Testing the IBM Manufacturing pack for IBM App Connect Enterprise

Laurent MARTIN

2023/04/17

Contents

Chapter 1

General

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

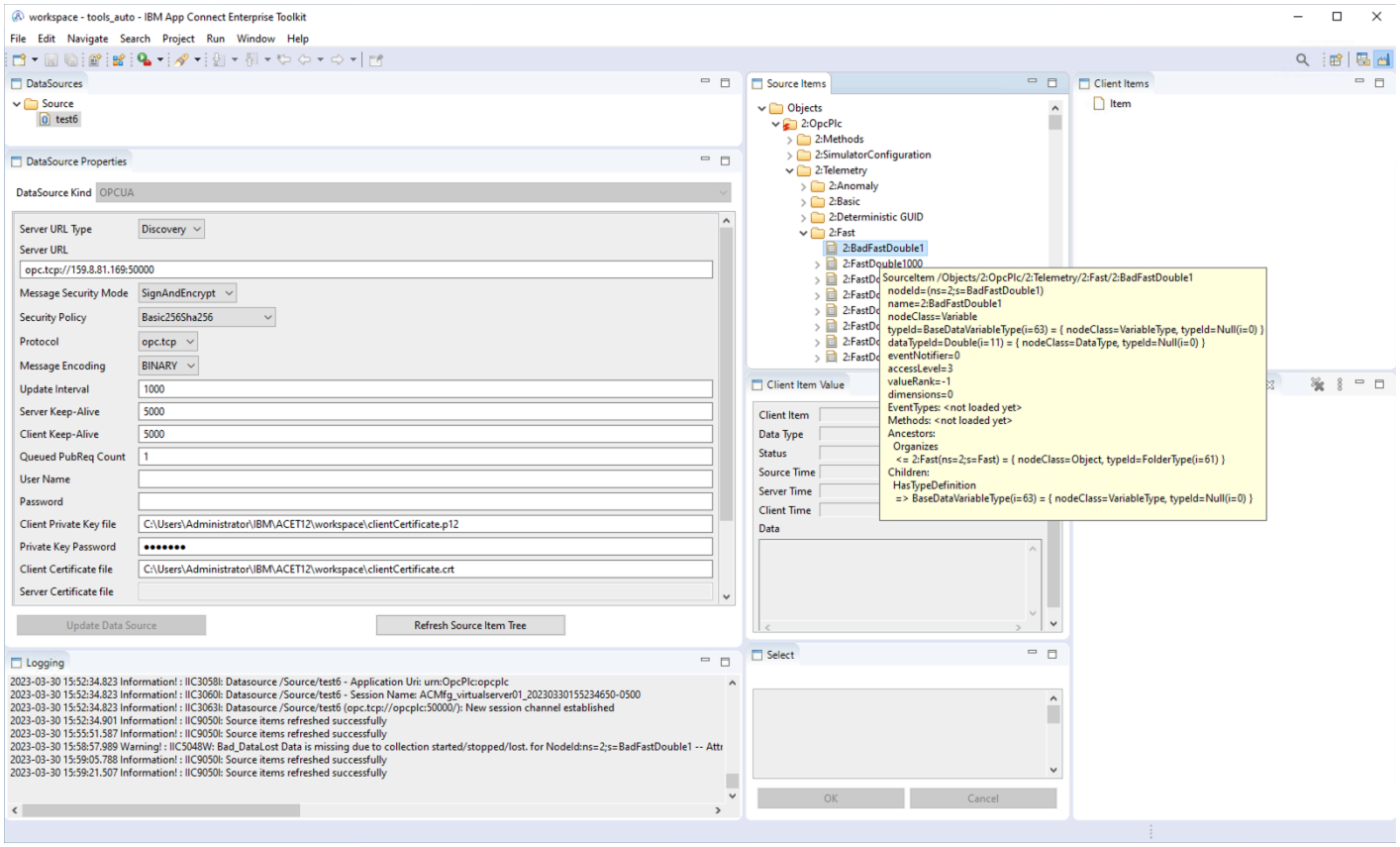
IBM ACE is also available with an extension supporting the OPC UA interface, as a client.

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).

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



Nice IBM Performance Report here.

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`.

