CHIPKIN

OPC UA: Open Platform Communications United Architecture Ethernet Driver FS-8705-51

Chipkin - Enabling Integration

salesgroup1@chipkin.com

Tel: +1 866 383 1657

© 2021 CHIPKIN AUTOMATION SYSTEMS

Driver Version: 1.0.2
Document Revision: 5

TABLE OF CONTENTS

© 2021 CHIPKIN AUTOMATION SYSTEMS

© 2021 CHIPKIN AUTOMATION SYSTEMS

1 OPC UA Description

The OPC UA driver allows the FieldServer to transfer data to and from devices over Ethernet using the OPC UA protocol. The FieldServer can emulate either a Server or Client.

If configured as an OPC UA Client, the FieldServer will create socket connections to configured OPC UA Server endpoints. Upon successfully connecting, the driver will create client sessions and using the session, will poll for the requested data points.

If configured as an OPC UA Server, the FieldServer creates one OPC UA endpoint that other OPC UA Clients can connect to. Nodes are added to OPC Server to store data points and tasks are configured to map data stored in the FieldServer (usually from other drivers) to the Value attribute of the nodes.

In OPC UA, data is stored in nodes. Each node has a unique nodeId which is a combination of a namespace and identifier. See Appendix A.4 for more information on nodeIds. Each node has various attributes, the actual data value and sometimes references to other nodes. For a list of supported data types and attributes, please refer to Append A.2 and A.3 respectively.

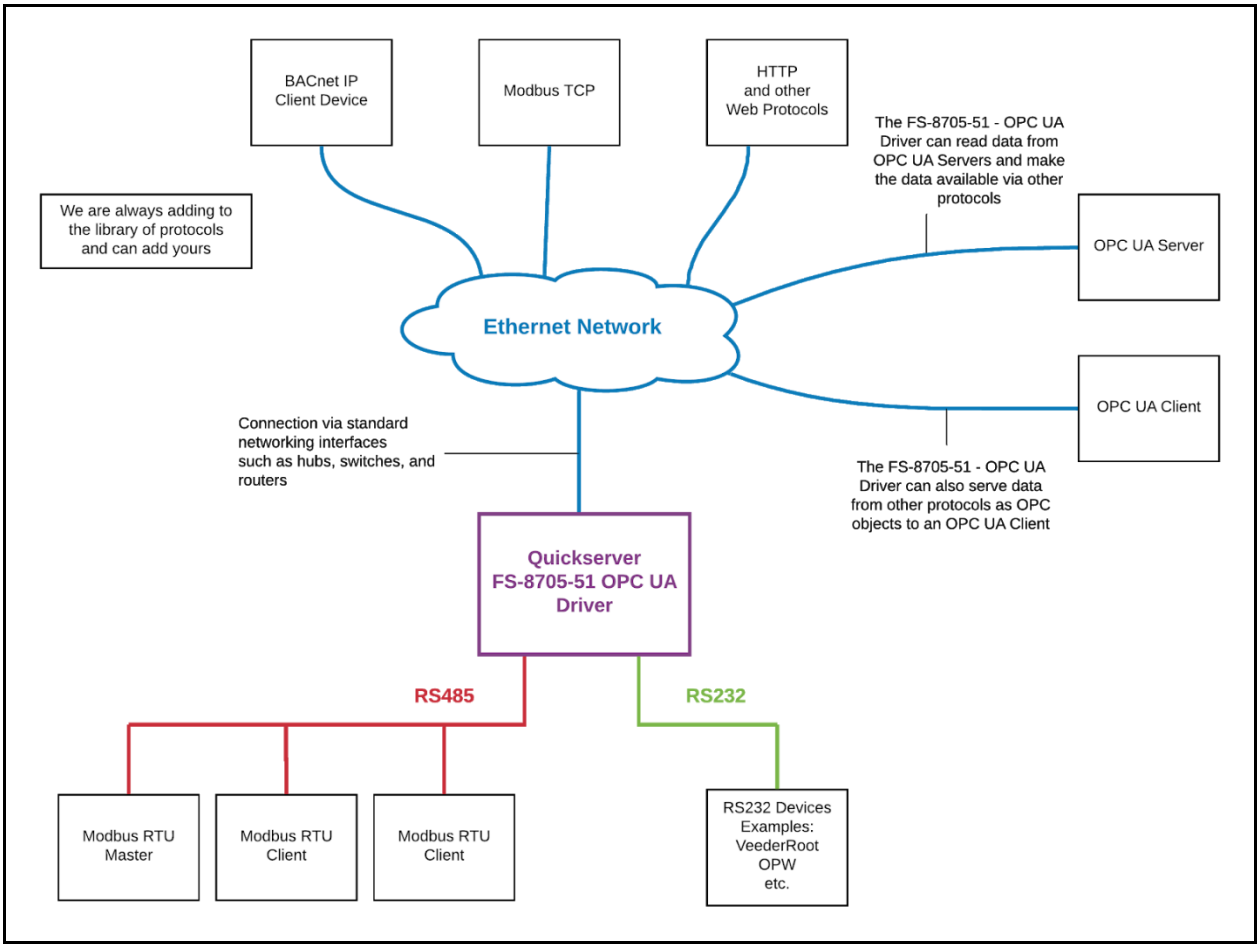
The information that follows describes how to expand upon the factory defaults provided in the configuration files included with the FieldServer.

.

© 2021 CHIPKIN AUTOMATION SYSTEMS

2 Connection Diagram

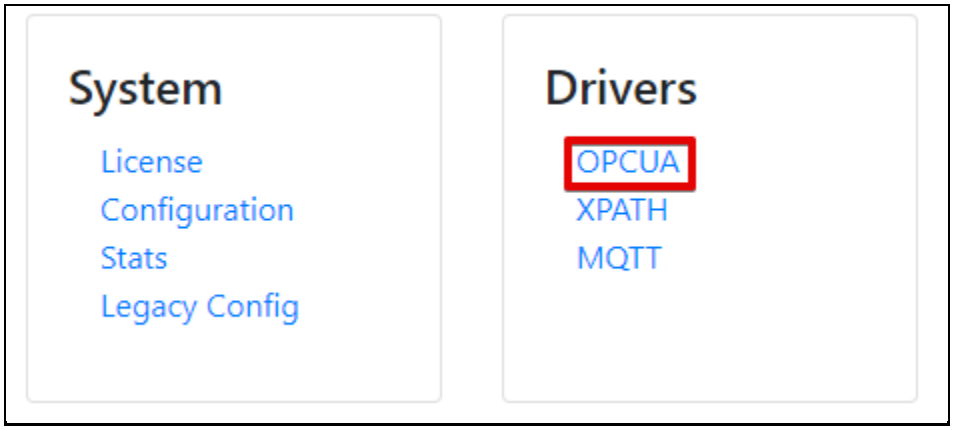
This block diagram lists common network connections that can monitor and/or serve OPC UA data using other protocols like Modbus® RTU/TCP, BACnet® and HTTP.



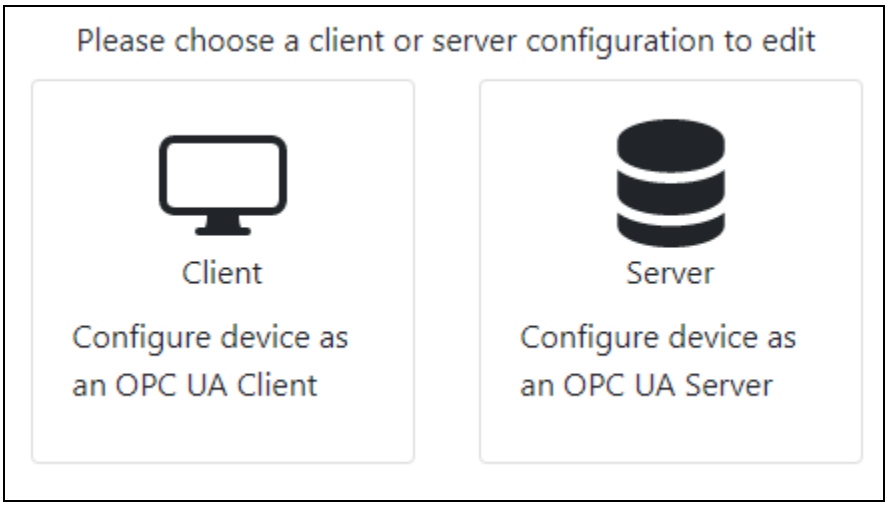
© 2021 CHIPKIN AUTOMATION SYSTEMS

3 OPC UA Configuration

To configure the OPC UA driver, from the home page, click on the OPC UA Driver link or by visiting http://{IP_ADDRESS}/opcuaDriver/ui/



On the OPC Configuration screen, select to configure the Fieldserver as an OPC UA Client or Server.



3.1 OPC UA Server Configuration

This section documents and describes the parameters necessary for configuring the FieldServer to communicate with an OPC UA Client.

The configure the FieldServer as an OPC UA Server, follow the instructions below to add a Connection (contains information about the server), Nodes (UA Objects organized by folders), and finally Tasks (UA Variables that belong to the UA Objects)

© 2021 CHIPKIN AUTOMATION SYSTEMS

3.2 Create Connection.

To set up an OPC Server, first create a connection. The connection contains information about this device acting as an OPC Server and how OPC UA Clients will connect to it.

1. Click on the "Create Connection" button to open the Create Connection form.
2. Fill out the fields in the form. The fields are as follows:

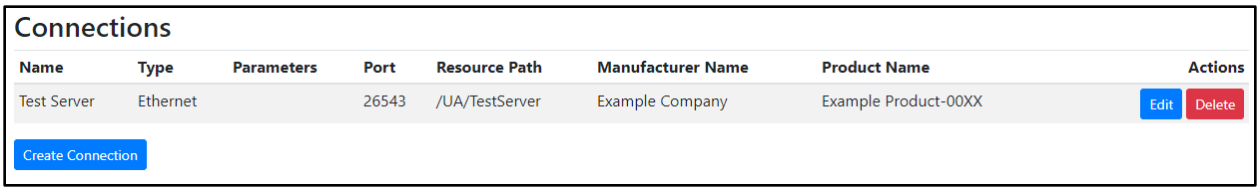
Column Title	Function	Legal Values
Name	Name of the server, used internally as an identifier	Text, must be unique
Type	The type of connection this is	**ethernet**
Parameters: Port	The physical port on the FieldServer to use	**n1**
Port	The TCP listening port for the OPC UA Server	**26543**, 4840, available TCP port
Hostname	The IP Address or domain host name of the device running the server.	**192.168.1.72**, example-pc
ResourcePath	The URI endpoint for the server. Examples: /UA or /UA/MyDevice. If blank, the uri is simply the root endpoint: opc.tcp://{IP Address}:{port}	/UA, /UA/TestServer, any text that specifies a path format
ManufacturerName	The name of the manufacturer of the server	Text, **Chipkin Automation Systems**
ProductName	The name of the product	Text, **CAS FS OPC UA Server**

* Bolded values are defaults

3. Click the "Save" button to add the connection.

   If successful, the new entry will be populated in the Connections table:

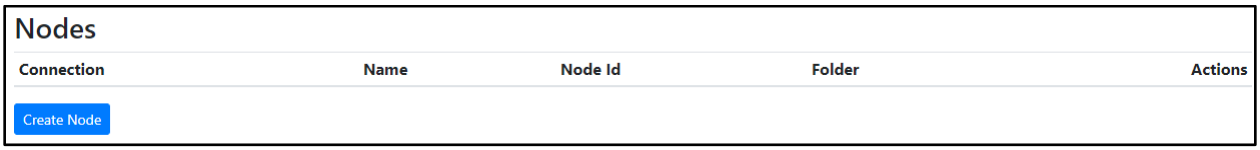
© 2021 CHIPKIN AUTOMATION SYSTEMS



***Note*:** Only one server connection can exist. If multiple connections are created, only the first one will be used.

3.3 Create Node

Follow the instructions below to add objects to the OPC UA Server.



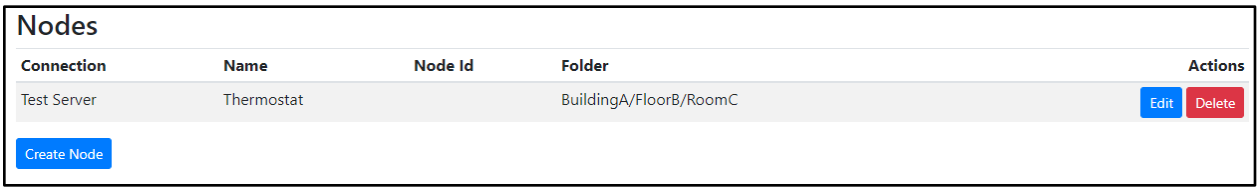
1. Click on the “Create Node” button to open the Create Node form.
2. Fill out the fields in the form. The fields are as follows:

Column Title	**Function**	**Legal Values**
Name	The name of the object	Text, must be unique
Connection	The name of the server to add this object.	Text (Use the name of the Connection created in the previous section)
NodeId	The nodeId to assign to this object. If left blank, the driver will generate one. See Appendix A.4 for description of a nodeId.	“ns=1;i=1001”, “ns=2;s=example_object”
Folder	The folder path where this object exists. This is primarily for organizing the objects. If left blank, the object will be placed in the root object folder of the OPC UA server.	Path Examples: “BuildingA/FloorB/RoomC” “MachineRoom1/CabinetX”

3. Click on the “Save” button to add the node.

   If successful, the new entry will be populated in the Nodes table:

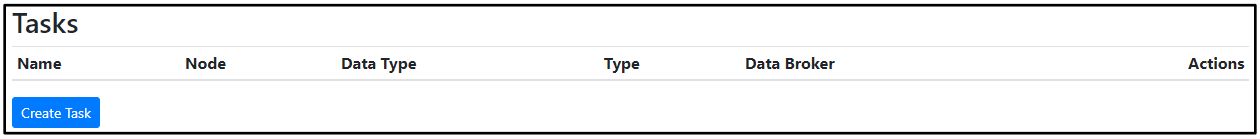
© 2021 CHIPKIN AUTOMATION SYSTEMS



Repeat the above steps to add additional objects.

3.4 Create Task

Create tasks to add variables to configured objects.



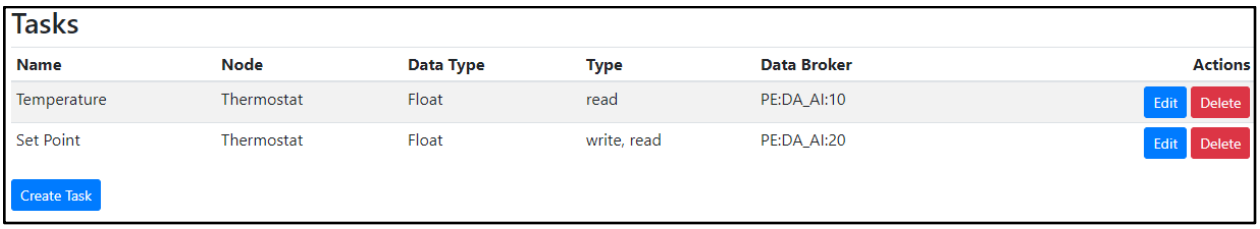
1. Click on the “Create Task” button to open the Create Task form.
2. Fill out the fields in the form. The fields are as follows:

Column Title	Function	Legal Values
Name	The name of the variable to add.	Text, must be unique
Node	The node that this variable belongs to.	Text (Use the name of a node created in the previous section)
DataBroker: Name	The data array in the protocol engine to retrieve the value.	One of the Data Array names
DataBroker: Start	The starting offset in the array to retrieve the value	0 to (“Data_Array_length” - 1)
DataType	The type of data that this data point represents	Refer to Appendix A.2
Type	Select whether this data point is read, read/write, or write only.	Read, Read/Write, Write
Desc	A description of the data point	Text

3. Click the “Save” button to add the task.

© 2021 CHIPKIN AUTOMATION SYSTEMS

If successful, the new entry will be populated in the Tasks table:



Repeat the above steps to add additional variables.

3.5 Saving the Server Configuration

When the configuration is complete, click on the "Save Configuration" button to save all of the updates and changes. For the configuration to take effect, reboot the system.

3.6 Resetting the Server Configuration

To clear the configuration and start over, click the "Reset Configuration" button. Then follow the instructions in the sections above to create new connections, nodes, and tasks.

3.7 OPC UA Client Configuration

To configure the FieldServer as an OPC UA Client, follow the instructions below to add a Connection (how to connect to OPC UA Servers), Nodes (OPC UA Client session information), and finally Tasks (data points on the OPC UA Server to read).

3.7.1 Create Connection

Add information on how to connect to one or more OPC UA Servers.

© 2021 CHIPKIN AUTOMATION SYSTEMS

1. Click on the “Create Connection” button to open the Create Connection form.
2. Fill out the fields in the form. The fields are as follows:

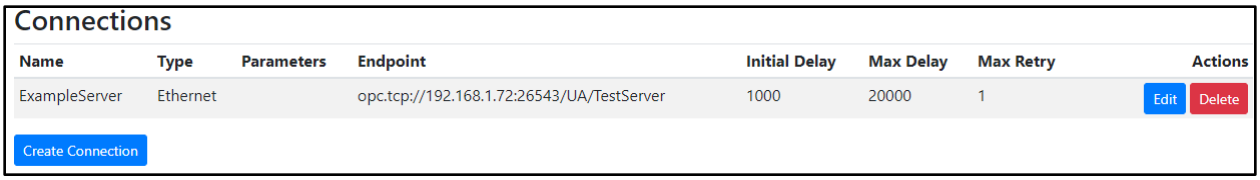
Column Title	**Function**	**Legal Values**
Name	The name of the server to connect to, used internally as an identifier	Text, must be unique
Type	The type of connection	**ethernet**
Parameters: Port	The physical port on the FieldServer to use	**n1**
Endpoint	The OPC UA Server endpoint to connect to.	opc.tcp://{hostname or ip address}:{port}/{referencePath} Example: opc.tcp://192.168.1.72:26543/UA/TestServer
InitialDelay	Delay in milliseconds before the first attempt to reconnect is made after initial connection failure	0-65535, **1000**
MaxDelay	Further reconnection attempts will have an increased delay. This parameter defines the maximum delay between two reconnections	0-65535, **20000**
MaxRetry	The maximum number of unsuccessful consecutive retries before the connection fails	0-20, **1**

* Bolded values are defaults

3. Click the “Save” button to add the connection.

If successful, the new entry will be populated in the Connections table:

© 2021 CHIPKIN AUTOMATION SYSTEMS



Repeat the above steps to add other servers to poll for data.

3.7.2 Create Node

For each connection, add a node that contains information on how to create an OPC UA Client Session.

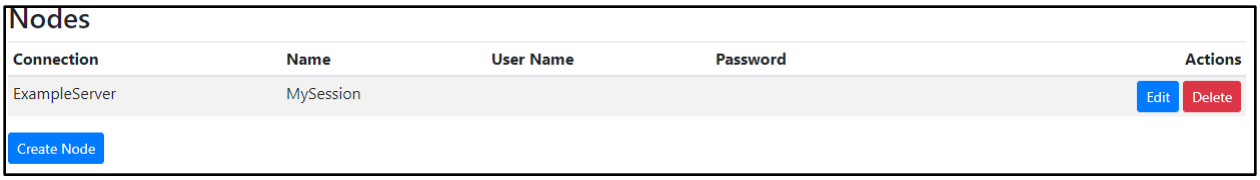
1. Click on the “Create Node” button to open the Create Node form.
2. Fill out the fields in the form. The fields are as follows

Column Title	Function	Legal Values
Name	The name of the session, used internally as an identifier	Text, must be unique
Connection	The connection that this session uses	Text (Use the name of a Connection created in the previous section)
UserName	If the session requires a user login, this is the username to use when creating a session. Can be blank if the OPC UA Server endpoint supports anonymous connections	Text
Password	If the session requires a user login, this is the password to use when creating a session. Can be blank if the OPC UA Server endpoint supports anonymous connections	Text

3. Click the “Save” button to add the node.

If successful, the new entry will be populated in the Nodes table:

© 2021 CHIPKIN AUTOMATION SYSTEMS



Repeat the above steps for each connection that was created.

3.7.3 Create Task

Add tasks to poll for specific data points.

1. Click on the “Create Task” button to open the Create Task form.
2. Fill out the fields in the form. The fields are as follows:

Column Title	**Function**	**Legal Values**
Name	The name of the data point	Text, must be unique
Node	The node that represents the session to use to poll for this data point	Text (Use the name of a Node configured in the previous section)
DataBroker: Name	The data array in the protocol engine to store the value.	One of the Data Array names
DataBroker: Start	The starting offset in the array to store the value	0 to (“Data_Array_length” - 1)
Type	The type of task which defines how the data point is polled	**read**
ScanInterval	How often in seconds to poll for data	0 - (Max UINT32), **30**
Datatype	The OPCUA Datatype of the data point	Null, Boolean, SByte, Byte, Int16, UInt16, Int32, UInt32, Int64*, UInt64*, **Float**, Double

© 2021 CHIPKIN AUTOMATION SYSTEMS

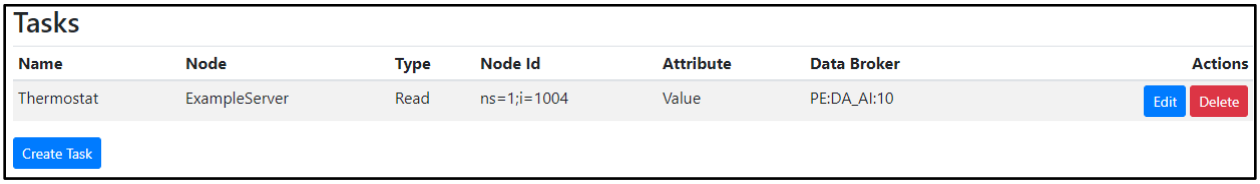
NodeId	The nodeId of the data point to poll	“ns=1;i=1001”, “ns=2;s=example_object” See Appendix A.4
Attribute	The attribute of the node to poll	**Value**, See Appendix A.3 for list of attributes

* Bolded values are defaults

Note: Int64 and UInt64 datatypes do not support write-through functionality.

1. Click the “Save” button to add the task.

   If successful, the new entry will be populated in the Tasks table:



   Repeat the above steps to add additional data points to poll.

3.7.4 Saving the Client Configuration

When the configuration is complete, click on the “Save Configuration” button to save all of the updates and changes. For the configuration to take effect, reboot the system.

3.7.5 Resetting the Client Configuration

To clear the configuration and start over, click the “Reset Configuration” button. Then follow the instructions in the sections above to create new connections, nodes, and tasks.

© 2021 CHIPKIN AUTOMATION SYSTEMS

4 OPC UA Test tools

A list of OPC UA testing tools that you can use to test the functionality of your system.

4.1 UaExpert

A Full-Featured OPC UA Client

https://www.unified-automation.com/products/development-tools/uaexpert.html

Features

- OPC UA Data Access View
- OPC UA Alarms & Conditions View
- OPC UA Historical Trend View
- Server Diagnostics View
- Simple Data logger CSV Plugin
- OPC UA Performance Plugin
- GDS Push-Model Plugin

4.2 OPC Reference Client

Official reference client used by the OPC Foundation for certification.

https://github.com/OPCFoundation/UA-.NETStandard

https://github.com/OPCFoundation/UA-.NETStandard/tree/master/Applications/ReferenceClient

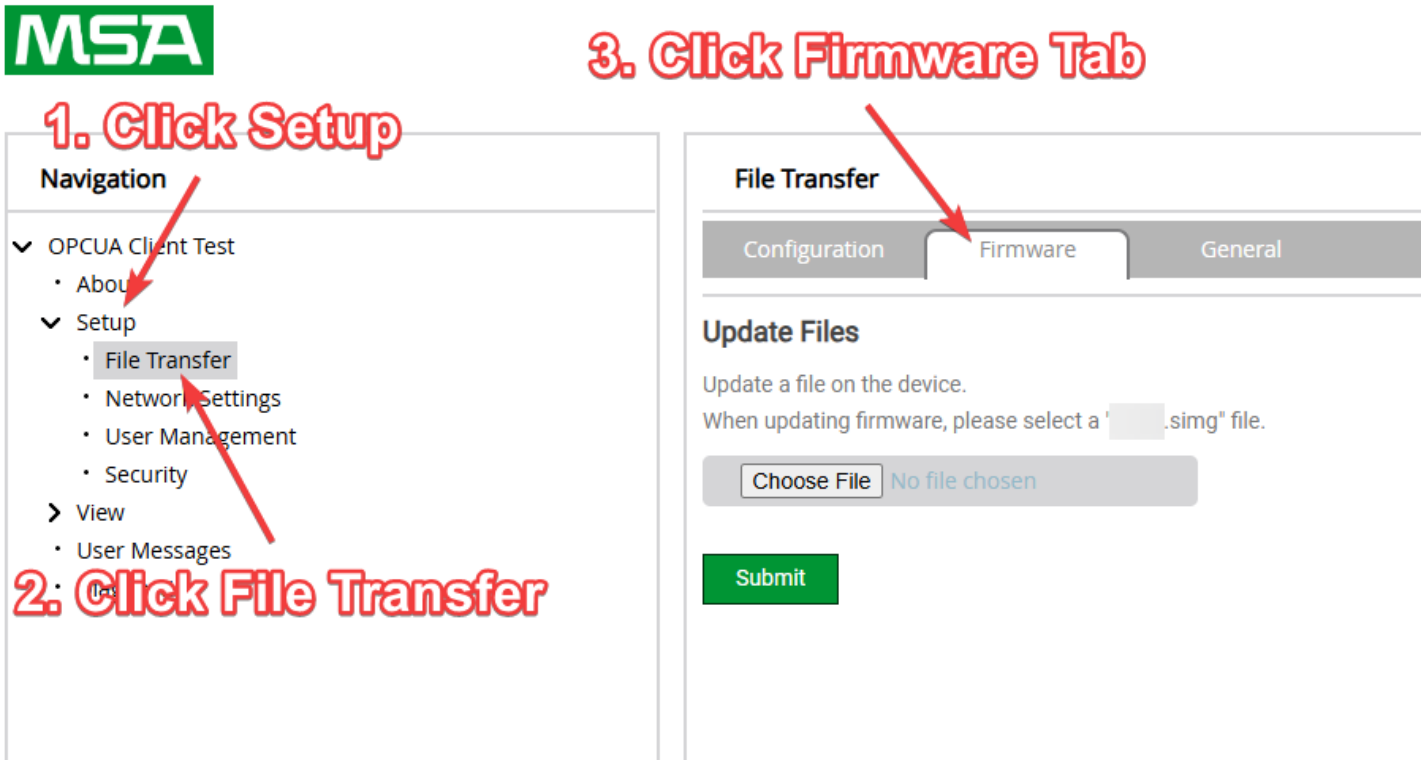
© 2021 CHIPKIN AUTOMATION SYSTEMS

5 Firmware

5.1 Uploading the OPC UA Firmware

To upload OPCUA Firmware to a FieldServer, open a web browser and navigate to the FSGUI page by typing in the following url: http://{IP Address}/htm/fsgui.htm where {IP Address} is the IP Address of the FieldServer.

On the FSGUI page click on the "Setup" menu, then "File Transfer" and finally the "Firmware" Tab.



Click the "Choose File" to select the file to import. The file will have the following format:

- fs-ae-chipkin-opcua-Chipkin-B0022-1.0.2-beta-armv7.simg or .pkg

Click the "Submit" button and the file upload will begin.

When prompted, Reboot the FieldServer by clicking the "System Reboot" button:

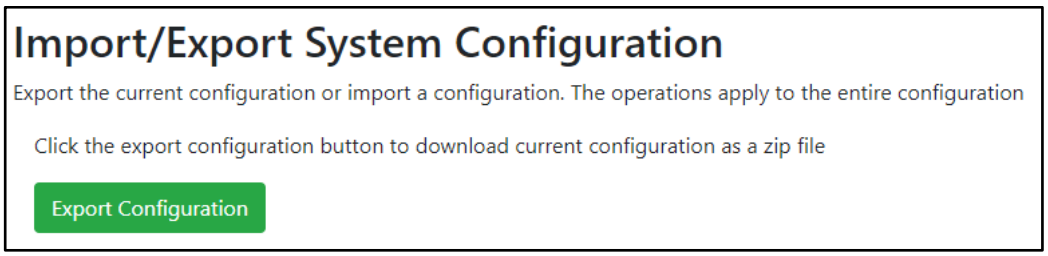
© 2021 CHIPKIN AUTOMATION SYSTEMS

6 Importing and Exporting Configurations

It is possible to export the current configuration to back it up or simply to make some edits. Users can also import either the entire configuration via a zip file or a PE (Protocol Engine) configuration.

6.1 How to Export the Configuration

1. Goto the system configuration page http://{IP_ADDRESS}/chipkinConfiguration/ui/
2. Click the Export Configuration button.



6.2 How to Import the Configuration

The file to import the configuration must be a zip file. The zip file should contain the following folders:

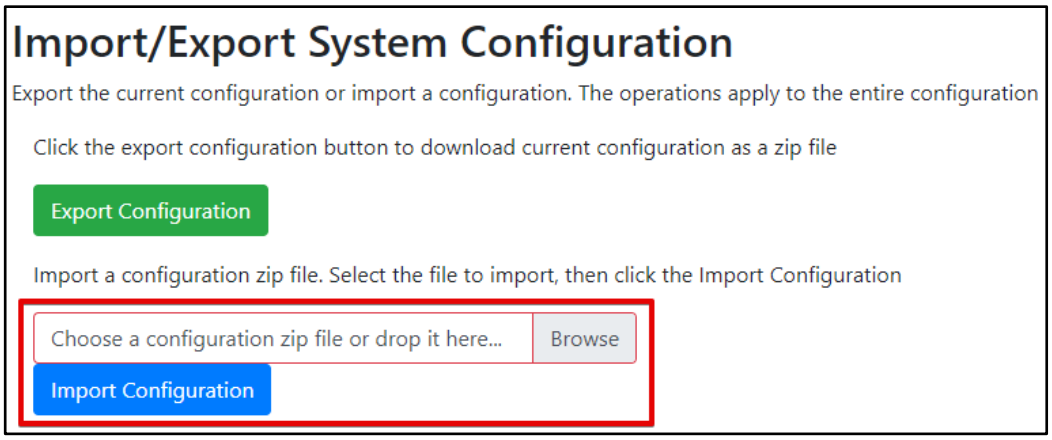
- ae - this folder contains any configuration files for the ae configuration
- documents - this folder contains any driver specific documents. For example, license product keys, etc.
- pe - this folder contains one config.csv file for the pe configuration.

To make sure the folder directory is correct, do an Export first, then extract the files, edit them, then zip them up again.

To import the configuration:

1. Goto the system configuration page http://{IP_ADDRESS}/chipkinConfiguration/ui/
2. Click the “Browse” button in the “Import/Export System Configuration” section and select the zip file containing the configuration to import.
3. Click the “Import Configuration” button and wait for the configuration to finish importing.
4. If successful, a success message will appear prompting a reboot of the Fieldserver for the changes to take effect.

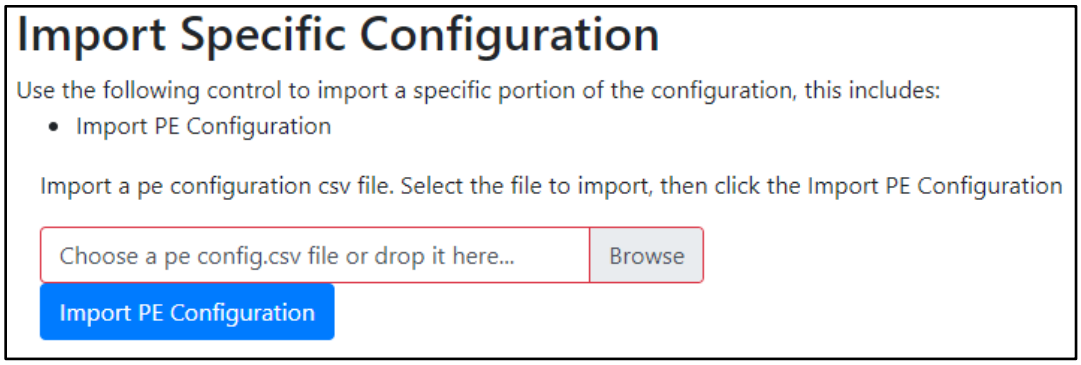
© 2021 CHIPKIN AUTOMATION SYSTEMS



6.3 How to Import a PE Configuration

It is possible to import a PE (Protocol Engine) configuration separately. To import a PE configuration:

1. Goto the system configuration page http://{IP_ADDRESS}/chipkinConfiguration/ui/
2. Click the “Browse” button in the “Import Specific Configuration” section and select the csv file containing the pe configuration to import.
3. Click the “Import PE Configuration” button and wait for the configuration to finish importing.
4. If successful, a success message will appear prompting a reboot of the Fieldserver for the changes to take effect.



© 2021 CHIPKIN AUTOMATION SYSTEMS

7 Appendix A - OPC UA Communication Functionality

7.1 Appendix A.1 - Supported Functionality

Client:

- Connect (using opc.tcp or https)
- CreateSession (with or without username and password)
- Read
- Write*

Note: Writes are available as Write-throughs. Please contact Chipkin for more information.

Server:

- Read
- Write

Note: The OPC UA Server does not support security certificates at this time**.**

7.2 Appendix A.2 - Supported Server Datatypes

- Null
- Boolean
- SByte
- Byte
- Int16
- UInt16
- Int32
- UInt32
- Int64
- UInt64
- Float
- Double
- String
- DateTime
- Guid
- ByteString

7.3 Appendix A.3 – Support Client Datatypes

- Null
- Boolean
- SByte
- Byte
- Int16
- UInt16
- Int32
- UInt32
- Int64*
- UInt64*
- Float
- Double

Note1: For OPCUA Client configurations, Int64 and UInt64 are read-only and do not support write-throughs.

© 2020 Chipkin Automation Systems

Note2: For OPCUA Client configurations, String, DateTime, Guid, and ByteString datatypes are not currently supported.

7.4 Appendix A.3 - Supported Client Attributes to Read

The following is a list of attributes that can be read from an OPC UA object. The most common one will be the 'Value' attribute, but the OPC UA driver supports reading from the other attributes as well.

Note: The values of these attributes must be one of the supported client datatypes listed above.

- NodeId
- NodeClass
- BrowseName
- DisplayName
- Description
- WriteMask
- UserWriteMask
- IsAbstract
- Symmetric
- InverseName
- ContainsNoLoops
- EventNotifier
- Value
- DataType
- ValueRank
- ArrayDimensions
- AccessLevel
- UserAccessLevel
- MinimumSamplingInterval
- Historizing
- Executable

7.5 Appendix A.4 - Description of NodeId

All data points and folders in OPCUA are identified with a unique NodeId.

In OPCUA, the NodeId exists in two parts:

- The namespace Index
- The Id, which can be one of the following types:
  - Numeric (a 32-bit integer)
  - String
  - GUID (globally unique identifier, 128 bits)
  - Byte String or Opaque (a binary data blob)

Example of NodeIds:

- Numeric: ns=2;i=10853
- String: ns=1;s=someIdentifier
- GUID: ns=1;g=BAEAF004-1E43-4A06-9EF0-E52010D5CD10
- Byte String: ns=2;b=00FF

© 2020 Chipkin Automation Systems

8 Appendix B – Troubleshooting

8.1 Appendix B.1 - Debugging an OPC UA connection

- If connections are not working properly, confirm the OPC UA Server Endpoint. Depending on how it has been configured, try using both the IP Address and the HostName. For example:
  - IP Address format: opc.tcp://192.168.1.72:26543/UA/TestServer
  - HostName format: opc.tcp://example-pc:26543/UA/TestServer

- Verify that the url contains the opc.tcp://. If it is missing, the connection will fail.

- Verify the resource path part of the endpoint. These are very specific and case sensitive. Some servers do not specify a resource path. See the following examples:
  - No resource path: opc.tcp://192.168.1.72:26543/
  - With resource path: opc.tcp://example-pc:26543/UA/
  - With longer path: opc.tcp://192.168.1.72:26543/UA/TestServer

- When using the FieldServer as an OPC UA Server, ensure that the FieldServer is on the same subnet as the OPC UA Client that is attempting to poll for data.

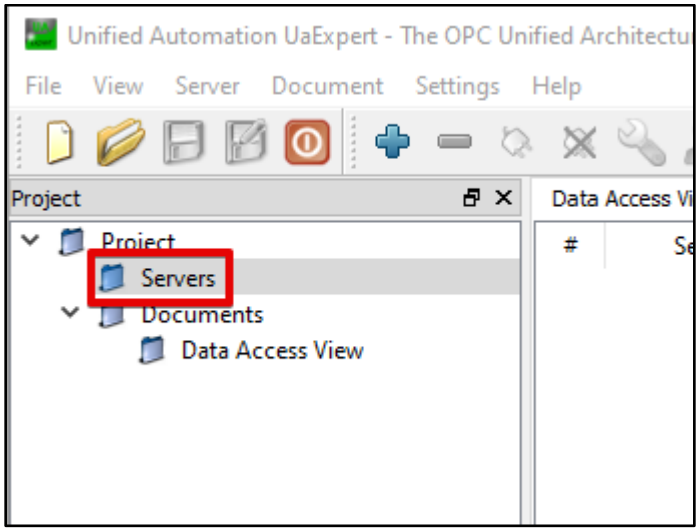
8.2 Appendix B.2 - Using UaExpert for Testing an OPC UA Server

Follow the steps in this section to setup the UaExpert tool to test a connection to an OPC UA Server (either the FieldServer configured as an OPC UA Server or to test an existing server to see what data is on it).

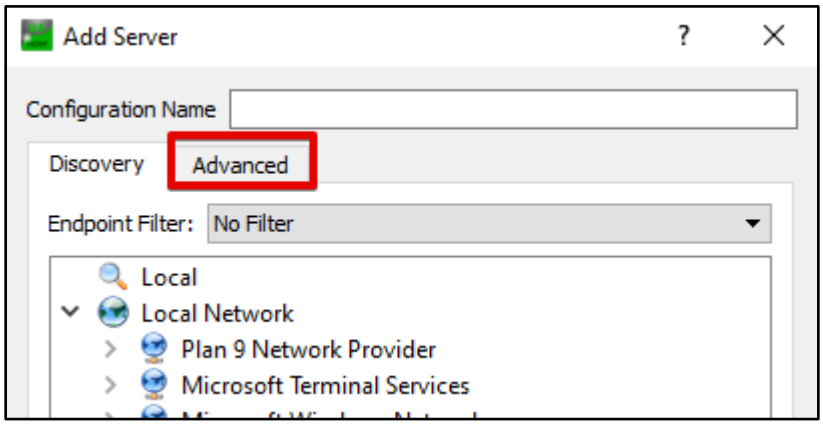
1. Download and install the tool: https://www.unified-automation.com/products/development-tools/uaexpert.html

2. In the tool, right-click on the Servers folder and click the Add option.

© 2020 Chipkin Automation Systems

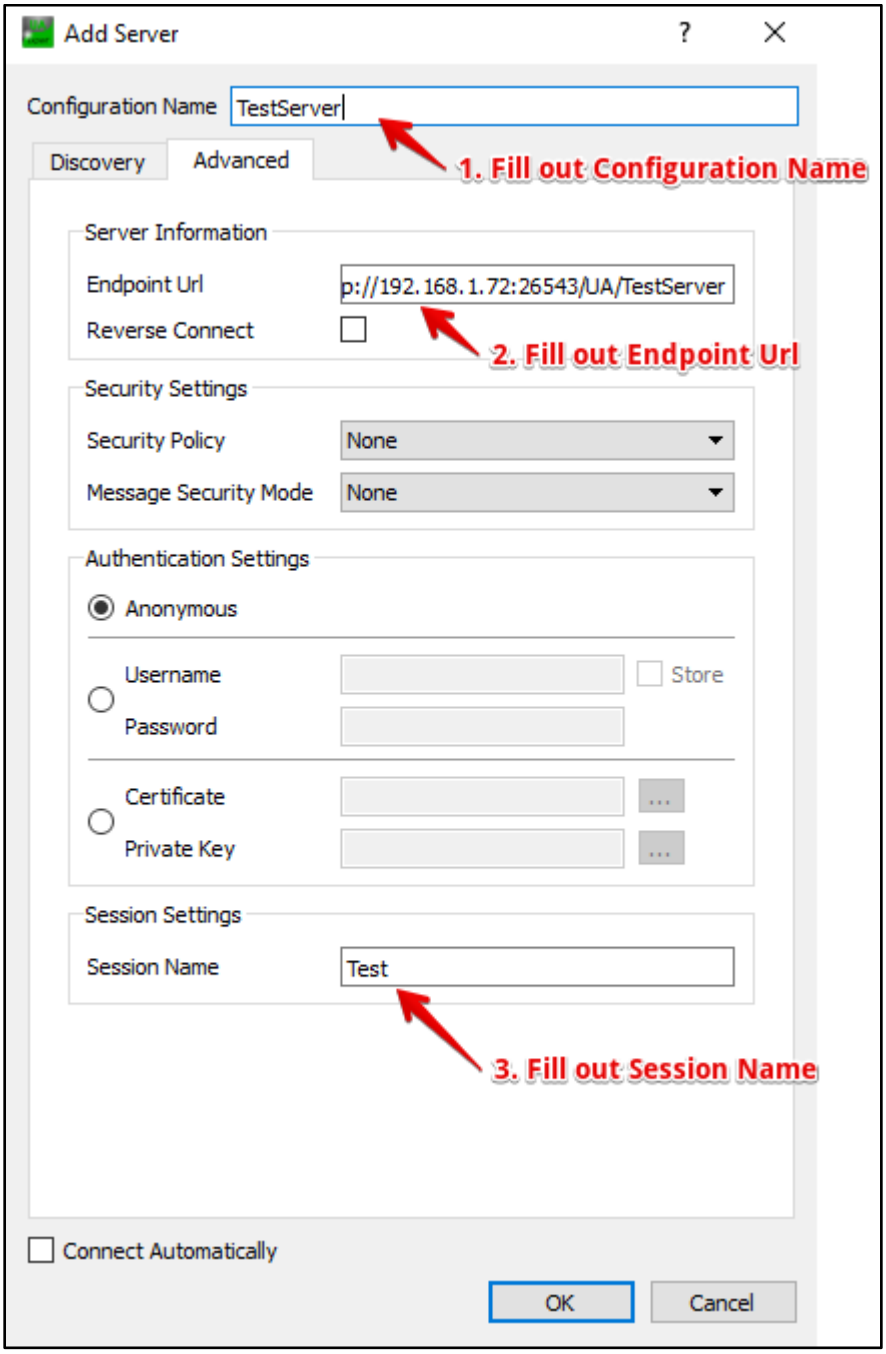


3. In the Add Server dialog box, click the “Advanced” tab



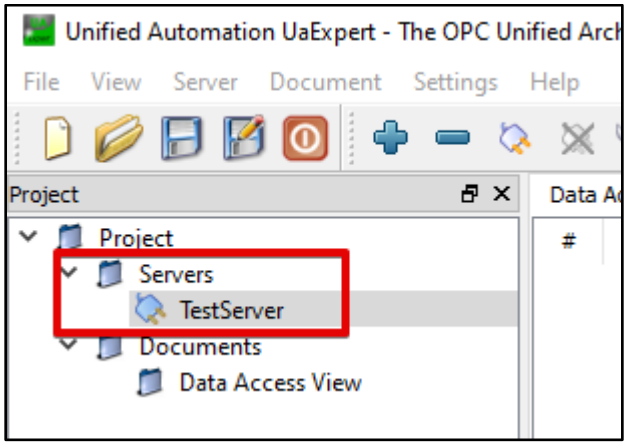
4. Fill out the following fields:
    a. Fill out the Configuration Name. This is a generic name to save this configuration
    b. Fill out the Endpoint Url. This should match the OPC UA Server endpoint.
    c. Fill out the Session Name. This is a generic name for the session.

