



ACADEMY

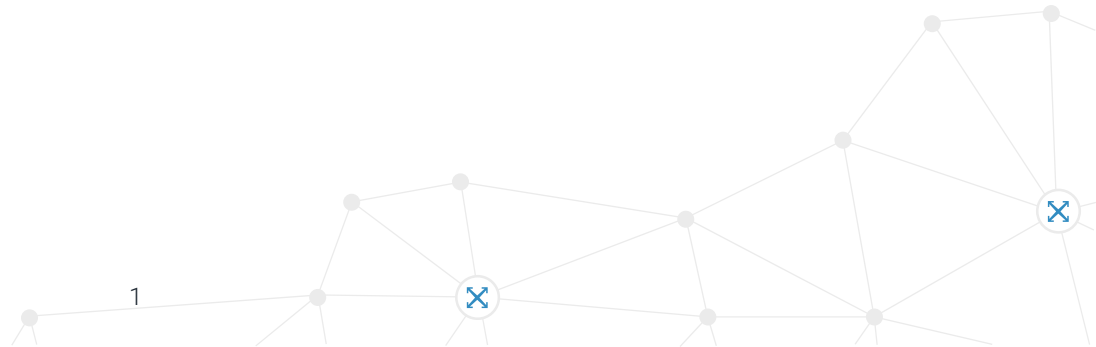


Crosser Online Training

ADVANCED SESSION

A1 - MACHINE CONNECTORS

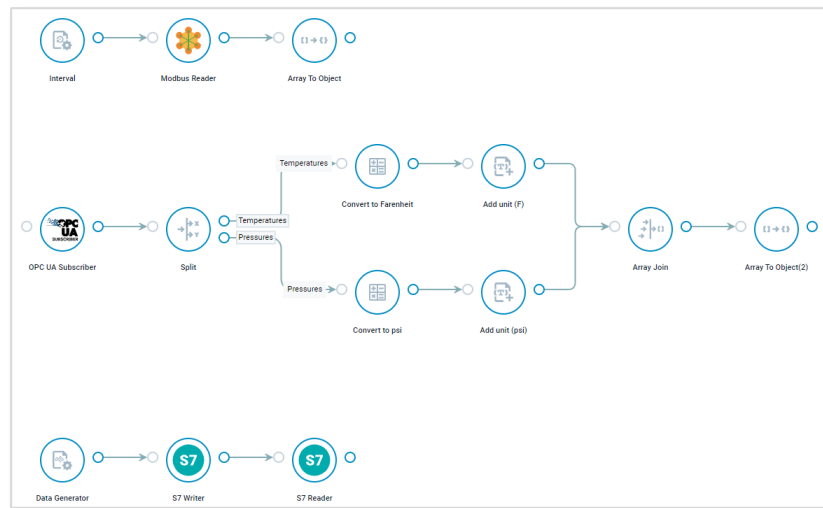
Master the different PLC and OPC UA modules



Session A1

Agenda

- Machine Connector overview
- Modules
 - Modbus Reader
 - S7 Writer
 - OPC UA Subscriber
- Tag list resources
- Exercises
 - Pull data from a PLC (Modbus) using a tag list resource
 - Subscribe to data from an OPC UA server
 - Use custom metadata
 - Write data to a PLC (S7)



Modules

Machine connectors



Modbus Reader



Modbus Writer



OPC UA Reader



OPC UA Subscriber



OPC UA Writer



S7 Reader



S7 Writer



S7 OUC Receiver



OPC UA Events



OPC UA Browser



Rockwell Reader



Rockwell Data Table Receiver



OSIsoft WebAPI Subscriber

Modules

Machine connectors – In exercises



Modbus Reader



Modbus Writer



S7 Reader



S7 Writer



S7 OUC Receiver



OPC UA Reader



OPC UA Subscriber



OPC UA Writer



Rockwell Reader



Rockwell Data Table Receiver



OPC UA Events



OPC UA Browser



OSIsoft WebAPI Subscriber

Modules

Naming conventions

Reader

→ Read selected data when triggered

Subscriber

→ Receive selected data when available

Receiver

→ Receive data when available

Writer

→ Write data when triggered

Module

Modbus Reader

- Read data from Modbus PLCs over TCP
- Data is read each time the module is triggered
- Data for all selected registers are delivered as an array
- Register mappings defined in UI or using a resource file (JSON)

Modbus Reader

Settings Common Documentation >

Name
Modbus Reader

Version
6.0.0

IP
10.7.0.66

Address to the PLC

Tags collection (Resource)
modbustest

Additional tags to monitor +

- > tag2, tag2, Read Holding Registers, Float, 0002
- > tag1, tag1, Read Input Registers, Short, 0000

Module

S7 Writer

- Write data to S7 PLCs over TCP
- Data is written each time the module is triggered
- One register written on each request
- Register mapping defined in UI or from the incoming message
- The value to write is always taken from the incoming message

Module Settings

S7 Writer
S7 Writer

Settings Common Documentation >

Name
S7 Writer

Version
1.3.0

Compatible with all

IP Address
10.6.0.77

Port
102

Rack
0

Slot
0

Tag Specification

S7 Data Area
Input

S7 Db Address
0

S7 Start Address
0

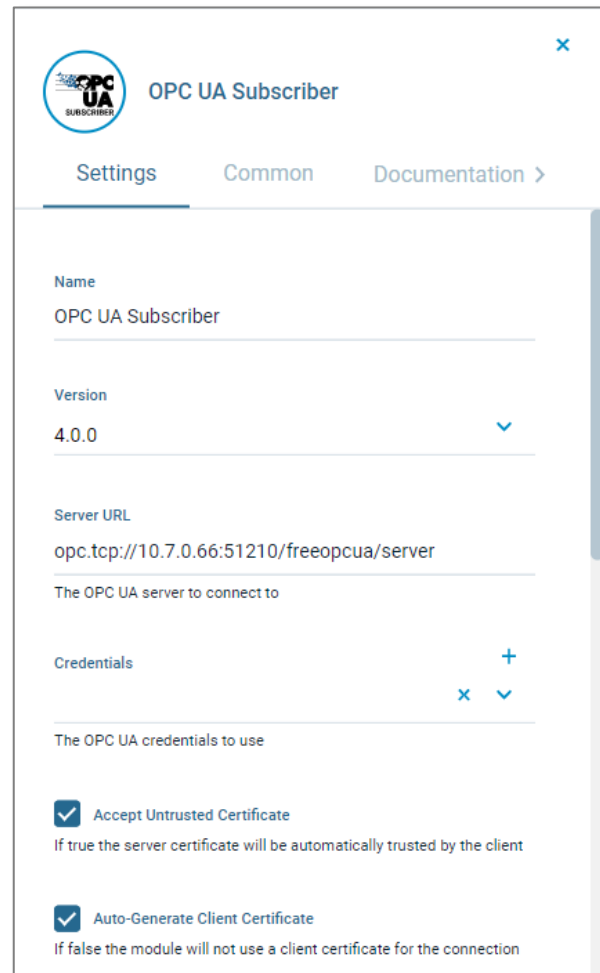
S7 Bit Address
0
Only needed when writing a boolean value

S7 Type
Bool

Module

OPC UA Subscriber

- The OPC UA Subscriber module is used to subscribe to messages from an OPC UA server.
 - Data is received when the values change.
- Specifying what data to get (OPC UA Node IDs):
 - From a resource file (JSON)
 - Manually added NodeIDs in module settings
 - **From a flow message (only on OPC UA modules)**
- Publish Interval
 - How often data can be received
 - Only nodes that have changed values will be received
 - There will be one message per NodeID
- The [OPC UA Reader](#) module is used to read all specified nodes when triggered (all specified NodeIDs in a single message)



The screenshot shows the configuration window for the 'OPC UA Subscriber' module. The window title is 'OPC UA Subscriber' and it has a close button in the top right corner. Below the title bar are three tabs: 'Settings' (selected), 'Common', and 'Documentation >'. The main content area is divided into several sections:

- Name:** A text field containing 'OPC UA Subscriber'.
- Version:** A dropdown menu showing '4.0.0'.
- Server URL:** A text field containing 'opc.tcp://10.7.0.66:51210/freeopcua/server'. Below it is a label 'The OPC UA server to connect to'.
- Credentials:** A section with a '+' icon and a dropdown menu showing 'x' and 'v' icons. Below it is a label 'The OPC UA credentials to use'.
- Accept Untrusted Certificate:** A checked checkbox with the label 'Accept Untrusted Certificate' and a sub-label 'If true the server certificate will be automatically trusted by the client'.
- Auto-Generate Client Certificate:** A checked checkbox with the label 'Auto-Generate Client Certificate' and a sub-label 'If false the module will not use a client certificate for the connection'.

Tag Lists

- Tag lists are registry mappings for PLCs and nodeId lists for OPC servers
- JSON files stored as Resources (docs [here](#)), referenced in the modules
- Useful when reading many tags (the whole list will be read)
- Add additional metadata per tag
- Can be combined with tags defined in module settings
- Can contain source specific settings, like Modbus byte order

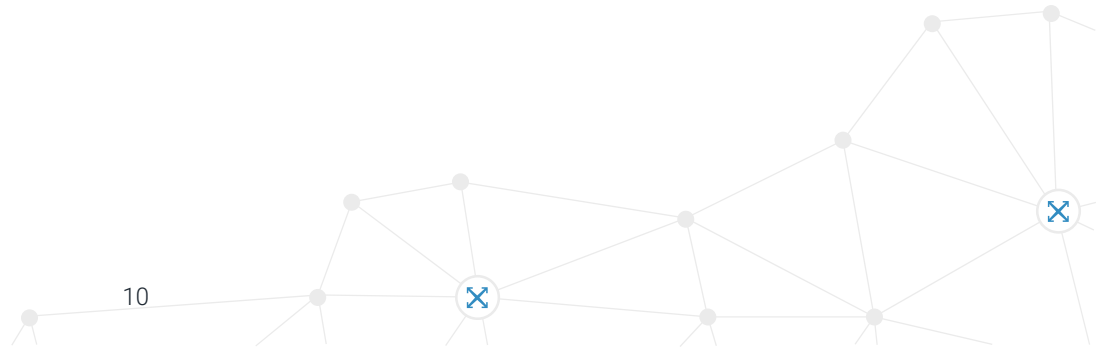
Modbus tag list

```
{
  "name": "SimPLC",
  "unitId": "1",
  "tags": [
    {
      "id": "InputReg0",
      "modbusDataType": "Short",
      "modbusFunction": "ReadInputRegisters",
      "address": "0000"
    },
    {
      "id": "InputReg1",
      "modbusDataType": "Int",
      "modbusFunction": "ReadInputRegisters",
      "address": "0001"
    }
  ],
  "byteOrder": {
    "twoByte": "10",
    "fourByte": "1032",
    "string": "01"
  }
}
```



EXERCISE A1

Working with PLC data

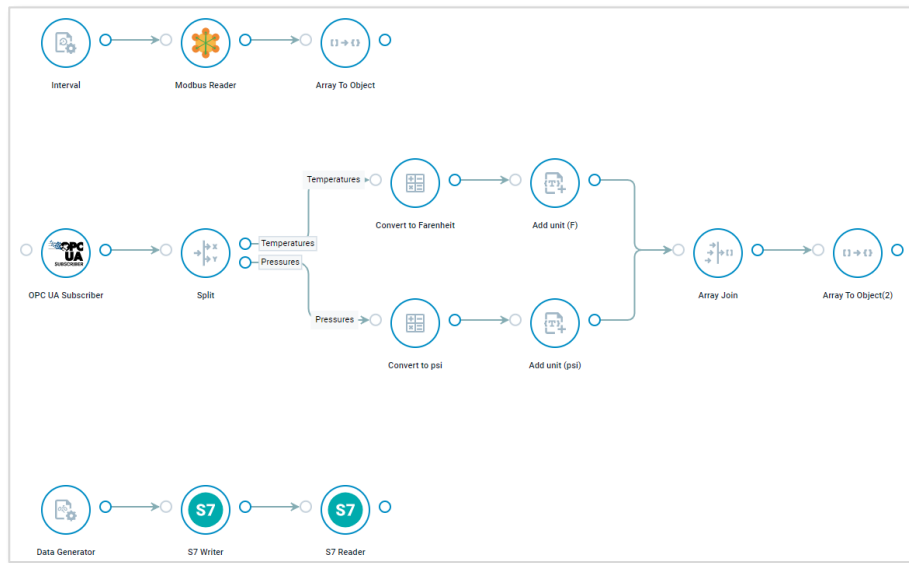


Exercise A1

Overview

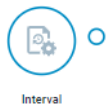
In this exercise you will learn how to:

- Pull data from a PLC (Modbus) using a tag list resource
- Subscribe to data from an OPC UA server
- Use custom metadata
- Write data to a PLC (S7)



Exercise A1.1

Modbus taglist resources



1. Create a new flow called [Exercise A1](#)
2. Add an [Interval](#) module
3. In the Flow Studio → Resources panel (right-hand side):
 - Click on “Add Resource” and then click on the ‘+’ button, to create a new resource
 - Set ‘Type’ to: [Modbus](#)
 - Copy the text in the box to the right into the UI
 - Click on “Create” to save the resource

```
{
  "name": "ModbusSim",
  "unitId": "1",
  "tags": [
    {
      "id": "tag1",
      "name": "temp1",
      "modbusDataType": "Float",
      "modbusFunction": "ReadInputRegisters",
      "address": "0000"
    },
    {
      "id": "tag2",
      "name": "pressure1",
      "modbusDataType": "Float",
      "modbusFunction": "ReadInputRegisters",
      "address": "0004"
    }
  ],
  "byteOrder": {
    "twoByte": "01",
    "fourByte": "0123",
    "eightByte": "01234567",
    "string": "01"
  }
}
```

