

Edge Connect – AGV Platform Document

Written by An, Zhi Yu on Monday, Dec. 13, 2021

This document contains tutorials for connecting AGV platform with Edge Connect on IOT 2050 via OPC UA Channels.

Important sources:

1. Python-OPCUA pkg doc: <https://python-opcua.readthedocs.io/en/latest/server.html>
2. OPC UA basics: 63481236_Part1 OPC-UA-Grundlagen_en.pdf
3. Edge Connect User Guide: EdgeConnect 用户使用手册(IOT2040 & IOT2050).pdf
4. Python OPC UA server example: server_minimal.py

A valid OPC UA connection consists of at least two parts:

1. a server (which sends the data) and
2. a client (which receives, modifies, or request data).

In practice, the AGV platform and Edge Connect can both either be server and client. But during any data sharing there must be two different roles. In this tutorial, we consider the situation which AGV is the server and Edge Connect is the client.

I. Requirements

The following are system requirements for completing this tutorial:

Server (AGV platform):

1. Python 3.5+
2. opcua

Client (IoT 2050):

1. Edge Connect

The following are internet connection requirements for completing this tutorial:

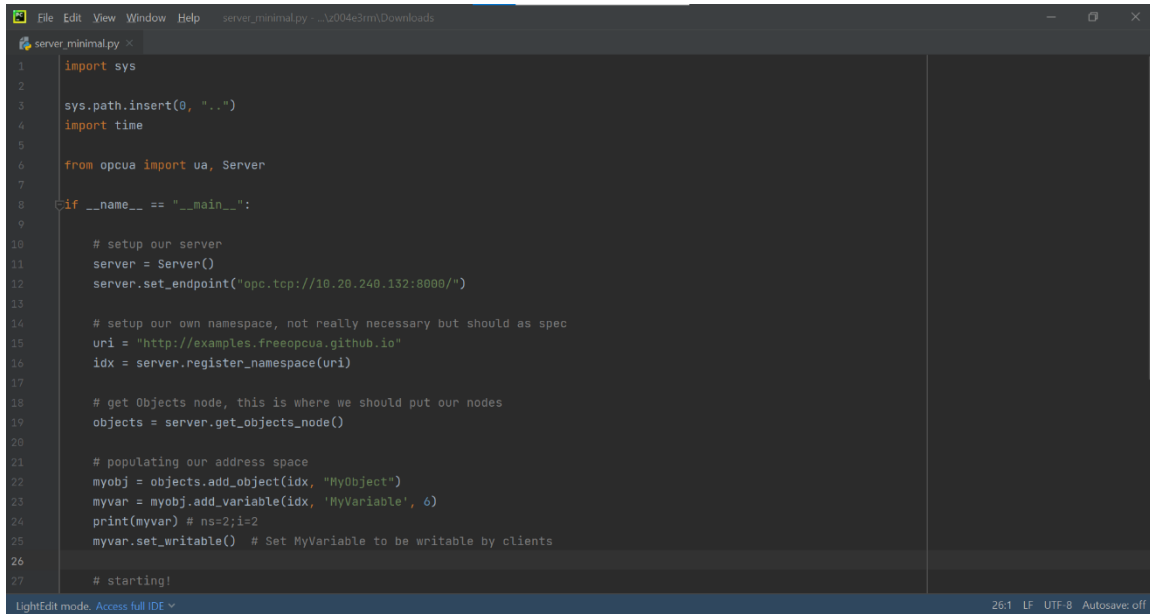
1. Server and Client must be under the same LAN.

When using a modem, this requires the AGV to wirelessly connect to a modem and IoT 2050 cabling to the same modem.

When using a 4G hotspot, this requires the AGV and IoT 2050 to connect to the same 4G hotspot.

II. Server setup

To set up a OPC UA server, launch the .py file which describes the server. Take the server_minimal.py as example:



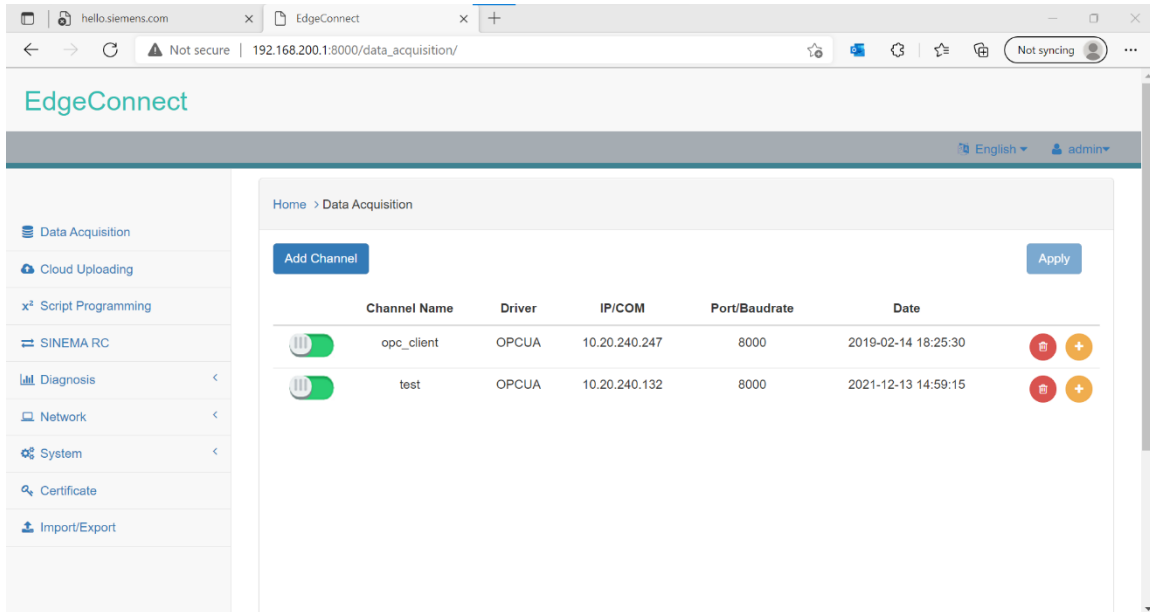
```
1 import sys
2
3 sys.path.insert(0, "..")
4 import time
5
6 from opcua import ua, Server
7
8 if __name__ == "__main__":
9
10     # setup our server
11     server = Server()
12     server.set_endpoint("opc.tcp://10.20.240.132:8080/")
13
14     # setup our own namespace, not really necessary but should as spec
15     uri = "http://examples.freopcua.github.io"
16     idx = server.register_namespace(uri)
17
18     # get Objects node, this is where we should put our nodes
19     objects = server.get_objects_node()
20
21     # populating our address space
22     myobj = objects.add_object(idx, "MyObject")
23     myvar = myobj.add_variable(idx, "MyVariable", 0)
24     print(myvar) # ns=2;i=2
25     myvar.set_writable() # Set MyVariable to be writable by clients
26
27 # starting!
```

1. In line 12, set an end point consistent with the server host's LAN. You may free to choose any port (in this case, 8000) as you like.
2. In line 15, you may free to choose any uri as long as it's secure and valid. The uri is just a way to derive "namespace".
3. In line 22 – 25, add as much variables as you need following the pattern in example. You can either use a single object or multiple objects. Note that in line 24, "print(var)" will give you the "namespace" and "index" of this variable.
4. Write update functions for the variable and start the server.

Be aware that the endpoint, namespaces, and indices are important for the steps follows and should follow the rule of simple, memorable, and distinguishable.

III. Client setup

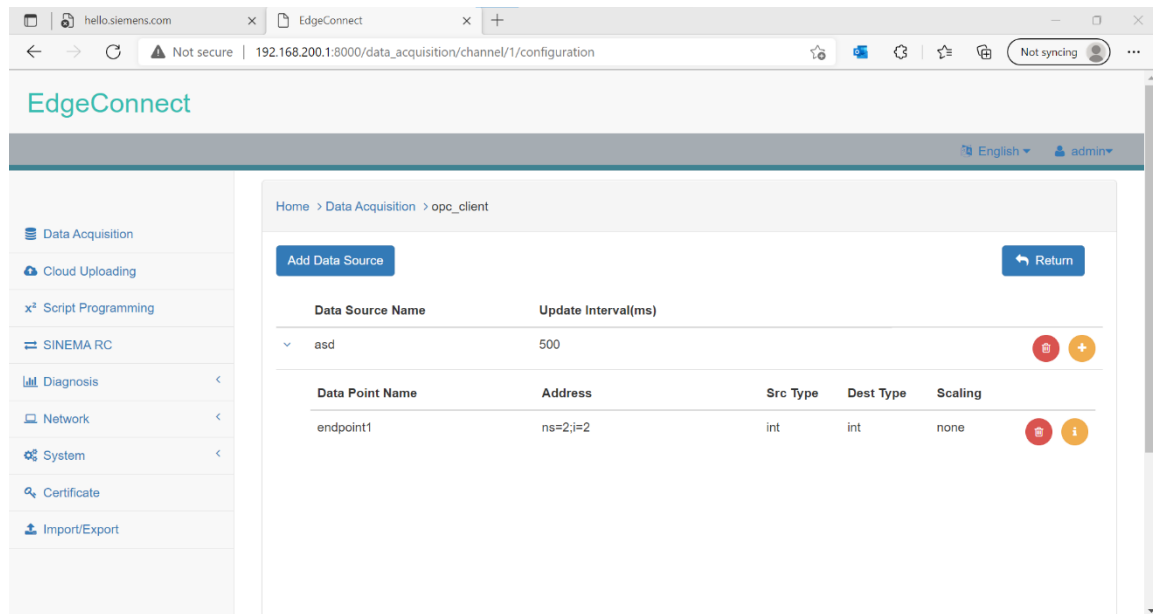
Setting up the client means to configure a OPC UA channel in the Edge Connect. We continue to illustrate using the `server_minimal.py` as the example server.



After logged in to the Edge Connect app, in the “Data Acquisition” sector, click “Add Channel”.

1. Set “Driver” to OPCUA
2. Set “IP/COM” to the endpoint of the server
3. Set “Port/Baudrate” to the port you chose for the server

Then, click the yellow “plus” button on the right.



Add the data sources and data points you need. You should map the variables you constructed in the server to these data points.

When creating the data source,

1. Choose the update interval. This is the freq. which client requests data.

Then for each data point,

1. Name the data point. This do not need to be align with variable names in the server.
2. Set the address. This is composed of the namespace and index of the variable you set in the server.

After launching the server and apply changes to the client, you are all set.