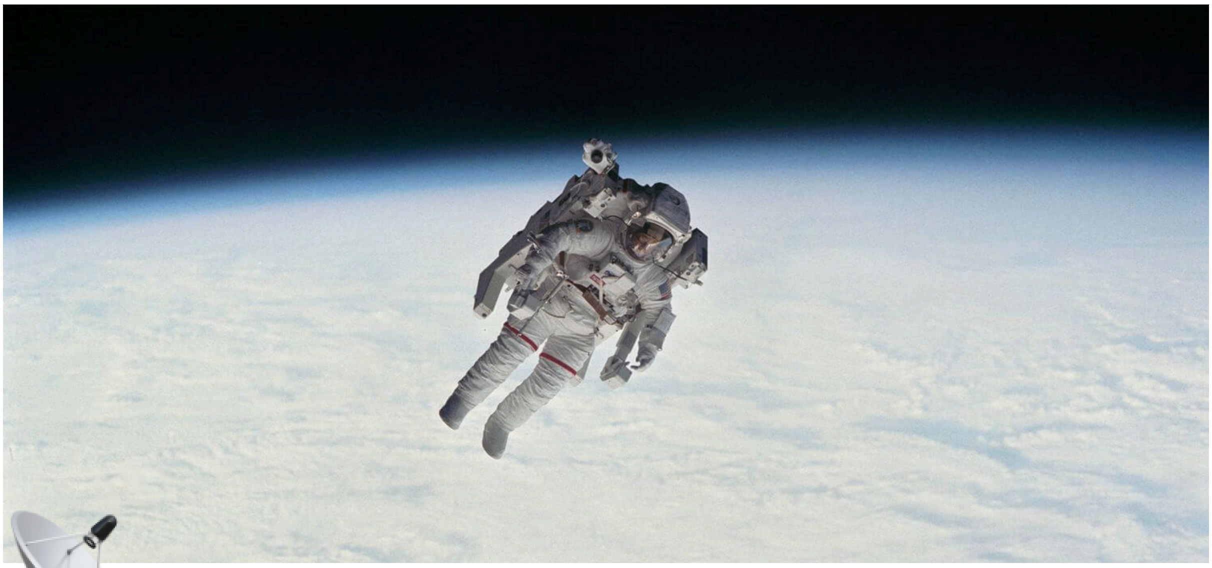

OPC UA

OPC UA (Open Platform Communications Unified Architecture) is a platform-independent, secure, and scalable protocol for industrial automation. It is designed for the reliable exchange of data between devices, systems, and software in industrial environments, such as manufacturing plants, energy systems, and IIoT (Industrial Internet of Things) applications

Key Features of OPC UA

1. **Platform Independence**:
    - Works across operating systems and hardware platforms.
    - Supports a variety of transport mechanisms like TCP, HTTPS, and WebSockets.
2. **Unified Architecture**:
    - Combines the functionality of older OPC specifications (like OPC DA, AE, HDA) into a single, unified protocol.
    - Supports data acquisition, event handling, and historical data access.
3. **Security**:
    - Built-in encryption, authentication, and user control.

    - Supports X.509 certificates for secure communication.

4. **Scalability**:
    - Suitable for small embedded devices as well as large enterprise systems.
5. **Standardized Information Models**:
    - Allows structured representation of data (e.g., objects, variables, and methods).
    - Enables complex systems to exchange meaningful data seamlessly.

How OPC UA Works

1. **Client-Server Model**:
    - **Server**: A device or system (e.g., PLC or database) that provides data.
    - **Client**: A software application or device that consumes or uses the data provided by the server.
2. **Data Exchange**:
    - OPC UA servers expose their data via **nodes** organized in a hierarchical namespace.
    - Clients connect to the server, browse the namespace, and read/write to nodes.
3. **Communication Layers**:
    - **Application Layer**: Defines the information models and client-server interaction.
    - **Transport Layer**: Handles message exchange (e.g., using TCP or HTTPS).

Benefits of OPC UA

1. **Interoperability**:

    Facilitates seamless communication between devices and systems from different vendors.

2. **Reliability**:

   Ensures data integrity and robust error handling.

3. **Future-Ready**:

   Supports modern technologies like IoT, cloud computing, and edge devices.

4. **Enhanced Security**:

   Mitigates cybersecurity risks with robust encryption and authentication mechanisms.

5. **Simplified Integration**:

   Reduces the complexity of connecting different systems in an industrial setup.

Comparison: OPC UA vs Traditional OPC

Feature	OPC Classic	OPC UA
Platform	Windows only	Cross-platform
Security	Limited	Advanced (encryption, certificates)
Transport Protocol	COM/DCOM	TCP, HTTPS, WebSockets
Scalability	Limited	Highly scalable
Information Models	Basic (flat namespace)	Rich and extensible

https://github.com/cacamille3/OPC-UA-Clients

Final Testing -

Step 1: Install an OPC UA Server on Windows 11

Option 1: Unified Automation OPC UA C++ Demo Server (Best for UAExpert)

1. **Download & Install**

   - Download the **OPC UA C++ Demo Server** from **Unified Automation**.

   - Install it and start the server.
2. **Get the Server Endpoint**
   - Open **UA C++ Demo Server**.
   - Note down the **Endpoint URL** (default: `opc.tcp://<Windows-IP>:4840`).
   - Find your **Windows IP** by running:(Look for "IPv4 Address".)

     ```
     CopyEdit
     ipconfig
     ```
3. **Enable Anonymous Authentication (For Testing)**
   - In the **UA C++ Demo Server settings**, enable **Anonymous Login** so you don't need credentials.

Alternative Server Options (If Needed)

Option 2: FreeOpcUa Python-Based Server (Lightweight)

If you prefer a script-based solution:

1. Install Python:

   ```
   sh
   CopyEdit
   pip install opcua
   ```

2. Run this simple server:

   ```
   python
   CopyEdit
   from opcua import Server
   from datetime import datetime
   import time

   server = Server()
   server.set_endpoint("opc.tcp://0.0.0.0:4840/freeopcua/server/")
   ```

```
idx = server.register_namespace("http://example.org")
myobj = server.nodes.objects.add_object(idx, "DataNode")
timestamp_var = myobj.add_variable(idx, "Timestamp", str(datetime.no
w()))
timestamp_var.set_writable()

server.start()
try:
    while True:
        time.sleep(1)
finally:
    server.stop()
```

3. Run the script:

```
sh
CopyEdit
python opc_ua_server.py
```

Step 2: Install UAExpert on macOS and Connect

1. **Download & Install UAExpert**
    - Download from Unified Automation.
    - Install it on your **MacBook Pro M1 Pro**.
2. **Add the OPC UA Server Connection**
    - Open **UAExpert**.
    - Go to **Server → Add Server**.
    - Enter the server's **Endpoint URL**:

    ```
    cpp
    CopyEdit
    ```

```
opc.tcp://<Windows-IP>:4840
```

- Click **Connect**.

Step 3: Send Timestamps Every 1 Second from Mac

1. In **UAExpert**, navigate to **Address Space**.
2. Find the **"Timestamp"** variable (or create one in your server).
3. **Write values manually**:
    - Right-click the variable → **Write** → Enter a timestamp.
4. **Automate Writing Every 1 Second**:
    - Use the **Scripting Panel** or **Data Generator** (if available).
    - If UAExpert doesn't support automated writing, use a **Python script** on Mac to write timestamps via OPC UA.

Final Checklist

✔ **Windows Firewall allows port 4840 (or 53530, depending on the server).**

✔ **Mac and Windows are on the same network.**

✔ **Anonymous authentication is enabled for easy testing.**

✔ **UAExpert successfully connects to the OPC UA server.**

Step 1: Connect UAExpert (macOS) to the OPC UA Server (Windows 11)

1. **Launch UAExpert** on your **MacBook Pro**.
2. **Add a new server connection**:
    - Go to **"Server" → "Add Server..."**
    - In the **"Custom Discovery"** tab, enter your server's endpoint:

```
cpp
CopyEdit
```

```
opc.tcp://GamerTop:48010
```

  - Click **"OK"** → Then **"Connect"**.
  - If prompted for authentication, select **Anonymous** (for now).

3. **Verify connection**:
  - If successful, you should see the **Address Space** (list of variables from the server).
  - If it fails, check:
    - Windows **Firewall settings** (allow inbound traffic on port **48010**).
    - **Network connectivity** (ensure both devices are on the same network).
    - Try using the **Windows IP address** instead of `GamerTop`:(Find **Windows IP** using `ipconfig` in Command Prompt.)

      ```
      cpp
      CopyEdit
      opc.tcp://<Windows-IP>:48010
      ```

Step 2: Create a Writable Timestamp Variable

Now, let's create a **new variable** on the server where the Mac will send timestamps.

Using UAExpert to Create a New Variable

1. In **UAExpert**, go to the **Address Space** window.
2. Navigate to **Objects → Server → Right-click** → **Add Node**.
3. Select **"Variable"**, name it `"Timestamp"` (or similar).
4. Set **Data Type** to `"String"` (for ISO 8601 timestamps).
5. Set **Access Level** to **Read & Write**.
6. Click **OK** to create the variable.

Step 3: Send a Timestamp Every 1 Second from macOS

Since **UAExpert** does not have an in-built automation tool for writing values, use **Python** on macOS to update the variable.

Install OPC UA Python Library on macOS

1. Open **Terminal** and install:

   ```
   sh
   CopyEdit
   pip3 install opcua
   ```

Final Checklist

✔️ **UAExpert on macOS connects successfully to the server.**

✔️ **A writable "Timestamp" variable is created on the server.**

✔️ **Firewall allows traffic on port 48010.**

✔️ **Python script on macOS updates the timestamp every 1 second.**

You're all set! 🎯 Would you like to log this data or visualize it in Grafana? 🚀

2️⃣ Install Missing Cryptography Library

The warning **"cryptography is not installed, use of crypto disabled"** suggests missing dependencies.

Install cryptography and opcua again

```
sh
CopyEdit
pip3 install cryptography opcua --upgrade
```

3️⃣ Check Windows Firewall

Your **Windows Firewall** might be **blocking** the connection.

Allow inbound traffic on port 48010

1. **Open Windows Defender Firewall** → Click **Advanced settings**.
2. Go to **Inbound Rules** → Click **New Rule**.
3. Select **Port** → Click **Next**.
4. Choose **TCP** and enter **48010** → Click **Next**.
5. Select **Allow the connection** → Click **Next**.
6. Apply to **Private, Public, and Domain** networks.
7. Name it **"OPC UA Server Rule"** and click **Finish**.

4️⃣ Test Connection with UAExpert on Mac

Before running the script, test if **UAExpert** on macOS can connect.

1. Open **UAExpert** on MacBook.
2. **Add Server → Custom Discovery**.
3. Enter the **new server URL**:

   ```
   cpp
   CopyEdit
   opc.tcp://192.168.1.100:48010
   ```

4. **Connect** → If successful, the issue was with hostname resolution.

5️⃣ Debug with telnet (Optional)

To verify if your Mac can reach the server:

```
sh
CopyEdit
telnet 192.168.1.100 48010
```

- **If it connects**, the problem is in your script.
- **If it fails**, something is blocking the connection (likely firewall).

Final Steps

✅ Use **Windows IP address** in Python script.

✅ Install **cryptography** package.

✅ Allow **port 48010** in Windows Firewall.

✅ Confirm **UAExpert** can connect before running Python.

✅ Test **telnet connection** from Mac to Windows.

Try these fixes and let me know if you still face issues! 🚀

4o

You said:

how to change the url on pc with uaccpserver app

ChatGPT said:

To change the **OPC UA server URL** on your Windows **UA C++ Demo Server (uaccpserver)**, follow these steps:

🔹 Method 1: Modify the `uaserverc++.ini` Configuration File

1. **Locate the Configuration File**
    - Navigate to:

      ```
      makefile
      CopyEdit
      C:\ProgramData\UnifiedAutomation\UaServerCPP\
      ```

    - Find the `uaserverc++.ini` file.
2. **Edit the Configuration File**
    - Open `uaserverc++.ini` in a text editor (**Notepad++ recommended**).
    - Look for the section:

      ```
      ini
      CopyEdit
      [Server]
      ```

```
Endpoint1=opc.tcp://GamerTop:48010
```

- Change `GamerTop` to your **Windows IP** or a custom hostname:(Change **IP address & port** as needed.)

  ```
  ini
  CopyEdit
  Endpoint1=opc.tcp://192.168.1.100:4840
  ```

3. **Save and Restart the Server**
   - Save changes.
   - Restart the **UA C++ Demo Server**.

🔷 Method 2: Change via UAExpert (If Supported)

1. Open **UAExpert** on Windows.
2. Go to **Servers** → **Select your UA C++ Server**.
3. If an option is available to change the **endpoint URL**, modify it.

🔷 Method 3: Run with Custom Command Line (Temporary)

Instead of modifying the INI file, you can launch the server with a **custom URL**:

1. Open **Command Prompt** and navigate to the **server installation folder**:

   ```
   sh
   CopyEdit
   cd "C:\Program Files\UnifiedAutomation\UaCPPServer\bin"
   ```

2. Run the server with a **new endpoint**:(Replace the **IP and port** as needed.)

   ```
   sh
   CopyEdit
   ```

```
uaccpserver.exe -endpoint opc.tcp://192.168.1.100:4840
```

5️⃣ Connect UAExpert (Windows) to MacBook

1. Open **UAExpert** on your Windows machine.
2. Click **Add Server** → **Custom Discovery**.
3. Enter your **MacBook's IP**:

   ```
   cpp
   CopyEdit
   opc.tcp://192.168.x.x:4840
   ```

4. Click **Connect** and **Monitor** `Timestamp` **Variable**.

6️⃣ (Optional) Firewall Rules for Mac

If Windows can't connect:

- Allow Python in **Mac's firewall**:

  ```
  sh
  CopyEdit
  sudo /usr/libexec/ApplicationFirewall/socketfilterfw --add /usr/bin/python3
  ```

- Open **port 4840**:

  ```
  sh
  CopyEdit
  sudo pfctl -f /etc/pf.conf
  sudo pfctl -e
  ```

✅ Final Test

- **MacBook (Server)** should send timestamps.
- **Windows (UAExpert Client)** should receive them.

🔷 How to Find Your MacBook's IP

Run this command in **Terminal**:

```
sh
CopyEdit
ifconfig | grep "inet " | grep -v 127.0.0.1
```

Look for an **IP like** `192.168.x.x` (e.g., `192.168.0.100`).

✅ Steps to Connect from UAExpert (Windows)

1. Start the **server** on your MacBook:You should see:

   ```
   sh
   CopyEdit
   python3 opcua_server.py
   ```

   ```
   pgsql
   CopyEdit
   ✅ OPC UA Server started on opc.tcp://192.168.0.100:4840
   Updated Timestamp: 2025-02-16T12:00:00.000000
   ```

2. Open **UAExpert** on Windows.
3. Click **Add Server** → **Custom Discovery**.
4. Enter:

   ```
   cpp
   CopyEdit
   ```

```
opc.tcp://192.168.0.100:4840
```

5. Click **Connect** and monitor the `Timestamp` variable.

✅ Final Code (with Correct IP)

```
python
CopyEdit
from opcua import Server
from datetime import datetime
import time

# Create OPC UA Server
server = Server()

# Set Endpoint - Use your actual MacBook IP
server_ip = "192.168.0.197"
server.set_endpoint(f"opc.tcp://{server_ip}:4840")

# Set Server Name
server.set_server_name("MacBook OPC UA Server")

# Create a new namespace
uri = "http://macbook-opcua"
idx = server.register_namespace(uri)

# Add a variable node for timestamp
timestamp_node = server.nodes.objects.add_variable(idx, "Timestamp", "")

# Set the variable to be writable
timestamp_node.set_writable()

# Start the server
server.start()
print(f"✅ OPC UA Server started on opc.tcp://{server_ip}:4840")

try:
    while True:
        # Update timestamp every second
        current_time = datetime.now().isoformat()
        timestamp_node.set_value(current_time)
        print(f"Updated Timestamp: {current_time}")
        time.sleep(1)

except KeyboardInterrupt:
    print("Stopping Server...")
    server.stop()
    print("Server Stopped.")
```

🔷 Steps to Connect from UAExpert (Windows)

1. Start the **server** on your MacBook:

   ```
   sh
   CopyEdit
   python3 opcua_server.py
   ```

   You should see:

   ```
   pgsql
   CopyEdit
   ✅ OPC UA Server started on opc.tcp://192.168.0.197:4840
   Updated Timestamp: 2025-02-16T12:00:00.000000
   ```

2. Open **UAExpert** on Windows.
3. Click **Add Server** → **Custom Discovery**.
4. Enter:

   ```
   cpp
   CopyEdit
   ```

```
opc.tcp://192.168.0.197:4840
```

5. Click **Connect** and check if `Timestamp` updates every second.

🔷 How to Run & Connect to UAExpert

1. **Run the server on MacBook**

   Open **Terminal** and execute:

   ```
   sh
   CopyEdit
   python3 opcua_server.py
   ```

   You should see:

   ```
   pgsql
   CopyEdit
   ✅ OPC UA Server started on opc.tcp://192.168.0.197:4840
   Updated Timestamp: 2025-02-16T12:00:00.000000
   ```

2. **Connect from UAExpert on Windows**
   - Open **UAExpert**.
   - Click **Add Server** → **Custom Discovery**.
   - Enter:

     ```
     cpp
     CopyEdit
     opc.tcp://192.168.0.197:4840
     ```

   - Click **Connect**.
   - You should see the `Timestamp` variable updating **every second**.

1️⃣ Check if Your Server is Running on MacBook

On your MacBook, open **Terminal** and **run the server**:

```
sh
CopyEdit
python3 opcua_server.py
```

You should see output like:

```
pgsql
CopyEdit
✅ OPC UA Server started on opc.tcp://192.168.0.197:4840
Updated Timestamp: 2025-02-16T12:00:00.000000
```

If this is **not running**, then **fix any errors first**.

2️⃣ Test the Connection on MacBook Itself

Before connecting from Windows, **test locally on your MacBook**.

📌 Install and Run a Local OPC UA Client

1. Open a new **Terminal** window.
2. Install the OPC UA Python client if you haven't:

   ```
   sh
   CopyEdit
   pip install opcua
   ```

3. Run this **client test script** on MacBook:

   ```
   python
   CopyEdit
   from opcua import Client
   ```

```
client = Client("opc.tcp://192.168.0.197:4840")

try:
    client.connect()
    print("✅ Connected to OPC UA Server on MacBook!")
except Exception as e:
    print(f"❌ Connection failed: {e}")
finally:
    client.disconnect()
```

4. If this **fails**, the server is not running properly.

3️⃣ Allow Incoming Connections on MacBook (Firewall)

If UAExpert **can't connect**, Mac's firewall may be blocking the **OPC UA server**.

📌 Disable Mac Firewall (Temporary)

1. Open **System Settings** → **Network** → **Firewall**.
2. Turn **Firewall OFF** (just to test).
3. Try connecting again from **UAExpert on Windows**.
4. If it **works**, re-enable the firewall and allow only OPC UA traffic.

📌 Allow OPC UA Port in Mac Firewall

If you want to **keep the firewall ON**, allow port `4840`:

1. Open **Terminal** and run:

   ```
   sh
   CopyEdit
   sudo pfctl -f /etc/pf.conf -e
   ```

2. Add a rule to allow **port 4840**:

```
sh
CopyEdit
echo "pass in proto tcp from any to any port 4840" | sudo pfctl -a com.apple/250.CustomFirewall -f -
```

4️⃣ Check Windows Connection (ping & telnet)

On Windows, open **Command Prompt** and check:

📌 Test if Windows Can Reach Mac

```
sh
CopyEdit
ping 192.168.0.197
```

- If it **fails**, Windows cannot communicate with the Mac.
- If it **works**, proceed to the next step.

📌 Check if OPC UA Port is Open

```
sh
CopyEdit
telnet 192.168.0.197 4840
```

- If you see **a blank screen**, the connection works!
- If it says **"Could not open connection"**, something is blocking the port.

5️⃣ Configure UAExpert to Connect

1. **Open UAExpert on Windows**.
2. Click **Add Server** → **Custom Discovery**.
3. Enter:

```
cpp
CopyEdit
opc.tcp://192.168.0.197:4840
```

3 March 2025 -

Tasks -

PLC PC comms to other pc

My project statement is to use OPC UA to communicate between a PC to PC but one pc has an PLC connected to it which is sending field data is - ON OFF signal / timestamp / date. but the data via the plc is read by modbus pull on an windows laptop which is generating an live excel file which has the on off data, which I want to transfer to an UAexpert server which will show all the filed data which I have stated above

2. Set Up an OPC UA Server

- Use **open62541 (C-based OPC UA stack), FreeOpcUa (Python-based), or Prosys OPC UA SDK (Java-based)**.
- If you prefer a no-code approach, **Kepware OPC Server** can read Excel data and act as an OPC UA server.

3. Write a Script to Update OPC UA Nodes

- Implement an OPC UA server that updates nodes with values from your Excel sheet.
- Modify `path_to_your_live_excel.xlsx` to point to the actual file.

Python OPC UA Server Implementation

You can modify the previous script to add a timestamp and date when updating OPC UA nodes:

Steps to Integrate with UAExpert

1. **Run the above Python script** on the Windows laptop that has the Excel file.

2. **Open UAExpert** and connect to:

   ```
   bash
   CopyEdit
   opc.tcp://your_pc_ip:4840/freeopcua/server/
   ```

3. **Browse for** `PLC_Data` and add the ON/OFF, Timestamp, and Date variables.
4. **Observe live updates** in UAExpert as the Excel file changes.

Next Steps

- If you want to match the timestamp with when the Excel entry was written (rather than when it is read), you may need to modify your Excel logging process to include timestamps at the time of data entry.
- Do you need the timestamp from an external source like the PLC instead of the system time? If so, let me know how the PLC provides data.