Exercise A1.1

Pull data from PLC (Modbus Reader)



4. Add a [Modbus Reader](#) module:

- IP: [10.0.48.117](#)
- In 'Tags Collection (resource)': [Chose the resource you just created](#)

5. Run the flow and check the output from the Modbus Reader module

6. Add an [Array to Object](#) module:

- Name Property: [name](#)
 - Value Property: [value](#)
- Run the flow and check the output from the Array to Object module

Exercise A1.2

Subscribing to data from an OPC UA server



1. Create an OPC UA resource:

- Set 'Type' to: **OPC**
- Copy the text in the box to the right into the UI
- Click on "Create" to save the resource

2. Add an OPC UA Subscriber module:

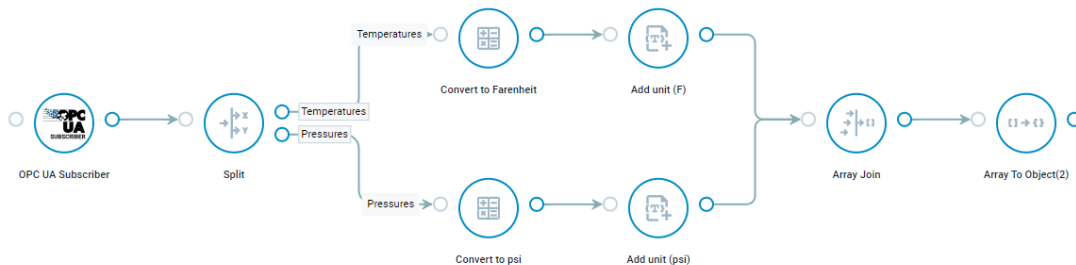
- Server URL: `opc.tcp:// 10.0.48.117 :51210/freeopcua/server/`
- In 'Tags Collection (resource)': **Chose the resource you just created**

3. Run the Flow and check the output

```
[
  {
    "nodeId": "ns=2;i=2",
    "name": "temp7",
    "unit": "C"
  },
  {
    "nodeId": "ns=2;i=3",
    "name": "temp8",
    "unit": "C"
  },
  {
    "nodeId": "ns=2;i=4",
    "name": "pressure7",
    "unit": "mBar"
  },
  {
    "nodeId": "ns=2;i=5",
    "name": "pressure8",
    "unit": "mBar"
  }
]
```

Exercise A1.2

Route data based on metadata



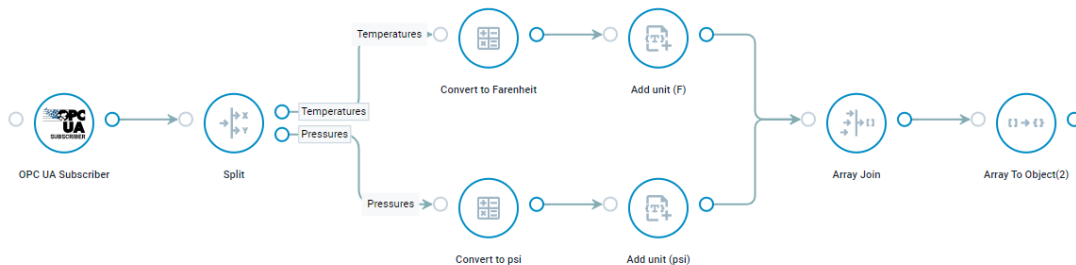
In the output from the OPC UA Subscriber module you see a property called 'unit'. This is custom metadata that was added in the resource file.

You can add any number of metadata properties on each tag, with different data types, to enhance the data received from the external source.

We will now use this information to provide conditional formatting of the tag data.

Exercise A1.2

Route data based on metadata



1. Add a **Split** module:

- Create two condition groups: **Temperatures** and **Pressures**
- In each, check if `data.unit` is equal to **C** and **mBar** respectively

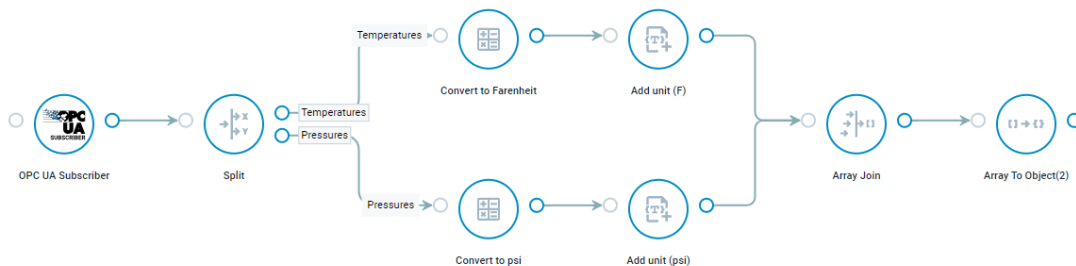
2. Add a **Math** module in each path and use the formulas from session 3 to convert the values

3. Add a Text Template module in each path, to add the unit to the sensor names:

- Target Property: `data.name`
- Template: `{data.name}_F` and `{data.name}_psi` respectively

Exercise A1.2

Combine multiple tags into a single object



4. Add an **Array Join** module:
5. Add and **Array To Object** module:
 - Name Property: **name**
 - Value Property: **value**
6. Run the Flow and check the output
(you should get an output similar to what you got in the Modbus exercise, but with some tweaks to the data)

Exercise A1.3

Writing data to a PLC (S7)



1. Add a **Data Generator** module:

- JSON template: `{"value": 1.5}`
- Type: **Double**

2. Add a **S7 Writer** module:

- IP Address: **10.0.48.117**
- Source Property: **value**
- Tag Specification:
 - S7 Data Area: **Data Block**
 - S7 Db Address: **1**
 - S7 Start Address: **80**
 - S7 Type: **Real**

Exercise A1.3

Writing data to a PLC (S7)



3. Add a **S7 Reader** module (to check the value written):

- IP Address: **10.0.48.117**
- Target Property: **data**
- Additional tags to monitor:
 - Name: **test**
 - Area: **Data Block**
 - Type: **Real**
 - Start Address: **80**
 - Db Address: **1**

4. Run the Flow and check the output

Note

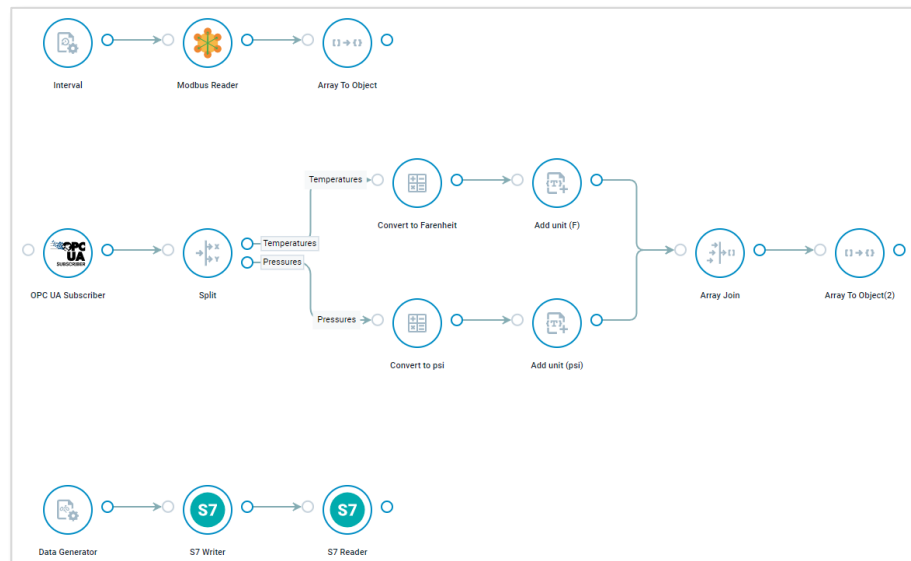
Now we provided the tag to write to in the UI settings. You can also provide the tag specification with the incoming message. This is useful when you want to write to multiple tags/registers with the same module. See the documentation for more information.

Exercise A1

Wrap-up

Things to test/consider:

- The OPC UA server sends data whenever a value changes. What happens with the output from the Array To Object module if the messages arrive at different times? (the Join module could be useful in this case)
- Modify the last exercise to take the tag specification from the incoming message



Note

The taglist resource files have different formats depending on the type of machine connector to use. Apart from that all the machine connectors work in the same way



SESSION – A1 END

PLC resources (register mappings)

Getting data from PLCs (pull and push)

Writing data to PLCs

Using metadata for conditional routing