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

4.1 Configuration without Encryption

For testing purpose **only**, it is possible to register a 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`)

ACE Configuration:

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

4.2 Configuration with Encryption

Copy 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`

4.3 Issue: ”Create Data Source” button greyed out

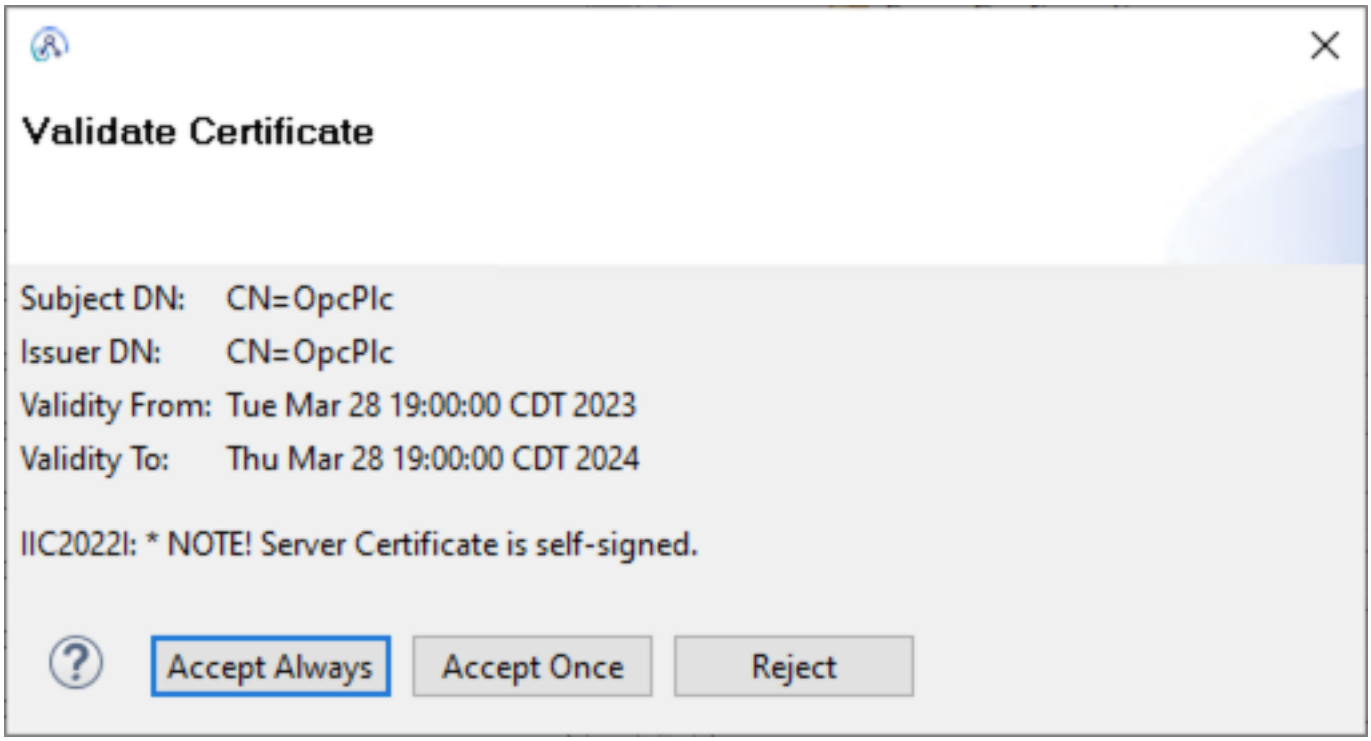
In some cases, the `Create Data Source` button in the ACE manufacturing view stays greyed out.

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.

4.4 Server certificate on client

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



4.5 Preparation of mapping nodes

The manufacturing view provides the following windows:

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

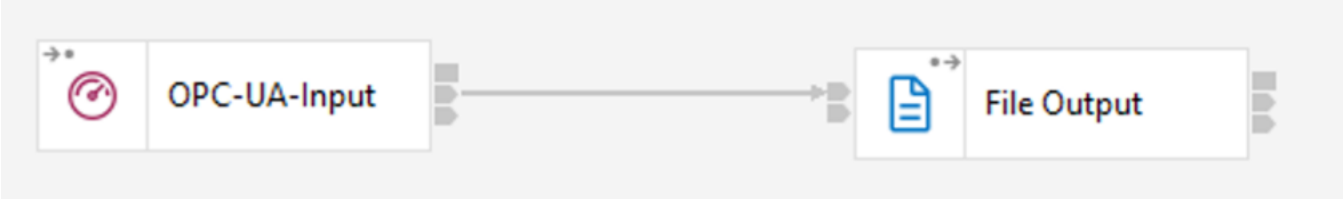
In the manufacturing view:

- Select the data source on top left
- Check that it is properly configured and connected
- Click on `Refresh Source Item Tree`
- Note that the window `Client Items` contains one element `Item`: it is the root item, it can also be renamed. Select it.
- in the `Source Items` tree, navigate to Objects→OpcPlc→Telemetry
- select either a full section, or a list of source items, or a single item.
- upon selection, the button `Create Client Item Tree` becomes available (multiple selections), or `Create Client Item` (single selection)
- Click on `Create Client Item Tree`: this will import the selected items from the list of available items into the selected (root) item.

4.6 Flow creation

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

And a simple collection 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

Chapter 5

Integration Server

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.

The server must also be configured with the client certificate: