

# Testing the IBM Manufacturing pack for IBM App Connect Enterprise

Laurent MARTIN

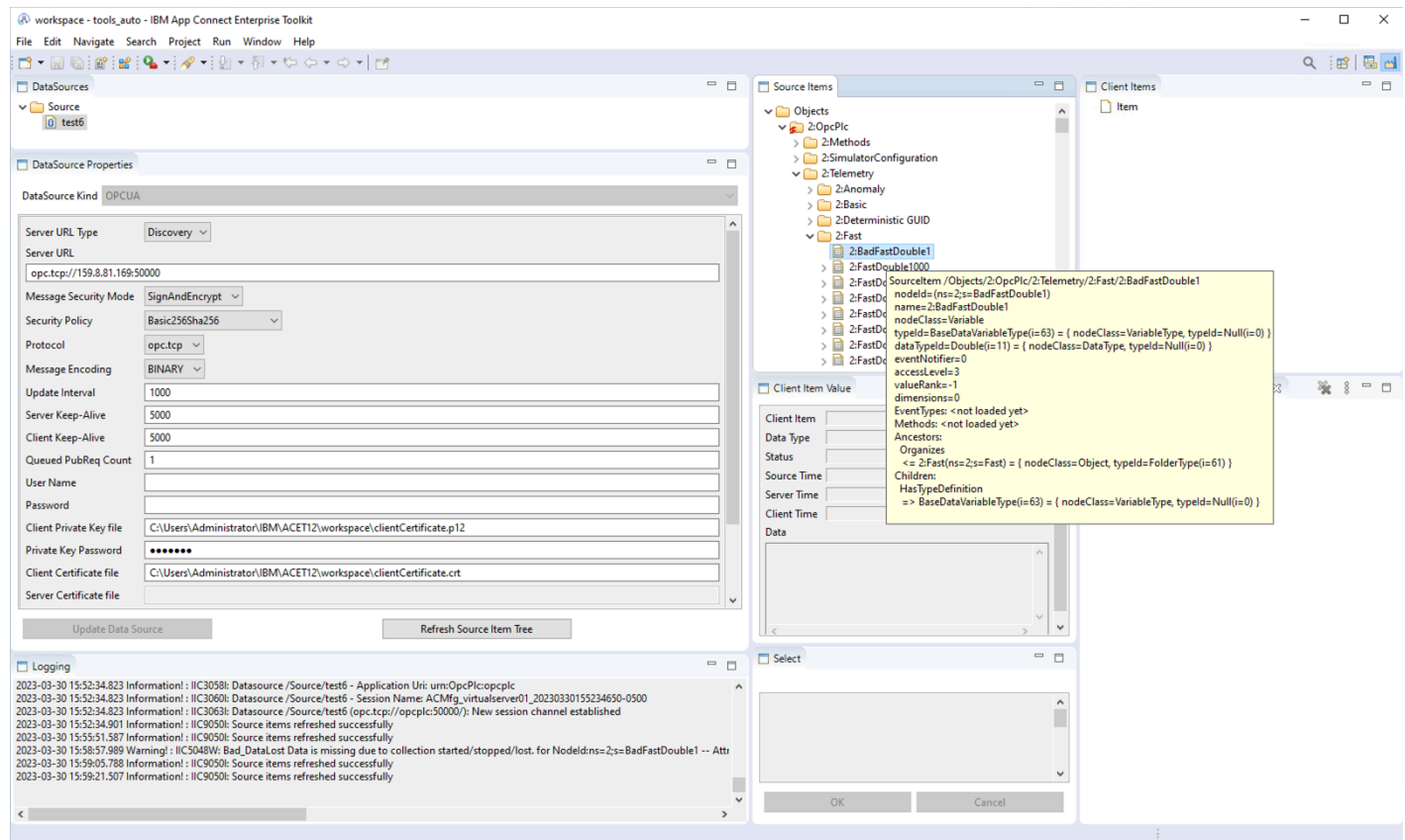
2023/04/18

# Contents

<b>1</b>	<b>General</b>	<b>2</b>
<b>2</b>	<b>Security and Encryption</b>	<b>3</b>
2.1	Generation of Client Certificate . . . . .	3
2.2	Comments on ACMfg documentation . . . . .	3
<b>3</b>	<b>OPC PLC simulator server</b>	<b>5</b>
3.1	Installation of client certificate . . . . .	5
3.2	Startup . . . . .	5
3.3	Server certificate . . . . .	6
<b>4</b>	<b>ACE Manufacturing</b>	<b>7</b>
4.1	Creation of data source . . . . .	7
4.2	Configuration of DataSource without Encryption . . . . .	7
4.3	Configuration of DataSource with Encryption . . . . .	8
4.4	Issue: "Create Data Source" button greyed out . . . . .	8
4.5	Server certificate on client . . . . .	8
4.6	Preparation of mapping nodes . . . . .	9
4.7	Flow creation . . . . .	9
<b>5</b>	<b>Integration Server</b>	<b>11</b>
5.1	ACMfg jars . . . . .	11
5.2	Vault and secrets . . . . .	11
5.3	Starting the integration server in a container . . . . .	12

# Chapter 1

## General



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 provided 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:

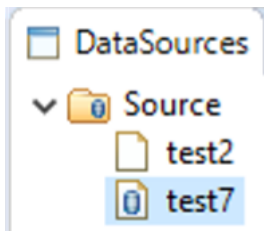
- DataSources
- DataSource Properties
- Logging
- Source Items
- Client Item Value
- Select
- Client Items
- Client Item Properties

### 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

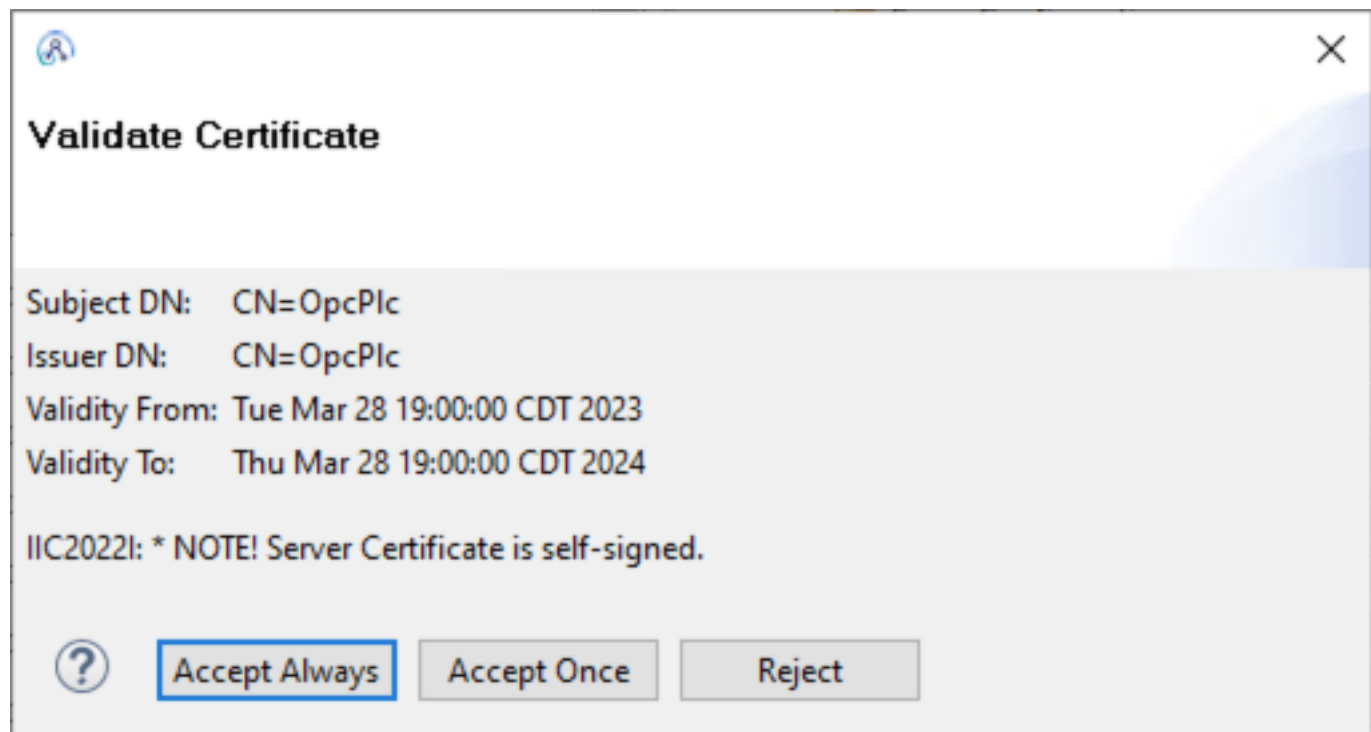
In some cases, the Create Data Source button in the ACE manufacturing view stays greyed out.

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

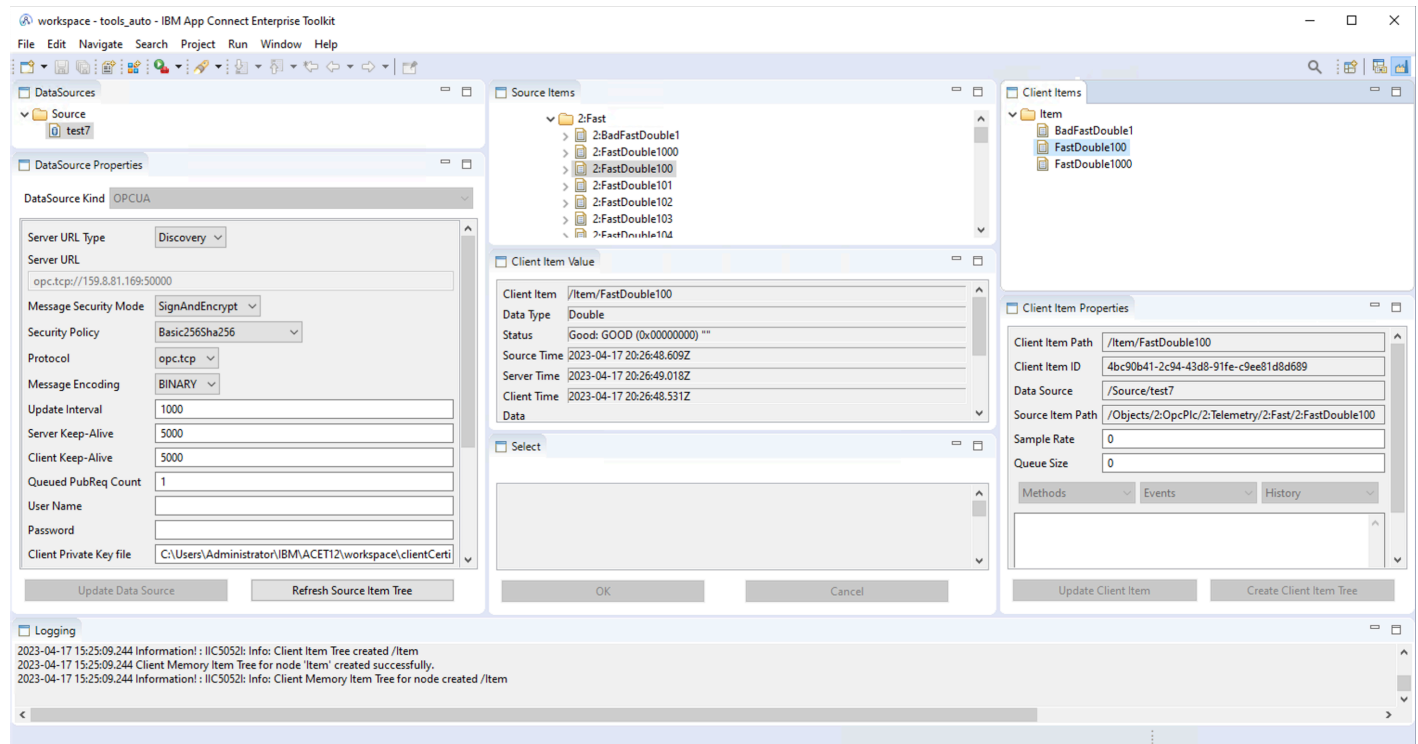
The ACE OPC UA client allows (for testing) to accept the server certificate manually:



## 4.6 Preparation of mapping nodes

In the manufacturing view, follow these steps:

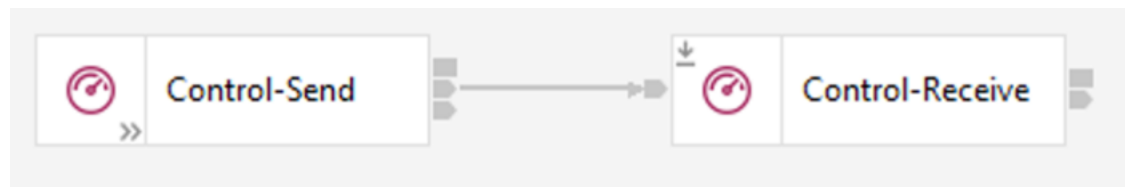
Window Tab	Action
DataSources	Select the data source
DataSource Properties	Check that it is properly configured and connected. Click on Refresh Source Item Tree
Source Items	Check that items were retrieved.
Client Items	Select the element Item: it is the root item (it can be renamed).
Source Items	Navigate 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.



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.industry.pack.industryclient.connector.ICConnectorFactory'
  jarsURL: 'C:/Program Files/IBM/ACMfg/3.0.1.1/runtime/amd64_nt_4'
  property1: 'trustCertificate=true;isSHA=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 --credential
```

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.

## 5.3 Starting the integration server in a container

### TODO

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.