**About This Document**

This document is a mixture of a software design specification for the software module “OPC UA” – which performs many aspects of data transfer and data manipulation when pulling data from a remote OPC server.

**Setup**

Depending whether this should be part of QiaService like a plugin or standalone application differs app setup

**Plugin**

App is executed from outside and goes to overridden Init method

**Standalone.Exe**

Run app using .exe file and configure app behavior in qia.opc folder

Also, you can run from IDLE and just set QIA.Plugin.Opc as startup

**Standalone.Docker**

To run app directly set the Docker as launch profile

To run with orchestration set docker-compose as startup. Following this way DB will be automaticaly created.

**Configuration**

In appsettings.json - main app configuration

• OpcUrl

• DB connection string

• AzureEventHub credentials

- advanced configuration:

• SaveToDb - save static values to db

• SaveToAzure - send data to eventhub

• RecreateDb - if we want to drop db and recreate

nodemanager.json - configuration to lookup for particular nodes on the server

• NodeId

• Name - custom node name

• NodeType - if equals to Subscription will be monitoring value on the server by accepting events

• Range - how many nodes to skip and then save to the DB

• Msecs - how often save to db

AppConfig.json - defines minor configuration when searching for nodes

• Name - (Initial/Advanced/Slim) app pulls the last one from the settings

• SkipPredefined - skips data like ServerStatus/Aliases

• CreateFullTree - if we want full tree to be created

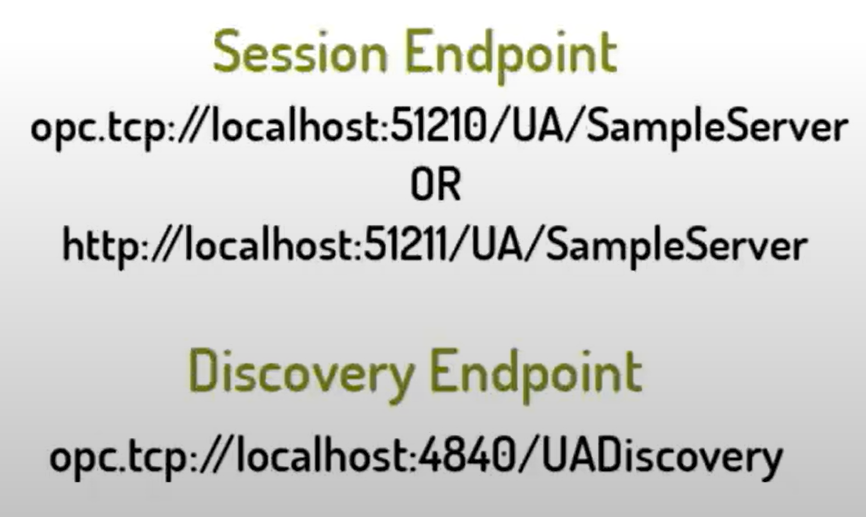
**Logging**

Currently Serilog+Seq is used. Logs are saved locally and also sends to the http://localhost:5341

// will be modified in future

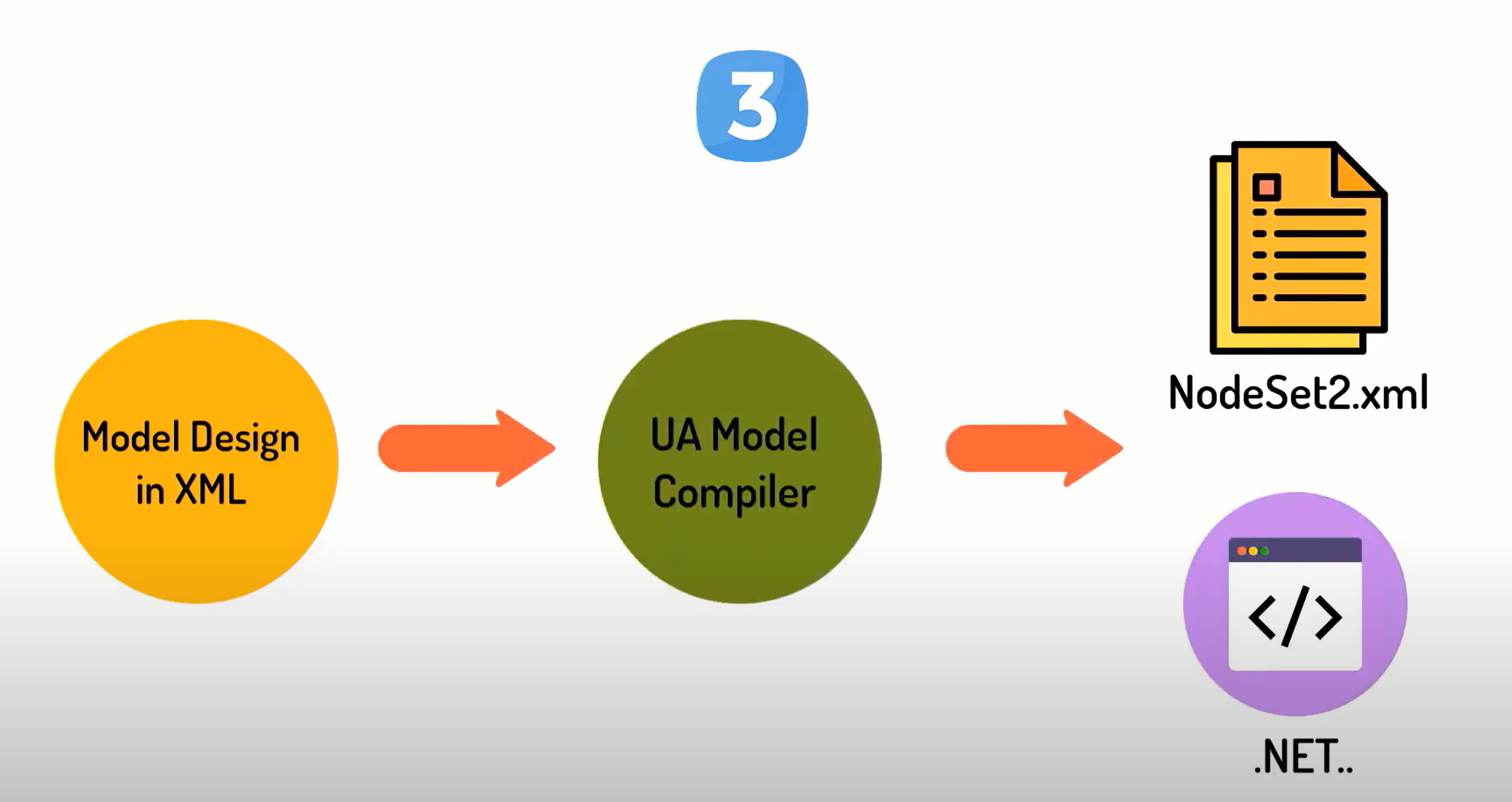
# OPC ua simulator

## Intro



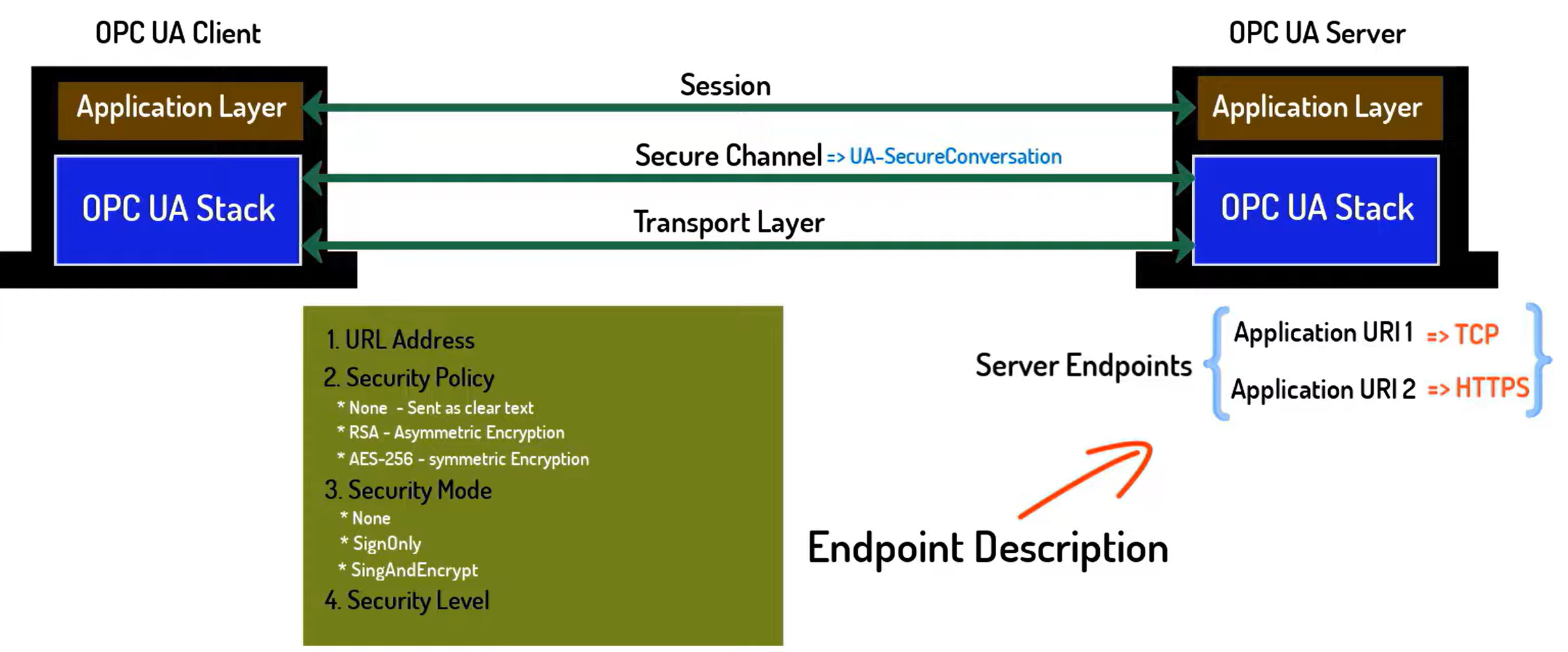
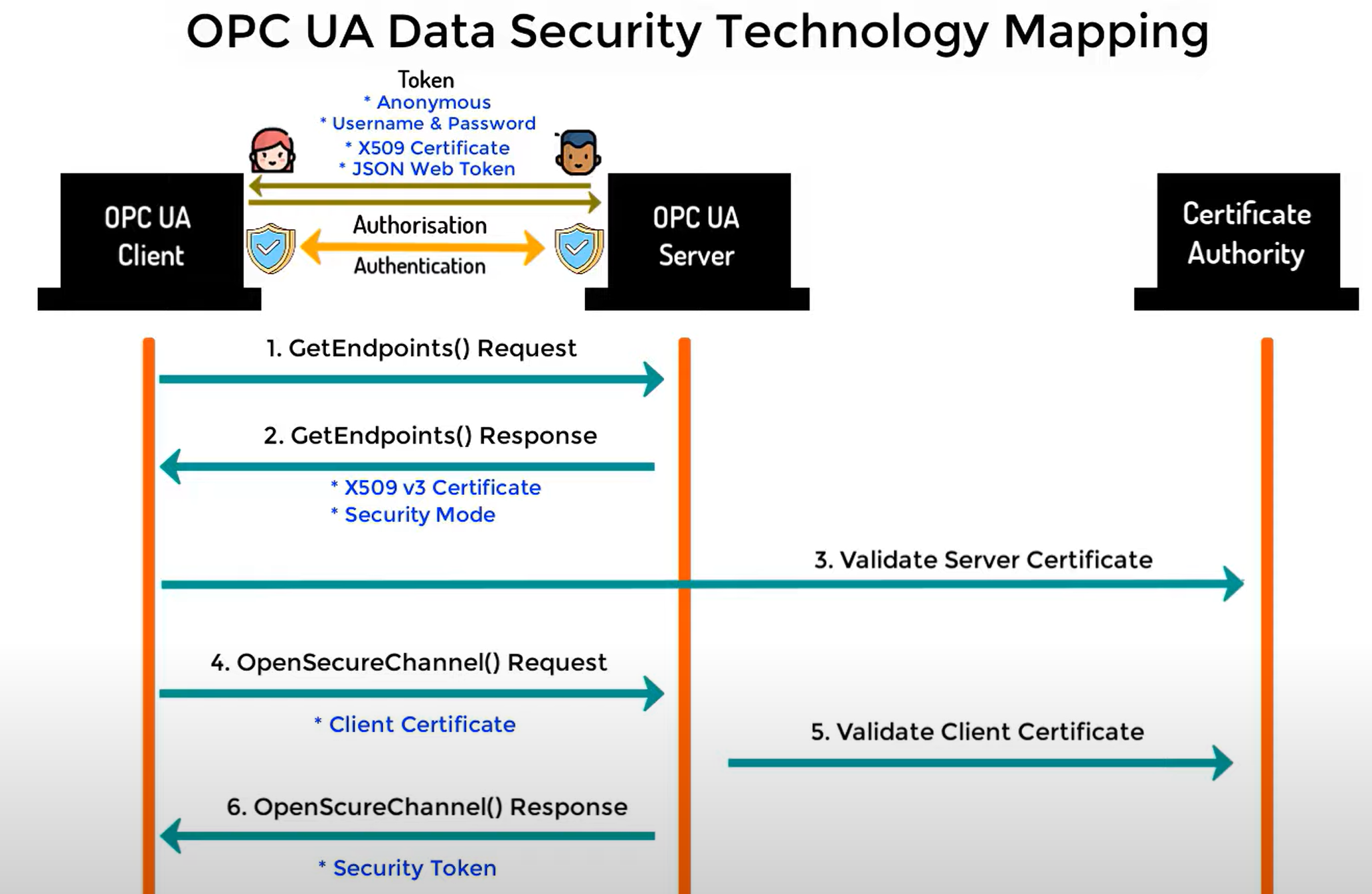
[Source](https://www.youtube.com/watch?v=vgE9P6KNC7g)





[Source](https://www.youtube.com/watch?v=gxA7SDNLHgc&list=PLIrJJXPVFRvjHsA9tta8yULOB8nPUO_G7&index=5)

<https://www.youtube.com/watch?v=yCd2j2WsgBM>

<https://opcua.rocks/step-3-setup-visual-studio-to-edit-opc-ua-modeldesign-files/>

<https://opcfoundation.github.io/UA-.NETStandard/help/index.htm#ua_sample_client.htm>

<https://cache.industry.siemens.com/dl/files/088/42014088/att_954894/v1/42014088_OPC_UAClient_DOKU_V12_en.pdf>

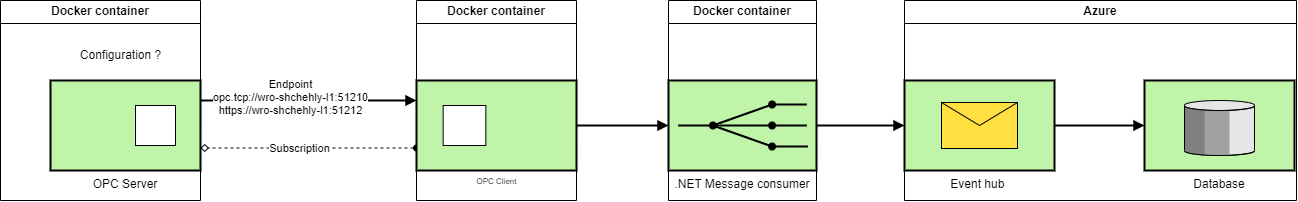
Node with attribute NodeId and value ns=3 ;s=Counter is unique and automatically generated identifier

ns is a namespace index

## Agenda

1. Run OPC UA on Docker
2. ...and on WIN10 Client as well
3. OPC UA Server as well
4. Communicate server and client

x. change tcp to https





### Sample solutions

* <https://hub.docker.com/_/microsoft-iotedge-opc-client>
* <https://hub.docker.com/_/microsoft-iot-opc-ua-test-server>
* <https://hub.docker.com/_/microsoft-iotedge-opc-plc>
* <https://github.com/Azure-Samples/iot-edge-opc-client>
* <https://github.com/Azure-Samples/iot-edge-opc-plc>

Reference

* <https://github.com/OPCFoundation/UA-.NETStandard-Samples>

Red are working console solutions

### Write together your outcome to a "whitepaper"

🡺 way you got there

🡺 Pitfalls

🡺 Documentation (brief) for developer

🡺 whitepaper: aspect Docker setup

## Solution



### Intro, new solution

<https://integrationobjects.com/sioth-opc/sioth-opc-unified-architecture/opc-ua-server-simulator/>

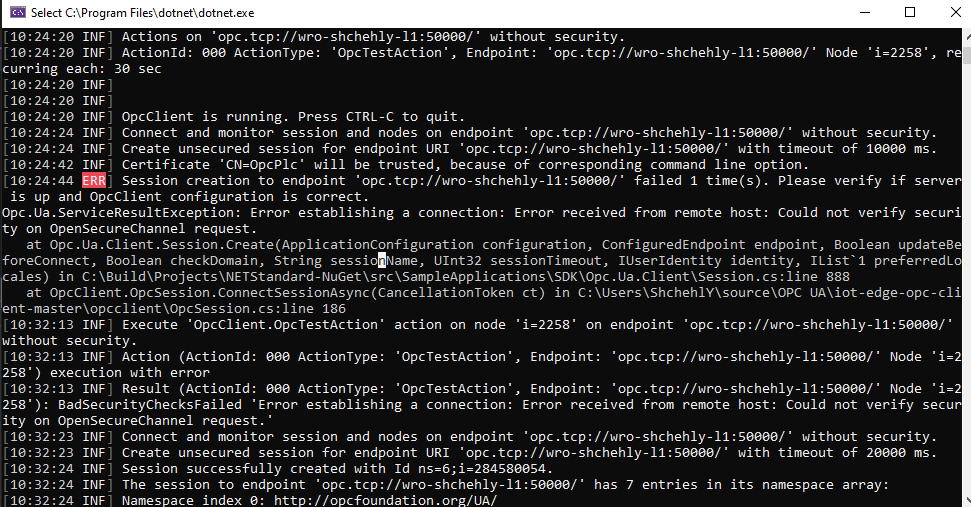
<https://opcfoundation.github.io/UA-.NETStandard/help/index.htm#client_development.htm>

* [Install Docker Desktop on Windows | Docker Documentation](https://docs.docker.com/desktop/install/windows-install/)
  + [Limiting Memory Usage in WSL2 | Aleksandr Hovhannisyan](https://www.aleksandrhovhannisyan.com/blog/limiting-memory-usage-in-wsl-2/)

### Certificate

[Certificate Store Directory Layout](https://reference.opcfoundation.org/v104/GDS/docs/F.1/)

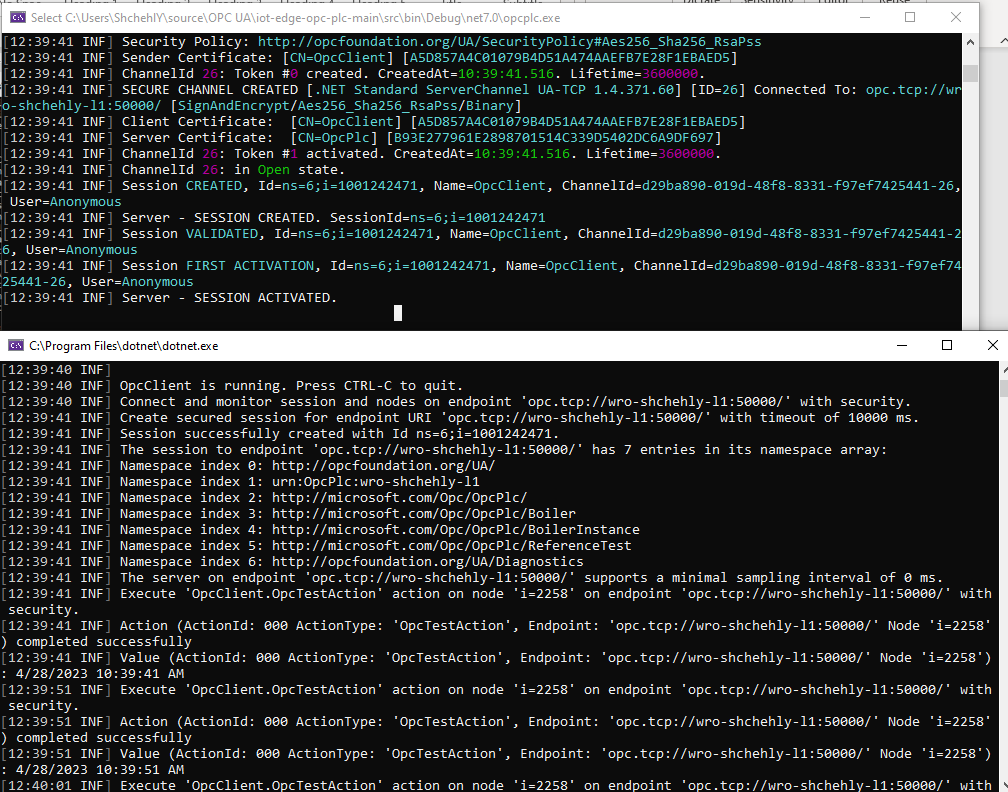
[OPC UA Certificate Stores](https://opclabs.doc-that.com/files/onlinedocs/QuickOpc/Latest/User's%20Guide%20and%20Reference-QuickOPC/Certificate%20Stores.html)



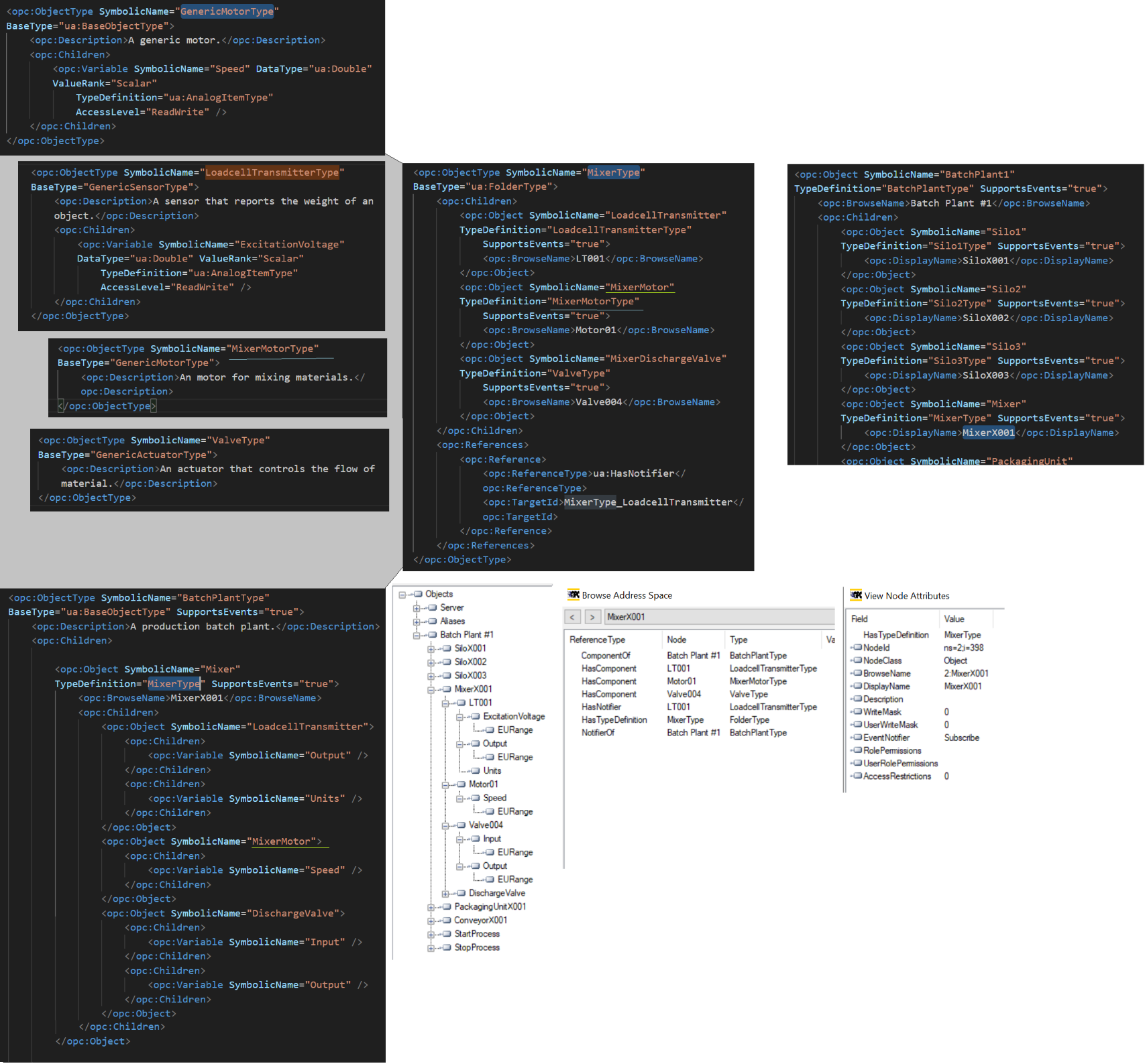
Copy from opc\_client\...\pki\own\certs to opc\_server \...\pki\trusted\certs and oposite for client and delete form pki\rejected\certs

**Lunch** client with --at X509Store arguments

Check the Program.cs:61 DefaultEndpointUrl

Result:

### Data generating

* Create custom ModelDesign.xml like <opc:Object> and convert the file using UA-ModelCompiler to c# class.
* 
* <https://github.com/OPCFoundation/UA-ModelCompiler> clone and build release
* .\Opc.Ua.ModelCompiler.exe -d2 C:\Models\ModelDesign.xml -cg C:\Models\ModelDesign.csv -o C:\Models -console -version v104
* Sample server config: 
* Copy all the generated content into ~\Data and start the server

## Opc as a worker

**Qia.Service**

* Command line arguments:

appSettings=Databases,Assemblies,Logging,Logins,Workers.OpcUa appControl=appControl configDir=C:\QIA.Service.TEST\config

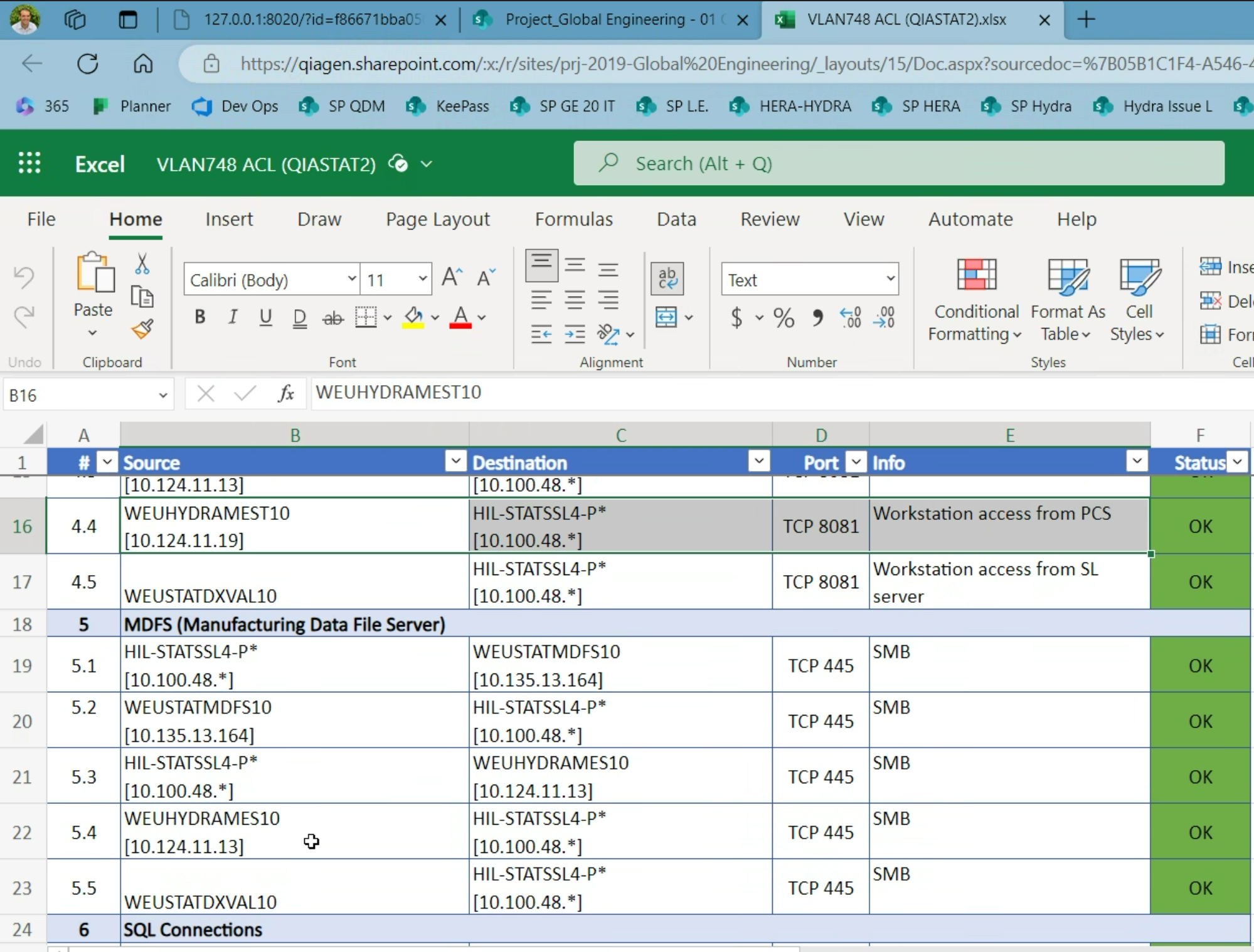
* Worker configuration:

 Check your “Location” in json

**Qia.OpcClient**

The main class (e.g. Program.cs) should derive from QIA.Library.Worker.QIAWorker abstract class and Init() method should start the worker.

## Servers



**Micron**:

WEUHYDRAMESt10

opc.tcp://10.100.48.100:4981

|  |
| --- |
| WEUSTATDXPROD10  WEUSTATDXDEV10  WEUSTATDXTEST10 |

--> cause we renamend the VAL to PROD

port for OPC UA is

TCP 4840

**Richiger**:

WEuhydrames10

## Troubleshooting

* “*waiting for WSL integration for Ubuntu: waiting for WSL distro integration to become ready in "Ubuntu": timeout*”

wsl --shutdown

wsl --update

* cannot update ubuntu or install packages

1. Make sure your VPN is open
2. If still, write to IT Support to check restrictions

## Architecture

Qia.Opc..API

|-------> Qia.Opc..Services

| |-------> Qia.Opc..OPCUAConnector

| | |-------> Qia.Opc..Common

| |

| |-------> Qia.Opc..Persistence

| |-------> Qia.Opc..Common

|

|-------> Qia.Opc..Common

1. Connect to opc server by providing url