC PROFILE(‘\*\*’) OBJTYPE(topic) GROUP(**‘ou=mqusers,dc=example,dc=com’**) AUTHADD(ALL)

SET AUTHREC PROFILE(‘@class’) OBJTYPE(topic) GROUP(**‘ou=mqusers,dc=example,dc=com’**) AUTHADD(CRT)

1. Create OpenShift ConfigMap in the project where FTM will be deployed. This example assumes the project name is ftm-demo.

# oc project ftm-demo

# oc create configmap ftm-mq-**<INSTANCE\_NAME>**-mqsc-ini-config --from-file=/var/tmp/mq-ldap

1. Make sure you have QMGR\_CHANNEL value in CR file is consistent with this file. Once the MQ pod is re-created, it will run these MQSC commands and configure LDAP authentication.  
   Ex: **QMLDAP.SVRCONN**

### ACE Certificates

FTM containerized solution requires the TLS certificates to be created for ACE components.

#### Generate ACE Certificate

Below are the steps to create ACE certificate.

1. Login to the server where OpenSSL is installed.
2. Create a directory where ACE certificates will be placed.

# mkdir -p /var/tmp/ace-certs

# cd /var/tmp/ace-certs

1. Generate self-signed certificate

# openssl req -newkey rsa:2048 -nodes -keyout **aceserver.key** -x509 -days 365 -out **aceserver.pem** -subj "/C=US/ST=TX/L=Austin/O=IBM/OU=FTM/CN=aceserver"

**Note:** Do not change the filenames as these are referenced in the images.

#### ACE Web Console User Configuration

1. ACE console can be configured with users and can provide roles to the users. You can create one or more types of users based on your choice.

Use the same server as in the previous step where you have created the ACE certificates.

1. Creating **Admin** users

Create a text file called admin-users.txt. These users will have READ, WRITE and EXECUTE access on the Integration Server. The file has the following format:

* + Lines starting with a "#" are ignored
  + Each line should specify the <user> <password>, separated by a single space
  + Each user will have "READ", "WRITE" and "EXECUTE" access on the integration server

**Ex: admin-users.txt**

admin1 password1

admin2 password2

1. Creating **Operator** users

Create a text file called operator-users.txt. It contains a list of users to be created as operator users. These users will have READ and EXECUTE access on the Integration Server. The file has the following format:

* Lines starting with a "#" are ignored
* Each line should specify the <user> <password>, separated by a single space
* Each user will have "READ" and "EXECUTE" access on the integration server

**Ex: operator-users.txt**

operator1 password1

operator2 password2

1. Creating **Editor** users

Create a text file called editor-users.txt. These users will have READ and WRITE access on the Integration Server. The file has the following format:

* Lines starting with a "#" are ignored
* Each line should specify the <user> <password>, separated by a single space
* Each user will have "READ" and "WRITE" access on the integration server

**Ex: editor-users.txt**

editor1 password1

editor2 password2

1. Creating **Audit** users

Create a text file called audit-users.txt. These users will have READ access on the Integration Server. The file has the following format:

* Lines starting with a "#" are ignored
* Each line should specify the <user> <password>, separated by a single space
* Each user will have "READ" access on the integration server

**Ex: audit-users.txt**

audit1 password1

audit2 password2

1. Creating **Viewer** users

Create a text file called viewer-users.txt. These users will have READ access on the Integration Server. The file has the following format:

* Lines starting with a "#" are ignored
* Each line should specify the <user> <password>, separated by a single space
* Each user will have "READ" access on the integration server

**Ex: viewer-users.txt**

viewer1 password1

viewer2 password2

Once these steps are done, you will have below files under /var/tmp/ace-certs.

* aceserver.key
* aceserver.pem
* \*-users.txt (additional user configuration files)

#### Add ACE certificate and Web Console User Config into OpenShift Secret

Create OpenShift secret in the project where FTM will be deployed. This example assumes the project name is ftm-demo.

# oc project ftm-demo

# oc create secret generic ftm-ace-ssl-cert-secret --from-file=/var/tmp/ace-certs

**Note:** Since the .txt files contain passwords in clear text, please delete them by keeping the user passwords at a safe place to be retrieved later when needed.

### Liberty Certificates

FTM containerized solution requires the certificates to be created for Liberty components.

#### Generate Liberty Certificates

1. Login to the server where OpenSSL is installed.
2. Create a directory where Liberty certificates will be placed.

# mkdir -p /var/tmp/liberty-certs

# cd /var/tmp/liberty-certs

1. Create private key and certificate.

# openssl req -newkey rsa:2048 -nodes -keyout **ftmKey.pem** -x509 -days 365 -out **ftmCertificate.pem** -subj "/C=US/ST=TX/L=Austin/O=IBM/OU=FTM/CN=liberty"

1. Create PKCS12 keystore with private key and certificate.

# openssl pkcs12 -export -in ftmCertificate.pem -inkey ftmKey.pem -out **key.p12** -name "ftmkeyStore" -password pass:<KSTORE\_PASSWORD>

1. Create JKS truststore with certificate.

# keytool -importcert -file ftmCertificate.pem -keystore **truststore.jks** -storepass <TSTORE\_PASSWORD> -alias ftmcert -storetype JKS -noprompt

**Note:** Please keep note of KSTORE\_PASSWORD and TSTORE\_PASSWORD at some secure place. Those are needed when deploying FTM.

1. Copy the Db2 certificate file mydbserver.arm into this directory.
2. Finally, delete the intermediate files ftmKey.pem and ftmCertificate.pem.

Once these steps are done, you will have below files under /var/tmp/liberty-certs.