© 2020 Chipkin Automation Systems

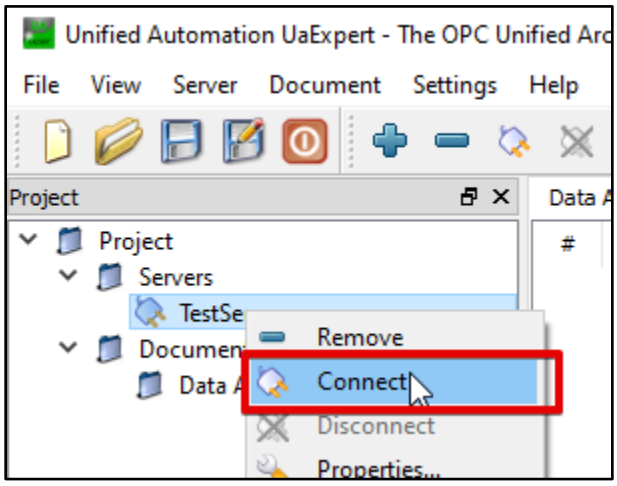


5. Click the “Ok” button to add it to the Server list

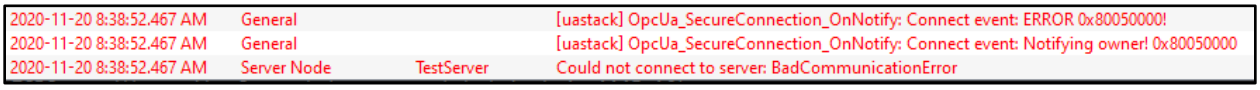
© 2020 Chipkin Automation Systems



6. Right-click the newly added server and click connect

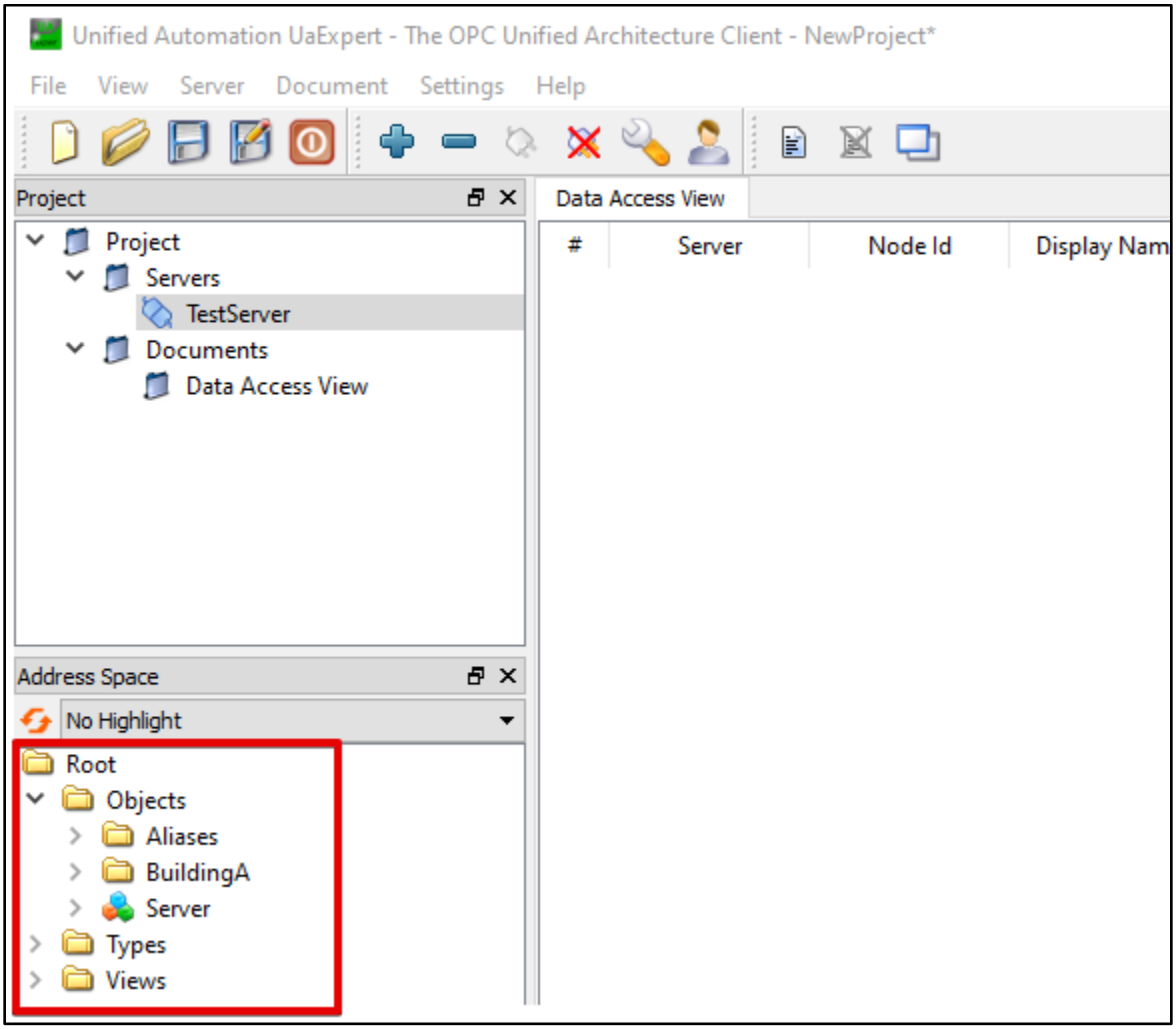


7. If there is an error, it will be displayed in the Log at the bottom of the screen. Here is an example of a BadCommunicationError:

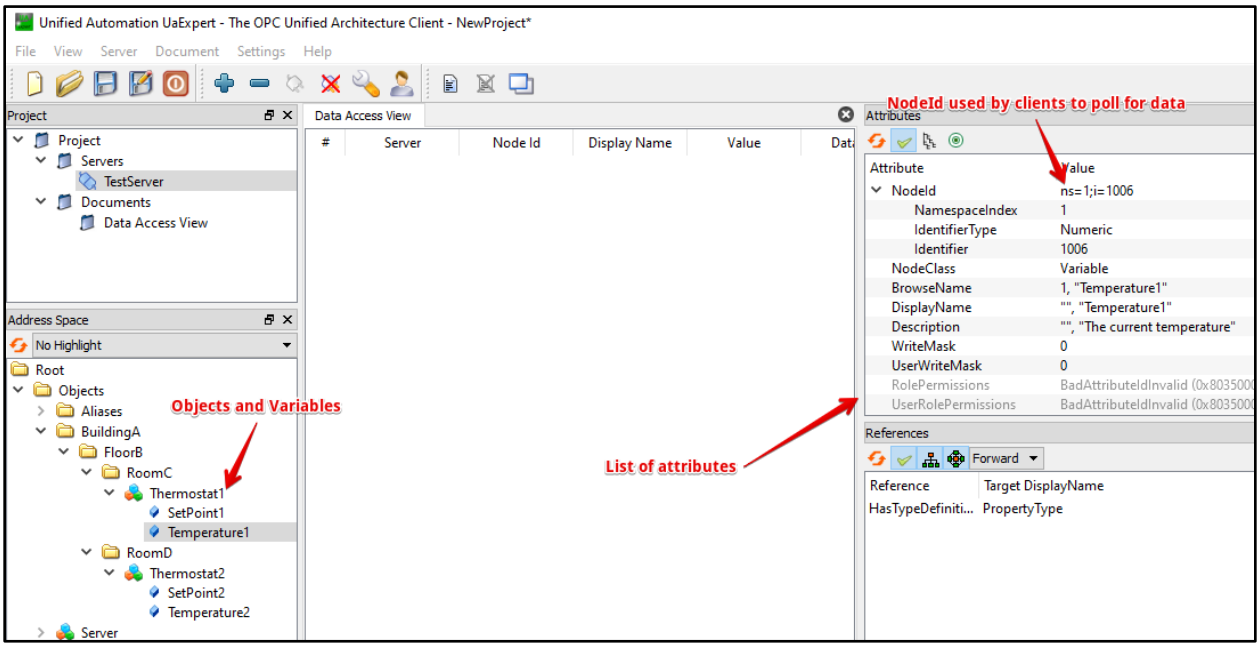


8. If successful, the OPC UA Server object tree will be populated in the Address Space section. You may see a Certificate Validation dialog box appear. Review the information and Accept the server certificate temporarily for this session.

© 2020 Chipkin Automation Systems



9. Open up any folders to see the objects that are available. You can see all the properties of the objects and attributes in the far right screens labelled Attributes and References.



© 2020 Chipkin Automation Systems

10. Drag and drop variables to the Data Access View screen in the middle to check data values.

Data Access View

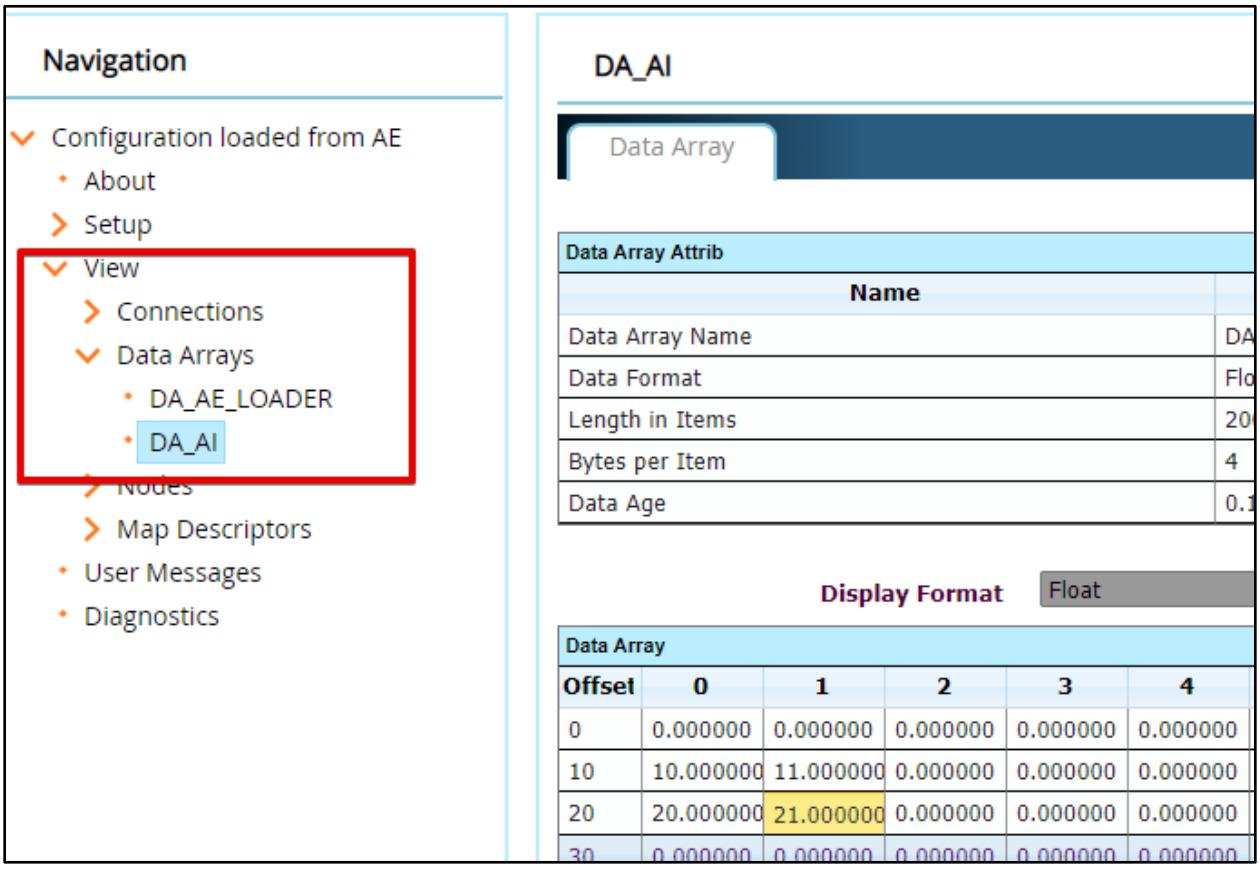
#	Server	Node Id	Display Name	Value	Datatype	Source Timestamp	Server Timestamp	Statuscode
1	TestServer	NS1|Numeric|1006	Temperature1	10	Float	8:42:16.570 AM	8:46:55.507 AM	Good
2	TestServer	NS1|Numeric|1007	SetPoint1	11	Float	8:42:18.100 AM	8:46:56.821 AM	Good
3	TestServer	NS1|Numeric|1008	Temperature2	20	Float	8:42:19.956 AM	8:46:57.972 AM	Good
4	TestServer	NS1|Numeric|1009	SetPoint2	21	Float	8:42:21.635 AM	8:46:58.843 AM	Good

8.3 Appendix B.3 - Testing Fieldserver as an OPC UA Server

The following instructions are how to confirm that a FieldServer that has been configured as an OPC UA Server is working correctly.

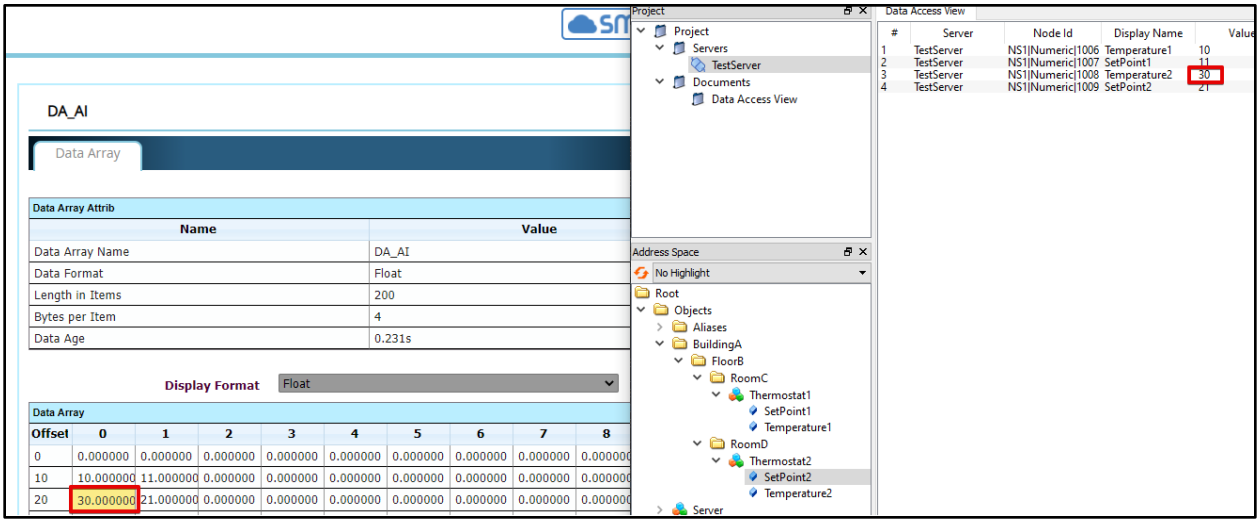
1. Follow the instructions in Appendix B.2 to use UaExpert to connect to the FieldServer configured as an OPC UA Server.

2. Access the Data Array page on the FieldServer interface

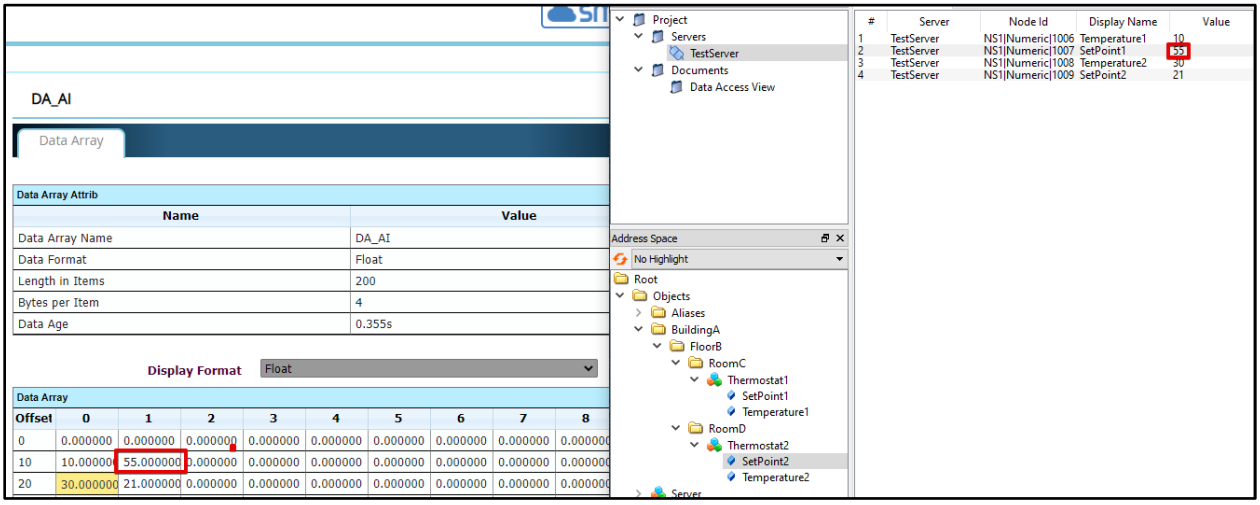


3. Change Values in the Data array and see the values updated in the UaExpert

© 2020 Chipkin Automation Systems



4. Change a value for a read/write or write only point in the UaExpert and see the value updated in the Data Array interface



© 2020 Chipkin Automation Systems

9 Appendix C - Example Configurations

```
{
  "ae": {
    "OPCUAServer": {
      "connections": [
        {
          "type": "ethernet",
          "name": "TestServer",
          "port": 26543,
          "hostname": "192.168.2.168",
          "resourcePath": "/UA/TestServer",
          "manufacturerName": "Example Company",
          "productName": "Example Product"
        }
      ],
      "nodes": [
        {
          "name": "Thermostat1",
          "connection": "TestServer",
          "folder": "BuildingA/FloorB/RoomC"
        },
        {
          "name": "Thermostat2",
          "connection": "TestServer",
          "folder": "BuildingA/FloorB/RoomD"
        }
      ],
      "tasks": [
        {
          "dataType": "Float",
          "type": ["read"],
```


© 2020 Chipkin Automation Systems

```
        "name": "Temperature1",
        "node": "Thermostat1",
        "dataBroker": { "pe": { "Name": "DA_AI", "Start": "10" } },
        "desc": "The current temperature"
      },
      {
        "dataType": "Float",
        "type": ["read", "write"],
        "name": "SetPoint1",
        "node": "Thermostat1",
        "dataBroker": { "pe": { "Name": "DA_AI", "Start": "11" } },
        "desc": "Setpoint for temperature"
      },
      {
        "dataType": "Float",
        "type": ["read"],
        "name": "Temperature2",
        "node": "Thermostat2",
        "dataBroker": { "pe": { "Name": "DA_AI", "Start": "20" } },
        "desc": "The current temperature"
      },
      {
        "dataType": "Float",
        "type": ["read", "write"],
        "name": "SetPoint2",
        "node": "Thermostat2",
        "dataBroker": { "pe": { "Name": "DA_AI", "Start": "21" } },
        "desc": "Setpoint for temperature"
      }
    ]
  }
```


© 2020 Chipkin Automation Systems

```
  }
}
```

9.1 Appendix C.2 - Example Client Configuration

```
{
  "ae": {
    "OPCUAClient": {
      "connections": [
        {
          "type": "ethernet",
          "name": "OPCUA_Test_Server",
          "parameters": {
            "port": "n1"
          },
          "endpoint": "opc.tcp://milo.digitalpetri.com:62541/milo",
          "maxDelay": 20000,
          "initialDelay": 1000,
          "maxRetry": 1,
          "taskDelay": 200,
          "maxTasksPerPoll": 50
        }
      ],
      "nodes": [
        {
          "connection": "OPCUA_Test_Server",
          "name": "Test_Session"
        }
      ],
      "tasks": [
        {
          "node": "Test_Session",
```


© 2020 Chipkin Automation Systems

```
      "type": "read",
      "attribute": "Value",
      "datatype": "Boolean",
      "name": "DemoDynamicBoolean",
      "dataBroker": { "pe": { "Name": "DA_BOOL", "Start": "0" } },
      "nodeId": "ns=2;s=Demo.Dynamic.Boolean",
      "scanInterval": "10"
    },
    {
      "node": "Test_Session",
      "type": "read",
      "attribute": "Value",
      "datatype": "Byte",
      "name": "DemoDynamicByte",
      "dataBroker": { "pe": { "Name": "DA_UINT", "Start": "0" } },
      "nodeId": "ns=2;s=Demo.Dynamic.Byte",
      "scanInterval": "10"
    },
    {
      "node": "Test_Session",
      "type": "read",
      "attribute": "Value",
      "datatype": "Float",
      "name": "DemoDynamicDataValue",
      "dataBroker": { "pe": { "Name": "DA_FLOAT", "Start": "0" } },
      "nodeId": "ns=2;s=Demo.Dynamic.DataValue",
      "scanInterval": "10"
    },
    {
      "node": "Test_Session",
      "type": "read",
```


© 2020 Chipkin Automation Systems

```
      "attribute": "Value",
      "datatype": "Double",
      "name": "DemoDynamicDouble",
      "dataBroker": { "pe": { "Name": "DA_FLOAT", "Start": "1" } },
      "nodeId": "ns=2;s=Demo.Dynamic.Double",
      "scanInterval": "10"
    },
    {
      "node": "Test_Session",
      "type": "read",
      "attribute": "Value",
      "datatype": "Float",
      "name": "DemoDynamicFloat",
      "dataBroker": { "pe": { "Name": "DA_FLOAT", "Start": "2" } },
      "nodeId": "ns=2;s=Demo.Dynamic.Float",
      "scanInterval": "10"
    },
    {
      "node": "Test_Session",
      "type": "read",
      "attribute": "Value",
      "datatype": "Int16",
      "name": "DemoDynamicInt16",
      "dataBroker": { "pe": { "Name": "DA_INT", "Start": "0" } },
      "nodeId": "ns=2;s=Demo.Dynamic.Int16",
      "scanInterval": "10"
    },
    {
      "node": "Test_Session",
      "type": "read",
      "attribute": "Value",
```


© 2020 Chipkin Automation Systems

```
      "datatype": "Int32",
      "name": "DemoDynamicInt32",
      "dataBroker": { "pe": { "Name": "DA_INT", "Start": "1" } },
      "nodeId": "ns=2;s=Demo.Dynamic.Int32",
      "scanInterval": "10"
    },
    {
      "node": "Test_Session",
      "type": "read",
      "attribute": "Value",
      "datatype": "Int64",
      "name": "DemoDynamicInt64",
      "dataBroker": { "pe": { "Name": "DA_INT", "Start": "2" } },
      "nodeId": "ns=2;s=Demo.Dynamic.Int64",
      "scanInterval": "10"
    },
    {
      "node": "Test_Session",
      "type": "read",
      "attribute": "Value",
      "datatype": "SByte",
      "name": "DemoDynamicSByte",
      "dataBroker": { "pe": { "Name": "DA_INT", "Start": "5" } },
      "nodeId": "ns=2;s=Demo.Dynamic.SByte",
      "scanInterval": "10"
    },
    {
      "node": "Test_Session",
      "type": "read",
      "attribute": "Value",
      "datatype": "UInt16",
```


© 2020 Chipkin Automation Systems

```
          "name": "DemoDynamicUInt16",
          "dataBroker": { "pe": { "Name": "DA_UINT", "Start": "1" } },
          "nodeId": "ns=2;s=Demo.Dynamic.UInt16",
          "scanInterval": "10"
        },
        {
          "node": "Test_Session",
          "type": "read",
          "attribute": "Value",
          "datatype": "UInt32",
          "name": "DemoDynamicUInt32",
          "dataBroker": { "pe": { "Name": "DA_UINT", "Start": "2" } },
          "nodeId": "ns=2;s=Demo.Dynamic.UInt32",
          "scanInterval": "10"
        },
        {
          "node": "Test_Session",
          "type": "read",
          "attribute": "Value",
          "datatype": "UInt64",
          "name": "DemoDynamicUInt64",
          "dataBroker": { "pe": { "Name": "DA_UINT", "Start": "3" } },
          "nodeId": "ns=2;s=Demo.Dynamic.UInt64",
          "scanInterval": "10"
        }
      ]
    }
  }
}
```


© 2020 Chipkin Automation Systems

10 Appendix D – Updating Client Configuration for Writes

OPCUA Client Write functionality was added in version 1.0.2 of the OPCUA Driver.

This functionality allows downstream devices (example: BACnet IP) to change values that will then be pushed to the OPCUA Server via the FieldServer.

As such, OPCUA Client configurations from before version 1.0.2 will need to be updated to include the updated "datatype" field in the tasks.

Valid values for datatype are:

- Null
- Boolean
- SByte
- Byte
- Int16
- UInt16
- Int32
- UInt32
- Int64*
- UInt64*
- Float
- Double

Note: For OPCUA Client configurations, Int64 and UInt64 data types do not support writes.

1) For configurations where tasks do not have a datatype field

Example – Prior to version 1.0.2:

```
{
  "ae": {
    "OPCUAClient": {
      "connections": [
        {
          "type": "ethernet",
          "name": "QuickReferenceServer",
          "endpoint": "opc.tcp://localhost:62541/Quickstarts/ReferenceServer",
          "initialDelay": 1000,
          "maxDelay": 2000,
          "maxRetry": 1
        }
      ],
```


© 2020 Chipkin Automation Systems

```
      "nodes": [{ "name": "MySession", "connection": "QuickReferenceServer" }],
      "tasks": [
        {
          "type": "read",
          "scanInterval": 30,
          "attribute": "Value",
          "name": "ScalarStaticFloat",
          "node": "MySession",
          "dataBroker": { "pe": { "Name": "DA_AI", "Start": "10" } },
          "nodeId": "ns=2;s=Scalar_Static_Float"
        },
        {
          "type": "read",
          "scanInterval": 30,
          "attribute": "Value",
          "name": "ScalarStaticInt16",
          "node": "MySession",
          "dataBroker": { "pe": { "Name": "DA_AI", "Start": "11" } },
          "nodeId": "ns=2;s=Scalar_Static_Int16"
        }
      ]
    }
  }
}
```

Add an OPCUA Datatype field for each task. Use the UAExpert tool to determine the datatype of each data point.

Example – Version 1.0.2 and after:

```
{
  "ae": {
    "OPCUAClient": {
      "connections": [
```

```
    {
      "type": "ethernet",
      "name": "QuickReferenceServer",
      "endpoint": "opc.tcp://localhost:62541/Quickstarts/ReferenceServer",
      "initialDelay": 1000,
      "maxDelay": 2000,
      "maxRetry": 1
    }
  ],
  "nodes": [{ "name": "MySession", "connection": "QuickReferenceServer" }],
  "tasks": [
    {
      "type": "read",
      "scanInterval": 30,
      "attribute": "Value",
      "name": "ScalarStaticFloat",
      "node": "MySession",
      "dataBroker": { "pe": { "Name": "DA_AI", "Start": "10" } },
      "nodeId": "ns=2;s=Scalar_Static_Float",
      "datatype": "Float"
    },
    {
      "type": "read",
      "scanInterval": 30,
      "attribute": "Value",
      "name": "ScalarStaticInt16",
      "node": "MySession",
      "dataBroker": { "pe": { "Name": "DA_AI", "Start": "11" } },
      "nodeId": "ns=2;s=Scalar_Static_Int16",
      "datatype": "Int16"
    }
```


© 2020 Chipkin Automation Systems

```
      ]
    }
  }
}
```

2) For configurations that have a datatype field with one of the following values: “boolean”, “number”, “64bit”:

Example – Prior to version 1.0.2:

```
{
  "ae": {
    "OPCUAClient": {
      "connections": [
        {
          "type": "ethernet",
          "name": "QuickReferenceServer",
          "endpoint": "opc.tcp://localhost:62541/Quickstarts/ReferenceServer",
          "initialDelay": 1000,
          "maxDelay": 2000,
          "maxRetry": 1
        }
      ],
      "nodes": [{ "name": "MySession", "connection": "QuickReferenceServer" }],
      "tasks": [
        {
          "type": "read",
          "scanInterval": 30,
          "attribute": "Value",
          "name": "ScalarStaticFloat",
          "node": "MySession",
          "dataBroker": { "pe": { "Name": "DA_AI", "Start": "10" } },
          "nodeId": "ns=2;s=Scalar_Static_Float",
          "datatype": "number"
```


© 2020 Chipkin Automation Systems

```
        },
        {
          "type": "read",
          "scanInterval": 30,
          "attribute": "Value",
          "name": "ScalarStaticBoolean",
          "node": "MySession",
          "dataBroker": { "pe": { "Name": "DA_AI", "Start": "11" } },
          "nodeId": "ns=2;s=Scalar_Static_Boolean",
          "datatype": "boolean"
        },
        {
          "type": "read",
          "scanInterval": 30,
          "attribute": "Value",
          "name": "ScalarStaticInt16",
          "node": "MySession",
          "dataBroker": { "pe": { "Name": "DA_AI", "Start": "12" } },
          "nodeId": "ns=2;s=Scalar_Static_Int16",
          "datatype": "number"
        }
      ]
    }
  }
}
```

Update the datatype field for each task using the specific OPCUA Datatype for that data point. Use the UAExpert tool to determine the datatype of each data point.

Example – Version 1.0.2 and after:

```
{
  "ae": {
```


© 2020 Chipkin Automation Systems

```
"OPCUAClient": {
  "connections": [
    {
      "type": "ethernet",
      "name": "QuickReferenceServer",
      "endpoint": "opc.tcp://localhost:62541/Quickstarts/ReferenceServer",
      "initialDelay": 1000,
      "maxDelay": 2000,
      "maxRetry": 1
    }
  ],
  "nodes": [{ "name": "MySession", "connection": "QuickReferenceServer" }],
  "tasks": [
    {
      "type": "read",
      "scanInterval": 30,
      "attribute": "Value",
      "name": "ScalarStaticFloat",
      "node": "MySession",
      "dataBroker": { "pe": { "Name": "DA_AI", "Start": "10" } },
      "nodeId": "ns=2;s=Scalar_Static_Float",
      "datatype": "Float"
    },
    {
      "type": "read",
      "scanInterval": 30,
      "attribute": "Value",
      "name": "ScalarStaticBoolean",
      "node": "MySession",
      "dataBroker": { "pe": { "Name": "DA_AI", "Start": "11" } },
      "nodeId": "ns=2;s=Scalar_Static_Boolean",
```


© 2020 Chipkin Automation Systems

```
          "datatype": "Boolean"
        },
        {
          "type": "read",
          "scanInterval": 30,
          "attribute": "Value",
          "name": "ScalarStaticInt16",
          "node": "MySession",
          "dataBroker": { "pe": { "Name": "DA_AI", "Start": "12" } },
          "nodeId": "ns=2;s=Scalar_Static_Int16",
          "datatype": "Int16"
        }
      ]
    }
  }
}
```


© 2020 Chipkin Automation Systems

11 Appendix E - Marketing

11.1 Appendix E.1 - Case study

A series of case studies for OPC can be found here

https://opcfoundation.org/resources/case-studies/

11.2 Appendix E.2 - Keywords

OPC, OPC UA, OPC-UA Server, OPC UA Client, OPC Unified Architecture, OPC technologies, Open Platform Communications, Industry 4.0

© 2020 Chipkin Automation Systems

12 Appendix F - Glossary of terms

- **SOA** - Service-oriented architecture
- **OPC** - Open Platform Communications. The acronym OPC was borne from OLE (object linking and embedding) for Process Control.
- **OPC-UA** - Unified Architecture
- **OPC-Classic** - The OPC standard that was restricted to the Windows operating system.
- **PubSub** - Publish - Subscribe. A mechanism for data and event notification
- **UACTT** - OPC UA Compliance Test Tool https://opcfoundation.org/developer-tools/certification-test-tools/opc-ua-compliance-test-tool-uactt/
- **ERP** - Enterprise Resource Planning
- **MES** - Manufacturing Execution Systems

© 2020 Chipkin Automation Systems

13 Revision History

This table summarizes the update history for this document. Please contact Chipkin for an updated version of this document if required.

Date	Agent	Revision	Comments
2020 Nov 17	SWS	1	Created initial document
2020 Nov 20	ACF	2	Added OPC UA Quickstart section.
2024 Jan 02	ACF	3	Fixed typo in Appendix B
2025 Apr 11	ACF	4	Updated Client configuration tasks to have datatype field Added information regarding Client writes Updated appendix of supported driver functionality Added instructions on how to update the configuration for all drivers that are below version 1.0.2 of the OPCUA Driver
2025 Jul 18	ACF	5	Removed License Key section, Updated Template

© 2020 Chipkin Automation Systems