Testing the IBM Manufacturing pack for IBM App Connect Enterprise

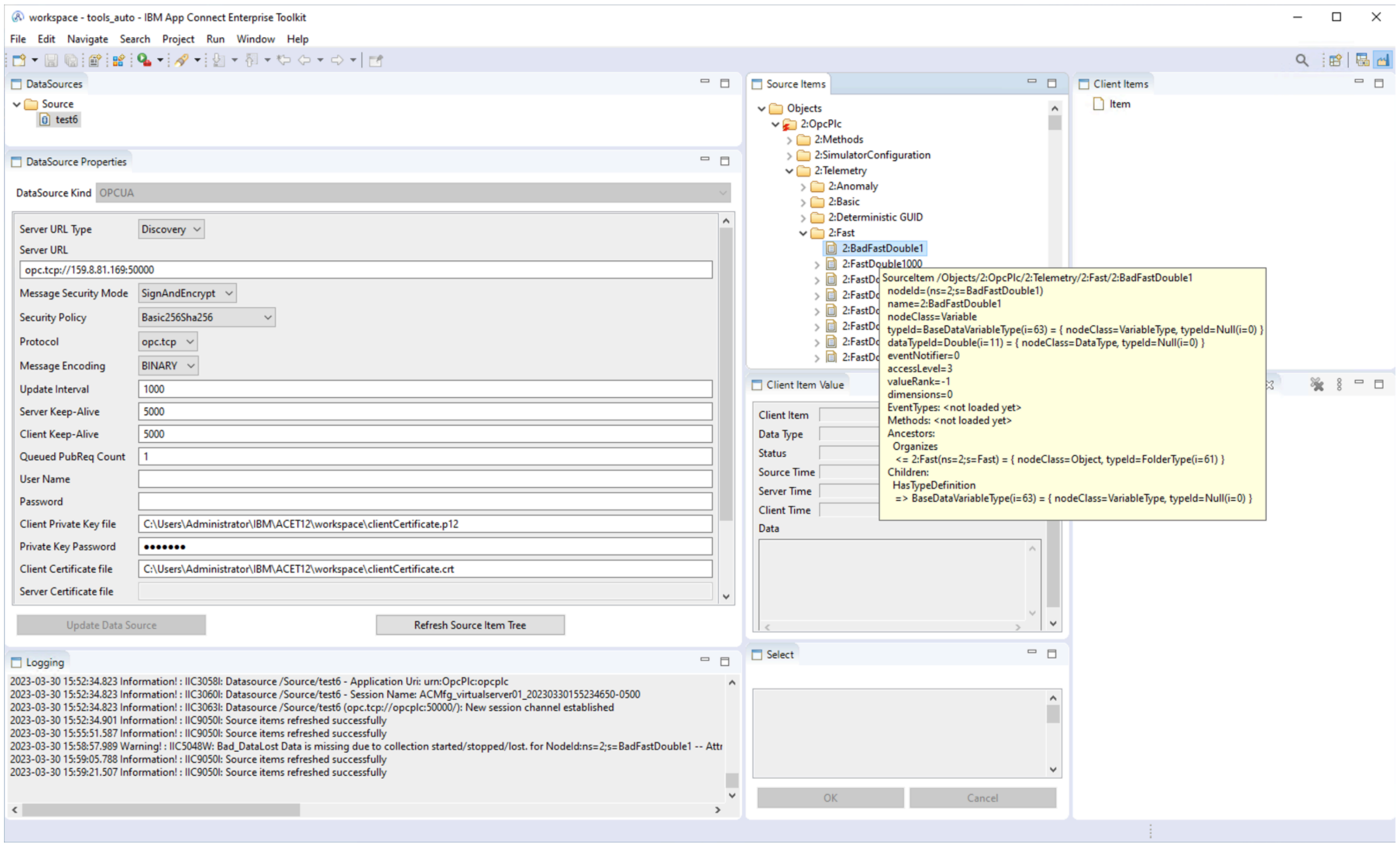
Laurent MARTIN

2023/04/19

Contents

Chapter 1

General



DataSource Kind	OPCUA
Server URL Type	Discovery
Server URL	opc.tcp://159.8.81.169:50000
Message Security Mode	SignAndEncrypt
Security Policy	Basic256Sha256
Protocol	opc.tcp
Message Encoding	BINARY
Update Interval	1000
Server Keep-Alive	5000
Client Keep-Alive	5000
Queued PubReq Count	1
User Name	
Password	
Client Private Key file	C:\Users\Administrator\IBM\ACET12\workspace\clientCertificate.p12
Private Key Password	•••••••
Client Certificate file	C:\Users\Administrator\IBM\ACET12\workspace\clientCertificate.crt
Server Certificate file	

```
SourceItem /Objects/2:OpcPlc/2:Telemetry/2:Fast/2:BadFastDouble1
 nodeId=(ns=2;s=BadFastDouble1)
 name=2:BadFastDouble1
 nodeClass=Variable
 typeId=BaseDataVariableType(i=63) = { nodeClass=VariableType, typeId=Null(i=0) }
 dataTypeId=Double(i=11) = { nodeClass=DataType, typeId=Null(i=0) }
 eventNotifier=0
 accessLevel=3
 valueRank=-1
 dimensions=0
 EventTypes: <not loaded yet>
 Methods: <not loaded yet>
 Ancestors:
  Organizes
   <= 2:Fast(ns=2;s=Fast) = { nodeClass=Object, typeId=FolderType(i=61) }
 Children:
  HasTypeDefinition
   => BaseDataVariableType(i=63) = { nodeClass=VariableType, typeId=Null(i=0) }
```

```
2023-03-30 15:52:34.823 Information! : IIC3058I: Datasource /Source/test6 - Application Uri: urn:OpcPlc:opcplc
2023-03-30 15:52:34.823 Information! : IIC3060I: Datasource /Source/test6 - Session Name: ACMfg_virtualserver01_20230330155234650-0500
2023-03-30 15:52:34.823 Information! : IIC3063I: Datasource /Source/test6 (opc.tcp://opcplc:50000/): New session channel established
2023-03-30 15:52:34.901 Information! : IIC9050I: Source items refreshed successfully
2023-03-30 15:55:51.587 Information! : IIC9050I: Source items refreshed successfully
2023-03-30 15:58:57.989 Warning! : IIC5048W: Bad_DataLost Data is missing due to collection started/stopped/lost. for NodeId:ns=2;s=BadFastDouble1 -- Attr
2023-03-30 15:59:05.788 Information! : IIC9050I: Source items refreshed successfully
2023-03-30 15:59:21.507 Information! : IIC9050I: Source items refreshed successfully
```

Ref: https://github.ibm.com/client-engineering-france/mvp-lyfe-datacoll

IBM ACE is available with an extension supporting the OPC UA interface, as a client.

For testing purpose an OPC UA server (generating samples) is needed. We can use the OPC PLC server.

The communication can be either un-encrypted (for tests only) or encrypted, but in that case X509 certificates must be put in place (both sides).

Linux UNIX

Note: The `Makefile` is intended to be run on a Unix-like system (macOS, Linux)

Windows

In the following sections, `$HOME` refers to `%USERPROFILE%` on Windows.

A IBM Performance Report for ACMfg is available.

Chapter 2

Security and Encryption

2.1 Generation of Client Certificate

The OPCUA protocol requires mutual authentication and supports encryption. Optionally, for tests, clear transmission can be used.

Prior to configuring the ACMfg, one needs to generate a certificate.

In production, Security will be used, this requires certificates on both the client and server.

The ACE documentation provides the steps to generate a self-signed certificate.

A `Makefile` is provided here to generate a simple self-signed certificate in the required format. It simply follows the manual steps described in the documentation.

First initialize the config, execute:

```
make init
```

This creates the folder `private` and file `private/config.env`

Edit this file and fill specific information:

```
CLIENT_ADDRESS=192.168.0.100
SERVER_ADDRESS=192.168.0.101
PASSPHRASE=_your_passphrase_here_
```

Then generate the certificate and key:

```
make
```

Generated files are located in folder `build`:

- `clientCertificate.p12` : Private key and certificate protected by a password in a PKCS12 container.
- `clientCertificate.crt` : The certificate alone in PEM format.

2.2 Comments on ACMfg documentation

A few comments on the ACE documentation:

- The ACE OPC UA client requires: The certificate's Private Key, the Key's passphrase, and the certificate.
- The documentation provides the steps to generate those. (used in the script)
- The documentation talks about ”PEM” format for both.
- The method proposed in documentation shows a PKCS12 container is generated.
- The UI tells, for the key: **Client Private Key in pem file (BASE64)**

In fact, the values to provide are:

- **Client Private Key file** : Expects the PKCS12 container, not PEM BASE64
- **Private Key password** : the password for the PKCS12 container

- **Client Certificate file** : The certificate in PEM format

If the key is not provided in the PKCS12 container, then the following error is logged:

```
ERROR! IIC2037E: Caught exception when trying to load the client certificate and key from C:\...\clientCertif
```

The content of the PKCS12 container (with both the key and certificate) can be displayed with:

```
openssl pkcs12 -info -in build/clientCertificate.p12 -nodes -password pass:_pass_here_
```

Chapter 3

OPC PLC simulator server

In order to simulate the manufacturing side, a simulator can be used. We use here the OPC PLC server.

3.1 Installation of client certificate

The current working directory in the container is : `/app`, as can be seen in the log:

```
[INF] Current directory: /app
...
[INF] Application Certificate store path is: pki/own
...
[INF] Trusted Issuer Certificate store path is: pki/issuer
...
[INF] Trusted Peer Certificate store path is: pki/trusted
...
[INF] Rejected Certificate store path is: pki/rejected
```

So, the default folders used in the container are:

```
/app
   /pki
      /own
      /issuer
      /trusted
      /rejected
```

If no server certificate is provided, the server generates a self signed certificate containing the hostname (of the container, so we fix the hostname value on container startup) in `/app/pki/own`.

On the podman host, in the user's home, we create a folder `pki` and copy the file `clientCertificate.crt` in it.

When the container is started, a volume is created to map this `pki` folder in the user's home to the `/app/pki` folder in the container.

The simulator is given the path to the client certificate (in the container) to add it to the trusted store.

This is automated here (copy startup script and certificate to podman host):

```
make deploy
```

3.2 Startup

The startup script is provider for convenience: `start_opc.sh`

Several parameters are provided to allow unencrypted use, trust of client cert, fix the container hostname.

3.3 Server certificate

Upon startup, the server will generate a self-signed certificate if none is already provided.

The server runs in the container, which has a hostname defaulting to the container id. The CN of the certificate is generated with the hostname, but that hostname changes upon each start of the container (container id), this will make subsequent start fail due to the changing name. A solution is to fix fix the container host name, so that the generated server certificate can be re-used. (in case we need it on the client side).

Chapter 4

ACE Manufacturing

ACMfg provides a manufacturing view with the following tabs:

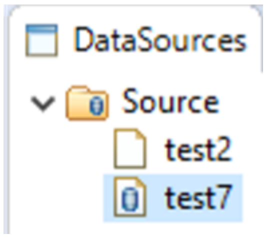
- `DataSources`
- `DataSource Properties`
- `Logging`
- `Source Items`
- `Client Item Value`
- `Select`
- `Client Items`
- `Client Item Properties`

4.1 Creation of data source

In the `DataSources` tab is located the root source, named `Source`. This name can be changed.

Sources can be configured in a hierarchical manner, i.e. sub nodes can be created under the root node or another node.

Then any source node can be configured with a server connection: select the source and enter configuration in the `DataSource Properties` tab.



Sources configured with a server get a blue icon.

4.2 Configuration of DataSource without Encryption

For testing purpose **only**, it is possible to register a source server without encryption and authentication. This is much simpler than using certificates.

> **Note:** The configuration of the startup script `start_opc.sh` allows connection from client without encryption. (option `--unsecuretransport`)

DataSource Properties:

- **Message Security Mode** : None
- **Security Policy** : None
- **Client Private Key file** : leave empty
- **Private Key password** : leave empty

- **Client Certificate file** : leave empty

4.3 Configuration of DataSource with Encryption

Copy the generated files: `build/clientCertificate.crt` and `build/clientCertificate.p12` to the ACE workspace.

ACE Configuration:

- **Message Security Mode** : `SignAndEncrypt`
- **Security Policy** : `Basic256Sha256`
- **Client Private Key file** : `[path to workspace]/clientCertificate.p12`
- **Private Key password** : the password used for the PKCS12 container
- **Client Certificate file** : `[path to workspace]/clientCertificate.key`

The main folder for ACMfg is: `$HOME/.acmfg`

Upon configuration, the following file is generated: `$HOME/.acmfg/mappings/datasources.json`

> **Note:** Take a note of the data source mappingPath, e.g. `/Source/test7`: it will be needed later on to populate the vault with the PKCS12 password. It is the value of field `mappingPath` in `datasources.json`, noted `<source mappingPath>`.

4.4 Issue: ”Create Data Source” button greyed out

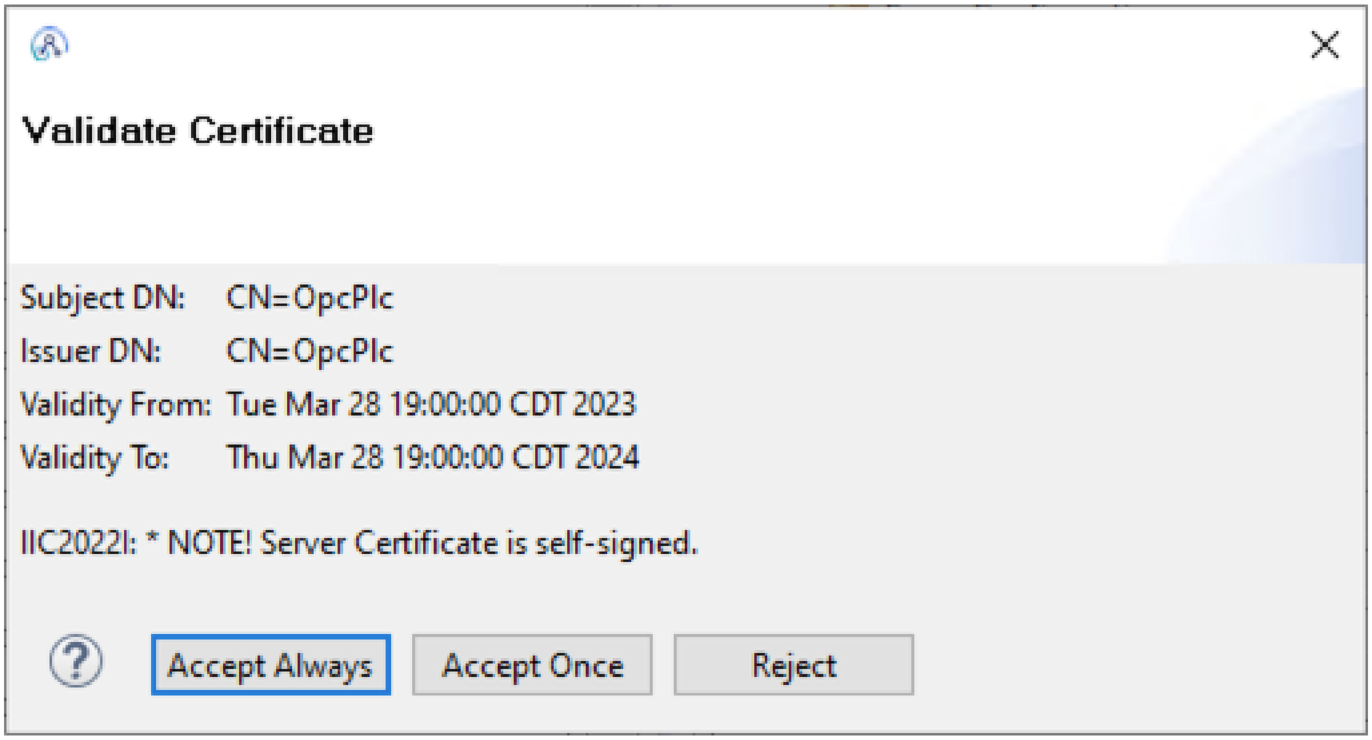
In some cases, the `Create Data Source` button in the ACE manufacturing view stays greyed out.

In this case:

1. Make sure you have created a data source in the folder above the data source properties, and that it is selected. Or un-select, and then select it again.
2. If that persists, close the toolkit, and restart. Eventually, the button shall be black.

4.5 Server certificate on client

The ACE OPC UA client allows (for testing) to accept the server certificate manually:


Validate Certificate

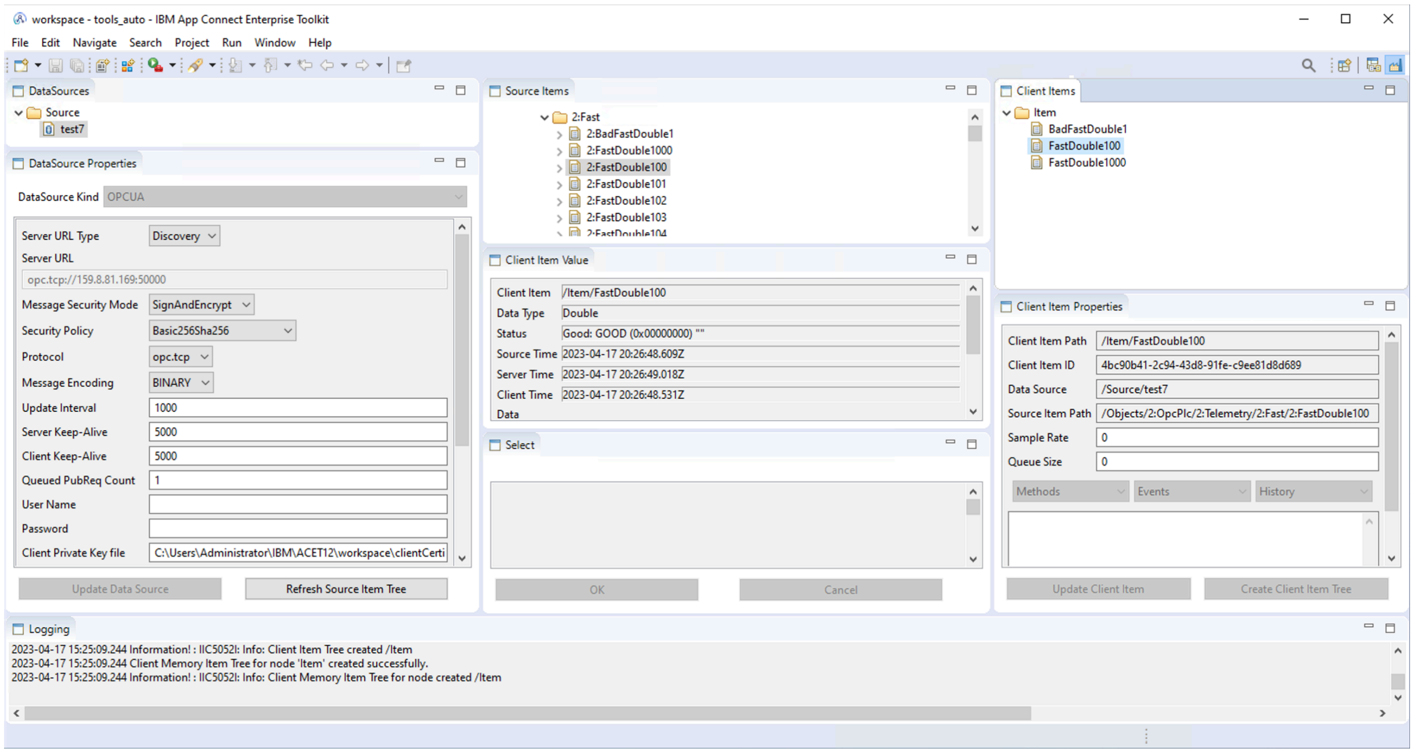
Subject DN: CN=OpcPlc
Issuer DN: CN=OpcPlc
Validity From: Tue Mar 28 19:00:00 CDT 2023
Validity To: Thu Mar 28 19:00:00 CDT 2024

IIC2022I: * NOTE! Server Certificate is self-signed.

4.6 Preparation of mapping nodes

In the manufacturing view, follow these steps:

Window Tab	Action
`DataSources`	Select the data source
`DataSource Properties`	Check that it is properly configured and connected. Click on `Refresh Source Item Tree`
`Source Items`	Check that items were retrieved.
`Client Items`	Select the element `Item`: it is the root item (it can be renamed).
`Source Items`	Navigate to Objects→OpcPlc→Telemetry. Select either a full section, or a list of source items, or a si
`Client Item Properties`	The button `Create Client Item Tree` becomes available (multiple selections), or `Create Client I`
`Client Items`	Note that items are now mapped.