* key.p12
* truststore.jks
* mydbserver.arm

#### Import Public Certificate of Db2 in Liberty Truststore

1. Import DB certificate mydbserver.arm**.** If you have generated the custom Db2 certificates (as per section Db2 Certificates), then take this file from the generated files. If you are using Db2 certificates that come with FTM, take this file from folder **liberty-ssl-certs** downloaded from Artifacts container.

# keytool -importcert -file mydbserver.arm -keystore truststore.jks -storepass <TSTORE\_PASSWORD> -alias db2cert -storetype JKS -noprompt

#### Import Public Certificate of MQ in Liberty Truststore

1. Import MQ certificate mqserver.crt (generated as per section MQ Certificates)

# keytool -importcert -file mqserver.crt -keystore truststore.jks -storepass <TSTORE\_PASSWORD> -alias mqservercert -storetype JKS -noprompt

1. Import CA certificate mqserver-ca.crt (For CA signed certificates)

# keytool -importcert -file mqserver-ca.crt -keystore truststore.jks -storepass <TSTORE\_PASSWORD> -alias mqservercacert -storetype JKS -noprompt

#### Import any additional custom certificates

In case customer has any root or intermediate CA certificates, those need to be added into the truststore. If there are both root and intermediate certificates, create a single file by concatenating root and intermediate certificate, and import it into the truststore. Any client certificates created to perform [CLIENT\_CERT authentication for FTM Liberty based web services](#_Authentication_–_Web) should also be added into the truststore.

# keytool -importcert -file <custom-cert> -keystore truststore.jks -storepass <TSTORE\_PASSWORD> -alias <my-custom-cert> -storetype JKS -noprompt

#### Add Liberty certificate and Db2 certificate OpenShift Secret

The keystore and truststore created in above steps need to be added into the OpenShift secret to make them available to FTM containers. Liberty secret also needs to have Db2 certificate file.

#### Collect Required Files

1. Login to the server from where you can connect to OpenShift with oc command.
2. Create a directory, say /var/tmp/liberty-ssl-certs to copy all the required files.
3. Copy the following files into that directory.

* mydbserver.arm
* In case of external Db2 which would be normally in case of production deployment, you will need to get this file from the Db2 server where the certificates are created. You can follow the steps from section “Db2 Certificates” to create the Db2 certs.
* In case you have created custom Db2 certificates even in development deployment by following the section “Db2 Certificates”, you need to get the file from there.
* In case you are using the default Db2 certificate that comes with the Db2 image, you need to get the file (/home/ftmuser/ftm/custom-artifacts/base/liberty-ssl-certs/mydbserver.arm) from Artifacts container.
* key.p12
* truststore.jks

#### Add Files into OpenShift Secret

Create OpenShift secret in the project where FTM will be deployed. This example assumes the project name is ftm-demo. (Delete existing secret if exists)

# oc project ftm-demo

# oc delete secret liberty-ssl-cert-secret

# oc create secret generic liberty-ssl-cert-secret --from-file=/var/tmp/liberty-ssl-certs

### Keep the files at secure location

All the files created in above steps should be kept at some secure location before adding them into the OpenShift secrets. You will need to update some of these files (for example truststore) to add new or additional certificates.

# References

* IBM FTM documentation in IBM Knowledge Center [*http://www.ibm.com/support/knowledgecenter/SSRH46/welcomeSSRH46.html*](http://www.ibm.com/support/knowledgecenter/SSRH46/welcomeSSRH46.html)
* IBM WebSphere Application Server Liberty documentation in IBM Knowledge Center [*https://www.ibm.com/support/knowledgecenter/SSEQTP\_liberty/as\_ditamaps/welcome\_liberty.html*](https://www.ibm.com/support/knowledgecenter/SSEQTP_liberty/as_ditamaps/welcome_liberty.html)
* IBM Db2 11.5 documentation in IBM Knowledge Center [*https://www.ibm.com/support/knowledgecenter/en/SSEPGG\_11.5.0/com.ibm.db2.luw.welcome.doc/doc/welcome.html*](https://www.ibm.com/support/knowledgecenter/en/SSEPGG_11.5.0/com.ibm.db2.luw.welcome.doc/doc/welcome.html)
* IBM App Connect Enterprise 11 documentation in IBM Knowledge Center [*https://www.ibm.com/support/knowledgecenter/SSTTDS\_11.0.0/com.ibm.ace.home.doc/help\_home.htm*](https://www.ibm.com/support/knowledgecenter/SSTTDS_11.0.0/com.ibm.ace.home.doc/help_home.htm)

# Notices

This information was developed for products and services offered in the U.S.A.

IBM® may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing

IBM Corporation

North Castle Drive, MD-NC119

Armonk, NY 10504-1785

US

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which was exchanged, should contact:

IBM Director of Licensing

IBM Corporation

North Castle Drive, MD-NC119

Armonk, NY 10504-1785

US

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

## Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at "Copyright and trademark information" at [www.ibm.com/legal/copytrade.shtml](file:///C:\Drive_D\Documents1\E\docs\CPCS\Development\ID\FTM_v4.0.0\www.ibm.com\legal\copytrade.shtml).

Red Hat®, JBoss®, OpenShift®, Fedora®, Hibernate®, Ansible®, CloudForms®, RHCA®, RHCE®, RHCSA®, Ceph®, and Gluster® are trademarks or registered trademarks of Red Hat, Inc. or its subsidiaries in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

## Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

### Applicability

The terms and conditions are in addition to any terms of use for the IBM website.

### Personal use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

### Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside of your enterprise, without the express consent of IBM.

### Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE