Setup and Run Instructions: OPC UA Simulator and Client

1. Introduction

This document provides detailed instructions on setting up and running an OPC UA based simulator and client project. It explains the installation of required tools such as Prosys OPC UA Simulation Server and Unified Automation UAExpert, followed by a step-by-step guide to running the client code.

2. Prerequisites

Before setting up the project, ensure that the following are available:
- A computer with Windows/Linux/Mac OS.
- Python 3.x installed (preferably Python 3.8 or above).
- pip (Python package manager).
- Prosys OPC UA Simulation Server.
- Unified Automation UAExpert Client.
- A code editor (VS Code, PyCharm, etc.) for running Python scripts.

3. Installing Required Tools

3.1 Install Python and Dependencies

1. Download and install Python 3.x from https://www.python.org/downloads/.
2. Verify installation using the command:
```
python --version
```
3. Install required Python libraries by running:
```
pip install opcua pandas openpyxl
```

3.2 Install Prosys OPC UA Simulation Server

1. Download Prosys OPC UA Simulation Server from the official site:

https://www.prosysopc.com/products/opc-ua-simulation-server/

2. Install the application and run the server.

3. By default, the server runs at opc.tcp://localhost:53530/OPCUA/SimulationServer.

4. Ensure that the server is started before running the client.

Configuring Dummy Tags:

In this project, the following 10 tags from the Prosys Simulation Server were used:

- Constant_Val
- Counter_Val
- Random_Val
- Sawtooth_Val
- Sinusoid_Val
- Square_Val
- Triangle_Val
- MyLevel_Gauge
- MySwitch_State
- Generic_Double

These tags provide a mix of static, dynamic, and simulated process values.

The client subscribes to these tags, reads their values once per minute,

and logs them into hourly sheets.

3.3 Install Unified Automation UAExpert Client

1. Download UAExpert from the official site: https://www.unified-automation.com/downloads/opc-ua-clients.html.
2. Install and launch the client.
3. Use it to connect to the Prosys Simulation Server and browse available nodes.
4. This helps in verifying that the server is running correctly and exposes nodes for communication.

4. Running the Client Code

1. Open your code editor or terminal.
2. Create a Python script (e.g., client.py) and paste the provided client code.
3. Run the script using:
```
  python client.py
```
4. The client connects to the Prosys server, subscribes to nodes, and logs the data.
5. Data will be stored in an Excel file for analysis.

The client is designed to create a **new log file at the beginning of each hour**.

- Example: If the client starts at 13:45, data from 13:45 to 13:59 will be saved in `OPC_Log_2025-09-01_13.csv`.

- At 14:00, the client automatically switches to a new file `OPC_Log_2025-09-01_14.csv`.

- This ensures logs are segmented by hour for easier analysis and compliance with assignment requirements.

5. Code Explanation

5.1 Connecting to the Server

The client connects to the Prosys Simulation Server using the opcua.Client class:
client = Client('opc.tcp://localhost:53530/OPCUA/SimulationServer')
client.connect()
This establishes communication with the OPC UA server.

5.2 Subscribing to Nodes

The client subscribes to specific nodes (variables) on the server. Whenever the node value changes, a callback function is triggered to log the data. This allows real-time monitoring of server values.

5.3 Logging Data to Excel

The logged data (timestamp and value) is stored in a Pandas DataFrame and then exported to Excel using the openpyxl library. This makes it easier to analyze data later using Excel.

Data Reading Frequency:

The client reads values once per minute.

This interval can be adjusted in the Python script if needed, but for assignment purposes it is fixed at 60 seconds.

6. Verification and Debugging

- Use UAExpert to verify that the server is running and that nodes are accessible.
- Ensure Python client code uses the correct server endpoint (opc.tcp://localhost:53530/OPCUA/SimulationServer).
- If Excel logging shows incorrect timestamps, check time zone settings in both the client machine and Excel.
- Use print statements in Python code to debug connections or subscription issues.

7. Conclusion

By following the above steps, you can successfully set up the Prosys OPC UA Simulation Server, connect it with UAExpert for browsing nodes, and run the Python client to log and analyze real-time data. This setup forms the foundation for industrial OPC UA applications and provides hands-on experience with client-server architectures.