DataSource Kind	OPCUA
Server URL Type	Discovery
Server URL	opc.tcp://159.8.81.169:50000
Message Security Mode	SignAndEncrypt
Security Policy	Basic256Sha256
Protocol	opc.tcp
Message Encoding	BINARY
Update Interval	1000
Server Keep-Alive	5000
Client Keep-Alive	5000
Queued PubReq Count	1
User Name	
Password	
Client Private Key file	C:\Users\Administrator\IBM\ACET12\workspace\clientCerti

Client Item	/Item/FastDouble100
Data Type	Double
Status	Good: GOOD (0x00000000) ""
Source Time	2023-04-17 20:26:48.609Z
Server Time	2023-04-17 20:26:49.018Z
Client Time	2023-04-17 20:26:48.531Z
Data	

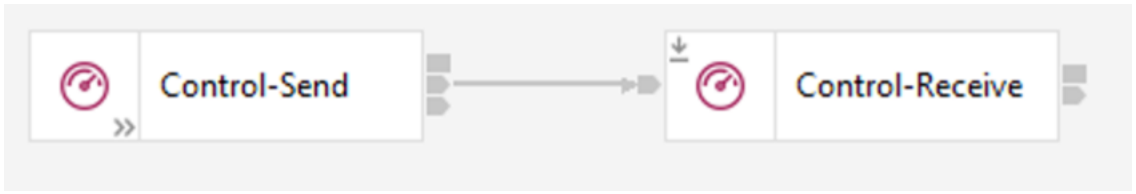
Client Item Path	/Item/FastDouble100
Client Item ID	4bc90b41-2c94-43d8-91fe-c9ee81d8d689
Data Source	/Source/test7
Source Item Path	/Objects/2:OpcPlc/2:Telemetry/2:Fast/2:FastDouble100
Sample Rate	0
Queue Size	0

```
2023-04-17 15:25:09.244 Information! : IIC5052I: Info: Client Item Tree created /Item
2023-04-17 15:25:09.244 Client Memory Item Tree for node 'Item' created successfully.
2023-04-17 15:25:09.244 Information! : IIC5052I: Info: Client Memory Item Tree for node created /Item
```

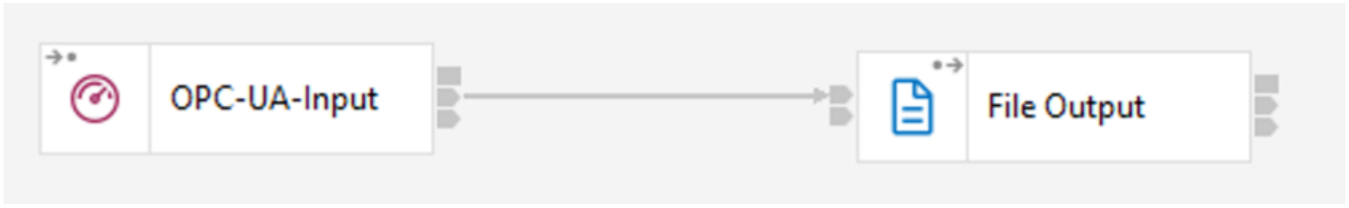
Upon configuration, the following file is generated: `$HOME/.acmfg/mappings/mapping_851eeb72-f996-4af4-8d63-d55c586c2826.`

4.7 Flow creation

As specified in the documentation, create one flow with both control nodes:



And a simple input flow can be:



To select sources for the OPC-UA-Input node follow this:

- in the connector configuration click on `Add`, this switches to the manufacturing view
- in `Client Items` select the desired items
- in `Select`, button `Add Trigger Item` is activated, click on it
- Eclipse switches back to the designer view.

Chapter 5

Integration Server

Create an Integration Server. For example in the toolkit (development): in this case also create the vault with a password.

5.1 ACMfg jars

The integration server (or node) needs to be equipped with the ACMfg jar. This is described in the documentation. Edit `server.conf.yaml`, and configure like this (e.g on Windows):

```
ConnectorProviders:
  ACMfg:
    connectorClassName: 'com.ibm.industrypack.industryclient.connector.ICConnectorFactory'
    jarsURL: 'C:/Program Files/IBM/ACMfg/3.0.1.1/runtime/amd64_nt_4'
    property1: 'trustCertificate=true;isHA=false'
```

Note: Update the jar path accordingly to the actual version installed.

The property: `trustCertificate=true` means that an unknown certificate from a server will be automatically added to the list of accepted certificates.

5.2 Vault and secrets

The integration server also needs the certificate and private key. A vault must be used. If the vault was not created when the IntegrationServer was created using toolkit, or of the IntegrationServer is remote.

See documentation

The vault is created in the IntegrationServer work dir: `<workdirectory>/config/vault/store.yaml`

Windows

Open an ACE Console.

Example of workdir of IntegrationServer: `$HOME/IBM/ACET12/workspace/TEST_SERVER3`

Linux UNIX

On Unix-like systems, open a terminal.

```
mqsivault --work-dir <workdirectory> --create --vault-key <vaultkeyname>
```

Build the `<credentialname>` like this: `<source mappingPath>/acmfgPrivateKeyUser`, e.g. `/Source/test7/acmfgPrivateKeyUser`

Note: The username parameter is not used.

```
mqsicredentials --work-dir <workdirectory> --vault-key <vaultkeyname> --create --credential-type ldap --crede
```

When the IntegrationServer is started it will look for that password based on `<source mappingPath>`.

The data source server information is read from `$HOME/.acmfg`, including the certificate, private key.

Chapter 6

ACE: Starting the integration server in a container

The idea here is to use the pre-built ACE container, and provide ACMfg jars in the mounted volume. (Alternatively a new container image could be built.)

First, create a configuration file :

```
make conf_ace
```

Get your `entitlement_key` from My IBM Product Services

Edit the file: `private/config_container.sh` and place your entitlement key and a secret for the IntegrationServer vault.

Deploy that to the VM where the container will be started:

```
make deploy_ace
```

Then, on the VM where podman will be used, load the tools:

```
for s in ace_container_{config,tools}.sh;do source $s;done
```

Login to IBM image repository

```
podman login cp.icr.io -u cp --password-stdin <<< $entitlement_key
```

6.1 Work directory: Creation

- Ref.: ACE Doc.: mqsicreateworkdir

Since we will mount an empty folder from the host, we must initialize the work directory for the Integration server using `mqsicreateworkdir`:

```
mkdir -p $host_work_directory
chmod 777 $host_work_directory
mqsicreateworkdir $container_work_directory
```

6.2 Vault: Creation

- Ref.: Youtube: Storing encrypted security credentials in a vault
- Ref.: ACE Doc.: mqsivault
- Ref.: ACE Doc.: mqsicredentials

Two commands are used:

- `mqsivault` : create a vault
- `mqsicredentials` : manage credentials in vault

Let's create an empty vault in the `IntegrationServer` work dir:

```
mqsivault --work-dir $container_work_directory --create --vault-key $vault_key
```

> **Note:** Any operation on vault **can** be done while `IntegrationServer` is stopped. When `IntegrationServer` is stopped the vault key **must** be provided on command line (option `--vault-key $vault_key`).

> **Note:** Some operations **must** be done while `IntegrationServer` is stopped: Creation of vault, deletion of entry.

> **Note:** Some operations on vault **can** be done while `IntegrationServer` is running: List entries, Add entry. When `IntegrationServer` is running the vault key is not needed on command line: Requests are made through the `IntegrationServer`.

6.3 ACE: Add Private Key password

Credentials (username/password, and sometimes client id and secret) are sensitive pieces of information. A good practice is to store them in a safe location. ACE provides several ways to store credentials:

provider	description
vault	in an encrypted vault
servercredentials	statically in the server's configuration file
setdbparms	in a parameter storage

Use the same command as previously with local server:

```
mqsicredentials \
--work-dir $container_work_directory \
--vault-key $vault_key \
--create \
--credential-type ldap \
--credential-name $source_path/acmfgPrivateKeyUser \
--username not_used \
--password "$pkcs12_key"
```

6.4 ACE: Start container

- Ref.: ACE Doc.: IntegrationServer command

Several ports are to be published to allow access to the Integration Server:

port	Usage
7600	Integration server web and management port
7700	Integration server debug port
7800	Integration server user API port
7843	Integration server port with TLS

```
podman run \
--detach \
--name aceserver \
--env LICENSE=accept \
--publish 7600:7600 \
--publish 7700:7700 \
--publish 7800:7800 \
--publish 7843:7843 \
--volume $host_work_directory:$container_work_directory \
--entrypoint=bash \
$ace_image \
-l -c \
"IntegrationServer --work-dir $container_work_directory --vault-key $vault_key"
```

Note: The container default entry point is overridden to allow additional arguments to be passed to the integration server (e.g. vault key). (It could also be provided through an environment variable, but not all options are available in env vars.)

Note: The vault key may also be provided through an env var, or through a RC file. (Refer to the `IntegrationServer` manual)