

Ignition 8.1 User Manual — OPC UA and Device Drivers

Generated 2026-10-12 from
<https://www.docs.inductiveautomation.com/docs/8.1/>

OPC UA

OPC Specification

Open Platform Communications (OPC) is a technology standard used to exchange information between hardware and software through specialized drivers. Information is exchanged between items such as programmable logic controllers (PLCs), gauges, and databases. OPC is published and maintained by the OPC Foundation, an organization comprised of hundreds of member companies that strives to ensure interoperability on the plant floor and beyond.

The original OPC specifications used Microsoft Distributed Component Object Model (DCOM) technology to provide a uniform way for industrial applications to share data. There were several separate specifications that provided functions such as Data Access (OPC-DA), Alarms and Events (A&E), and Historical data (HDA).

DCOM always proved difficult to work with, and by 2004 it was clear that a more modern solution was needed. Therefore, a new specification was developed that used common networking principles (like TCP/IP) instead of DCOM, was platform independent, and combined the various separate specifications into one: Open Process Connectivity Unified Architecture (OPC UA).

OPC UA

OPC UA is the leading industrial standard for platform and vendor-neutral data access. Connecting any PLC device to Ignition is easy with OPC UA. Device connections are done over the Ethernet for those devices that have an Ignition device driver. The OPC UA Module makes Ignition act as an OPC UA server, serving data collected by its built in drivers to other Ignition modules, as well as to third-party OPC UA clients.

OPC UA is the latest revision of the OPC specification, which offers platform and vendor neutral transfer and use of industrial data. The specification plays a crucial role in Ignition, and is the primary data access specification used in the Gateway. Ignition supports connections to any number of OPC UA servers created by any manufacturer, provided that they are compliant to the specification. The data is then used to drive all aspects of the system. Creating connections to OPC UA servers is described below in [Connecting with OPC UA](#).

Distributed Systems with OPC UA

OPC UA breaks down boundaries and enables free data flow. Using standard TCP/IP instead of legacy DCOM, OPC UA makes it easy to securely transfer data between networks and through firewalls. All OPC UA connections are based on the same OPC, which means that a connection to your local machine is not entirely different than a connection to a machine that's far away. This enables the creation of a highly distributed system, and in combination with other features of Ignition can lead to more connected enterprises.

For example, imagine a corporate network with an office in the center, and remote processes connected through a VPN, which would pass through a variety of connections. Each remote site could have an Ignition installation running only an OPC UA module that would report data back to a central facility and record it in a database. The overall system cost would be very low. The data could be managed centrally in a single location, and then made available to all interested parties through the Vision module or any application that could access the database.

Servers and Clients

When discussing OPC (as the specifications are often called collectively), it is common to hear about OPC **servers** and OPC **clients**. An OPC server is a piece of software that implements the OPC interface and provides data. An OPC client is an application which connects to an OPC server and uses the specification to retrieve and work with data.

The Ignition platform inherently offers OPC UA client functionality. Even with no modules installed, the Gateway can connect to any compliant OPC UA server and work with data. With the addition of the OPC UA module, Ignition becomes an OPC server as well, hosting device drivers that read and publish data.

The **OPC COM module** is available to provide client access to older, DCOM based, OPC-DA servers.

Technology

The OPC UA specification offers a wide range of flexibility in choosing technologies, from the transport mechanism, to the way data is encoded, to the encryption used to secure the data. Ignition supports the UA/TCP transport with the UA/Binary encoding scheme for maximum performance.

Additionally, Ignition supports all of the common encryption schemes. This means that Ignition connects to OPC UA servers (and allows connections from clients) over TCP/IP using encryption, and sends data by first encoding it into an efficient format defined by the OPC UA specification. This is in

contrast to other schemes outlined in the specification, which can use web services and XML encoding that are not as efficient.

Connecting with OPC UA

Connecting to a Device

With the Ignition OPC UA module and device drivers installed, connecting to a device is simple. To quickly get connected to one of your devices, go to the **Config** tab of the Ignition Gateway and scroll down to **OPC UA > Device Connections**. The Device page will appear showing all of your installed devices.

To add a new device, click on **Create New Device....** Ignition has several device drivers available, and the Device Type list is populated based on what Driver modules you have installed. Select the type of device connection you want and fill in the details for your device connection. See the device types below for more information on connecting to your specific device type:

- **Allen-Bradley Ethernet**
 - **Connecting to CompactLogix**
 - **Connecting to ControlLogix**
 - **Connecting to Logix**
 - **Connecting to Micro800**
 - **Connecting to MicroLogix**
 - **Connecting to PLC5**
 - **Connecting to SLC**
 - **Allen-Bradley Connection Paths**
- **Modbus**
 - **Connecting to Modbus Device**
 - **Modbus Addressing**
 - **Modbus Address Mapping**
- **Siemens**
- **UDP and TCP Driver**
- **DNP3**
 - **DNP3 Driver**
 - **Legacy DNP3 Driver**
- **Omron NJ Driver**
- **Omron FINS Driver**

- [Programmable Device Simulator](#)
- [BACnet](#)
 - [BACnet Local Device](#)
 - [BACnet Remote Device](#)
- [IEC 61850 Driver](#)
- [Mitsubishi TCP Driver](#)

The screenshot shows the Ignition Gateway configuration interface. The top navigation bar includes the Ignition logo, a 'Help' link, and a 'Get Designer' button. The breadcrumb trail indicates the current path: **Config > Opcua > Devices**.

The left sidebar contains the following menu items:

- SYSTEM**
 - Overview
 - Backup/Restore
 - Licensing
 - Modules
 - Projects
 - Redundancy
 - Gateway Settings
- NETWORKING**
 - Gateway Network
 - Email Settings
- SECURITY**
 - Auditing
 - Users, Roles
 - Service Security
 - Identity Providers
 - Security Levels
 - Security Zones
- DATABASES**
 - Connections
 - Drivers
 - Store and Forward

The main content area displays a list of device drivers, each with a radio button and a description:

- ☒ **Allen-Bradley CompactLogix (Legacy)**
Connect to CompactLogix firmware v20 and prior processors.
- ☐ **Allen-Bradley ControlLogix (Legacy)**
Connect to ControlLogix firmware v20 and prior processors.
- ☐ **Allen-Bradley Logix Driver**
Connect to Allen-Bradley Logix family devices, including devices with firmware v21+.
- ☐ **Allen-Bradley MicroLogix**
Connect to MicroLogix 1100 and 1400 series PLCs.
- ☐ **Allen-Bradley PLC5**
Connect to PLC5s via Ethernet.
- ☐ **Allen-Bradley SLC**
Connect to SLC 5/05s via Ethernet.
- ☐ **DNP3 Driver**
Connect to a DNP3 outstation.

Connecting to a Server

If you don't see the type of device you want, then you can always connect with another OPC UA or OPC DA server.

- [Third Party OPC Servers](#)
- [OPC COM](#)

OPC Quick Client

After you connect to a device, you can access the **OPC Quick Client** page on the Gateway by following **Config > OPC Client > OPC Quick Client**. It allows for quick, simple testing of any devices

connected to the server. You can browse the OPC UA Server by expanding the tree nodes, and read from or write to OPC tags by clicking on the **[r]** and **[w]** links next to the tags.

Clicking on the **[s]** link will create a subscription and show live value changes in the Subscriptions area at the bottom of the screen. Use the **Set** button to confirm any changes made in the **Subscription name** and **Rate (ms)** fields.

Ignition-ignition8-ubuntu-64bit

admin | Sign Out

Ignition

Help

Get Designer

Home

Status

Config

SYSTEM

Overview

Backup/Restore

Licensing

Modules

Projects

Redundancy

Gateway Settings

NETWORKING

Gateway Network

Email Settings

SECURITY

Auditing

Users, Roles

Service Security

Identity Providers

Security Levels

Security Zones

DATABASES

Connections

Drivers

Store and Forward

ALARMING

General

Journal

Search...

Config > Opc > OPC Quick Client

TYPE	ACTION	TITLE
Server	refresh	Ignition OPC UA Server
Object		Server
Object		ServerConfiguration
Object		ServerCapabilities
Object		ServerDiagnostics
Object		VendorServerInfo
Object		ServerRedundancy
Object		Namespaces
Tag	[s][r][w]	ServerArray
Tag	[s][r][w]	NamespaceArray
Tag	[s][r][w]	ServerStatus
Tag	[s][r][w]	ServiceLevel
Tag	[s][r][w]	Auditing
Tag	[s][r][w]	EstimatedReturnTime

Subscription 1 [x] [Add]

Server	Address	Value	Quality	Timestamp
--------	---------	-------	---------	-----------

Subscription name :

Subscription 1

Rate (ms) :

1000

Set

How Do I Get Data from My PLC?

Getting data from your PLC into Ignition is a two step process:

1. Add a device, see [Connecting to a Device](#).
2. Add some tags, see [Creating Tags](#).

It requires you to touch both the Ignition Gateway and the Ignition Designer. There are also some limitations as to what kind of devices you can connect to Ignition and these are explained throughout

the user manual, however, included below is an overview of what you can expect when it comes to compatibility.

Brief Summary of Device Connection in Ignition

- Ignition can only connect directly to devices over Ethernet.
- Ignition can only connect directly to devices for which there is an Ignition device driver. Visit the [OPC UA Drivers](#) page for an overview of supported devices and drivers.
- Ignition can connect to third party OPC servers via OPC UA or OPC-DA (using the OPC-COM module) for devices that do not have a supported driver.

Adding a Device to Ignition

Ignition Supported OPC UA Device

Most commonly you will be adding a device that is supported by one of the built-in device drivers. The first step is connecting your device to Ignition. This is done through the Ignition Gateway Config tab under the OPC UA > Device Connections page.

1. Click **Create new Device....**
2. Select the driver for the device you wish to add. Click **Next**.
3. When adding a device you will notice that there are some common settings that are shared by all devices. You can find an explanation of these settings on the individual page for each [OPC UA Driver](#).
4. Specify any of the required device specific settings for the device (for example, hostname, etc.)
5. Check the status of your device to see if it is connected.

As long as all the device information you entered was correct you should see your device in a connected state. The only exception to this is if you chose to add a Siemens or Modbus device. Since these devices don't support the browsing of Tags, you will have to create and address some Tags in the Ignition Designer before the device will stop cycling from a connected to disconnected state.

If you need to address your Tags for your Siemens or Modbus device, you'll want to read about adding Tags in the Ignition Designer as well as how addressing works for the different protocols. You will have to first add a Tag in the Ignition Designer and then edit the OPC Item Path of the Tag using the appropriate addressing scheme.

Adding Connection to Third Party OPC Server Via OPC UA

If your device does not have an Ignition driver, you can use a third party OPC server to connect to your device and then have Ignition connect to the server as a client. If the OPC server offers support

for OPC UA, you can add a new OPC UA server connection in the Ignition Gateway. For more information, see the [OPC UA Client Connection Settings](#) page. Additionally, the [Connecting to Kepware OPC UA](#) is useful when connecting to Kepserver.

Adding Connection to Third Party OPC Server Via OPC COM

When attempting to connect Ignition to an OPC COM server, the OPC COM module must be installed. More information about configuring OPC COM connections can be found on the [OPC COM](#) page.

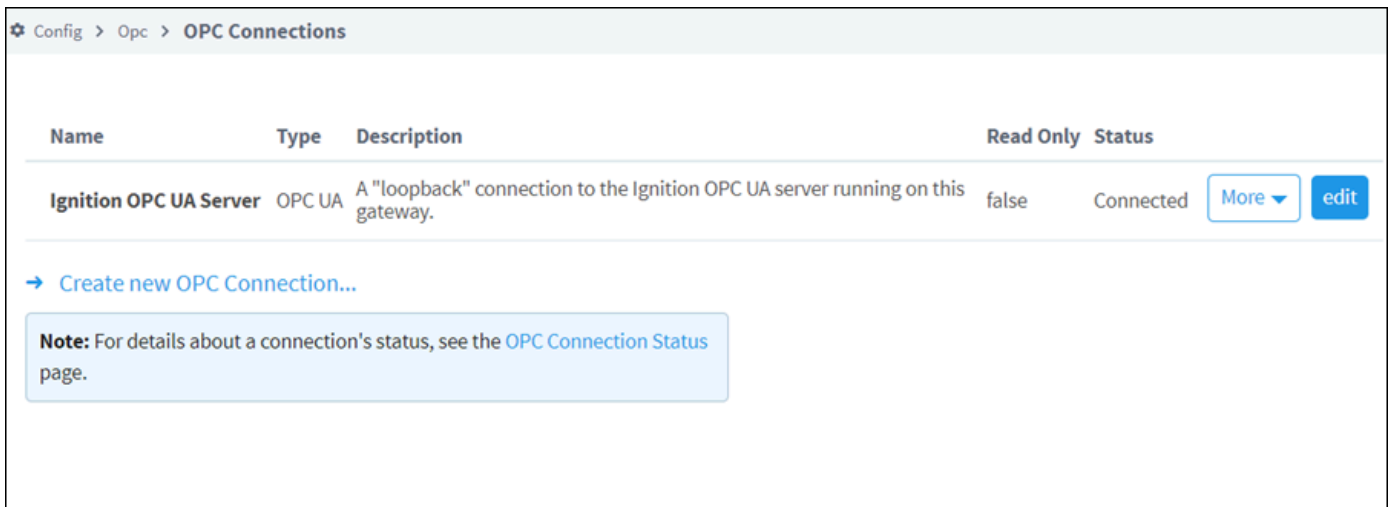
OPC UA Client Connection Settings

Configuring an OPC Client Connection

An OPC UA connection is used to communicate with an OPC UA compliant server, such as the one the [OPC UA Module](#) provides.

The following steps walk through connecting Ignition (as an OPC UA client) to a OPC UA server. For information on connecting to a third party server, refer to the [Connecting to Kepware OPC UA](#) page for an example.

1. On the Config tab of the Gateway Webpage, go to **OPC Client > OPC Connections**. The OPC Server Connections page is displayed.
2. Click on the **Create new OPC Connection**.



Config > Opc > OPC Connections

Name	Type	Description	Read Only	Status	
Ignition OPC UA Server	OPC UA	A "loopback" connection to the Ignition OPC UA server running on this gateway.	false	Connected	More edit

[→ Create new OPC Connection...](#)

Note: For details about a connection's status, see the [OPC Connection Status](#) page.

3. Select **OPC UA** from the list and click **Next**. The Server Discovery page appears.
4. Enter an OPC UA endpoint URL for the OPC UA server Ignition should connect to. The format should be as follows:

```
opc.tcp://IpAddress:Port
```

Alternatively, you can use a host name instead of an IP address:

opc.tcp://myServer:12345

NOTE

An Advanced Configuration link was added to this flow, allowing you to manually configure connection settings. This is useful in cases where a server doesn't allow anonymous endpoint access, but provides separate discovery endpoints.

5. Click **Next**.
6. Choose a server, then click **Next**.

The screenshot shows the 'OPC Connections' configuration window. At the top, there is a breadcrumb trail: 'Config > Opc > OPC Connections'. A blue link '→ Skip to Advanced Configuration' is in the top right. Below this is a section titled 'Choose a Server'. Inside this section, 'Ignition OPC UA Server' is listed with a radio button selected next to the endpoint 'opc.tcp://localhost:62541/discovery'. At the bottom of the window, there are two blue buttons: 'Previous' on the left and 'Next' on the right.

7. Choose an endpoint, then click **Next**.

The screenshot shows the 'OPC Connections' configuration window at the 'Select Endpoint' step. The breadcrumb trail 'Config > Opc > OPC Connections' and the '→ Skip to Advanced Configuration' link are at the top. The 'Select Endpoint' section contains a radio button selected next to the endpoint 'opc.tcp://localhost:62541'. Below the endpoint, the 'Security Policy' is set to 'Basic256Sha256' and the 'Security Mode' is set to 'SignAndEncrypt'. At the bottom, there are 'Previous' and 'Next' buttons.

8. A Manage Certificate window will open if you haven't previously trusted a certificate. If the certificate that the Endpoint sends you to is already in Ignition's trust store, this step is skipped. Trust the Certificate and click **Next**.

Manage Certificate

Common Name: Ignition OPC UA Server
Organization: Inductive Automation
Organizational Unit:
Locality: Folsom
State or Province: CA
Country: US
SHA-1 Fingerprint: 7e326915c0d7a659898b65a47ba385a7566d6191
Expiration: Fri May 19 00:00:00 PDT 2023

Trust certificate?

☒ No ☐ Yes

Previous

Next

9. If you entered a discovery URL in step 4, you also have an option to enter another URL if the host is unreachable.
10. Select a Security Policy and Message Security configuration to use when connecting to the endpoint, then click **Finish**. The policies that appear here are determined by the server.
11. A confirmation page is displayed. Click **Finish**.

⚙ Config > Opc > OPC Connections

→ Skip to Advanced Configuration

Confirm Connection Settings

Discovery URL: opc.tcp://localhost:62541/discovery
Endpoint URL: opc.tcp://localhost:62541
Endpoint Host Override:
Endpoint Security Mode: SignAndEncrypt
Endpoint Security Policy: Basic256Sha256

Previous

Finish

12. On the **New OPC UA Connection Settings** page, give the connection a name. Some OPC UA servers may require a username and password, but this is not always the case. Check with the OPC UA server's documentation for more details. Credentials for Ignition's OPC UA server can be found on the [Ignition's OPC UA Server](#) page.

Main	
Name	New Name OPC Server 
Description	
Enabled	☒ (default: true)
Read Only	☐ If selected, the connection to this OPC server will be read-only; all calls to write will fail. (default: false)

Authentication	
Username	
Password	 
Password	  Re-type password for verification.

☐ Show advanced properties

Create New OPC Connection

13. Once credentials have been entered, click the **Create New OPC Connection** button.

Ignition is now connected to the OPC UA server.

OPC UA Client Connection Settings

The following tables describe all the available properties.

Main

Property	Description
Name	A name used to identify this connection.
Description	Short description of this connection.
Enabled	Disable the connection to the OPC server.
Read-only	Puts the connection into read-only mode. All writes sent to this server will fail.

Authentication

Property	Description
Username	A username the connection will use when authenticating with the UA server.
Password Fields	The password to use when authenticating with the UA server.

Advanced

Property	Description
Host Override	When specified, if the endpoint address returned by the OPC server has a different IP address or hostname than the discovered endpoint, the overridden value will be used. Expects just an IP address or hostname, for example: 192.168.1.10
Connect Timeout	The timeout, in milliseconds, when opening a socket connection to a remote host. Default is 5,000.
Acknowledge Timeout	The timeout, in milliseconds, to wait for an Acknowledge message in response to the client's Hello message. Default is 5,000.
Request Timeout	Maximum amount of time, in milliseconds, to wait for the response to a request. Default is 60,000.
Session Timeout	Requested session timeout value, in milliseconds. Default is 120,000.
Max Per Operation	Specify the maximum number of nodes to read, write, subscribe, or unsubscribe to in any given UA server request. Default is 8,192.
Max References Per Node	Configures the number of references per node. A "node" in this case is any item inside of a UA server, so items like tags and folders would qualify as a node, while references simply means a reference to another node. This setting is useful in situations where the address space is completely flat, so a large number of adjacent nodes could potentially run into a maximum message size. In these cases decreasing the value of this property can be useful. However, most systems will not need to change this setting. Defaults to 8,192 references.
Max Pending Public Requests	The number of concurrent Publish Requests allowed to be pending at any given time. Default is 2.
Max Notifications Per Publish	The maximum number of notifications per publish. Default is 65,535.

Property	Description
Max Message Size	The maximum allowable size of an OPC UA application layer message. Default is 33,554,432.
Max Array Length	The maximum allowable size for arrays. Default is 2,147,483,647.
Max String Length	The maximum allowable size for strings. Default is 2,147,483,647.
Type Dictionary Fragment Size	The fragment size to request when reading the server's type dictionary. Default is 8,192.
Keep-Alive Failures Allowed	Number of consecutive failures allowed before disconnecting. Setting this to ≤ 0 means consecutive failures will not cause a disconnect. Default is 1.
Keep-Alive Interval	Interval, in milliseconds, between keep-alive requests.
Keep-Alive Timeout	Max duration, in milliseconds, to wait for a response to a keep-alive request. Default is 10,000.
Browser Origin	The Node that browsing should originate from. Options are OBJECTS_FOLDER or ROOT_FOLDER. Most OPC UA Servers use OBJECTS_FOLDER, but some non-standard servers may require ROOT_FOLDER to browse correctly. Ignition's OPC UA Servers uses OBJECTS_FOLDER.

Failover

Property	Description
Failover Enabled	Enable failover on the connection, allowing the UA client to switch to a backup server in the event the primary server is unavailable.
Failover Threshold	The number of retry attempts before the failover connection is used. The default is 3.
Failover Discovery URL	The discovery URL for the backup server's OPC UA server. Expects the following format: <code>opc.tcp://hostname:port</code>
Failover Endpoint URL	The endpoint of the failover server. Example: <code>opc.tcp://192.168.1.0:62541</code>
Failover Host Override	When specified, if the endpoint address returned by the failover OPC server has a different IP address or hostname than the discovered endpoint, the overridden value will be used. Expects just an IP address or hostname. Example: <code>192.168.1.10</code>

Security

Property	Description
Certificate Validation Enabled	New in 8.1.0 Enables validation of server certificates. This is required by the OPC UA specification, but it may be disabled for troubleshooting or temporarily connecting to servers with invalid or untrusted certificates. Default is true. Caution: Disabling certificate validation compromises the security of the connection.
KeyStore Alias	The alias of the certificate and private key stored in the client KeyStore.
Password Fields	The password to use when authenticating with the UA server.

Failover Versus Backup Properties

The Failover properties should be used when a single Ignition Gateway needs to connect to a pair of redundant OPC UA servers. The failover OPC UA server will be used in the event the primary OPC server goes down. To enable failover, set the **Failover Enabled** property to true, and specify the **Failover Endpoint**. The **Failover Threshold** can be adjusted if desired.

NOTE

Failover events are "sticky." That means once control has moved to a backup OPC UA server, it stays there until that server fails.

Ignition's OPC UA Server

Ignition's OPC UA server, provided by the [OPC UA module](#), allows an Ignition installation to utilize Ignition's various device driver modules. In addition, with the module installed, OPC UA clients can connect to Ignition's UA server, exposing any connected devices to third party systems.

Settings for the server can be found under the **Config** section of the Gateway Webpage. On the sidebar, locate **OPC UA > Server Settings**.

Default Credentials

Ignition's OPC UA server does not initially support anonymous access, but can be configured to do so (see the settings table below). Authenticated connections require the following credentials:

- **Username:** opcuauser
- **Password:** password

New installations of Ignition will automatically create the user above, allowing the Gateway to initially connect as a UA client to its own UA server.

Connecting with UA Discovery

Ignition's OPC UA server is initially, and intentionally, difficult to discover on new installations. To aid with discovery attempts, a separate unsecured endpoint is available, allowing UA clients a means of finding the server. When attempting to discover the server, the endpoint URL should include `"/discovery"` at the end:

```
opc.tcp://<localhost>/discovery
```

OPC UA Server Settings

NOTE

Changes made to any of the following OPC UA Server settings requires a restart (of either the Gateway or the OPC UA module) before the changes will take effect.

The table below represents settings on Ignition's OPC UA server. They'll only become available if the [OPC UA Module](#) is installed on the Gateway.

Endpoint Configuration

Setting	Description	Default Value
Bind Port	The port the UA server will bind to.	62,541
Bind Addresses	The address the server will bind to. If you want to expose the OPC UA server to external sources, you need to use 0.0.0.0 or the IP address of the computer.	localhost
Endpoint Addresses	A comma separated list of endpoint addresses that the UA server can be reached at. It is important that this is set to addresses that can be reached by any UA clients attempting to connect to the server. When entering addresses into this property, they can be just an IP address or hostname: 10.10.10.100 Alternatively, angled brackets can be used. When applied to an address, the server attempts to find the hostname, or resolve the value to as many addresses or hostnames as it can find. <10.10.10.100>	<hostname> , <localhost>
Security Policies	A comma separated list of acceptable security policies. Available policies are: • None • Basic256Sha256 • Aes128_Sha256_RsaOaep • Aes256_Sha256_RsaPss In addition, the following deprecated policies are available, but not recommended: • Basic128Rsa15 (deprecated) • Basic256 (deprecated)	Basic256Sha256

Authentication

Setting	Description	Default Value
Anonymous Access Allowed	Specifies if UA clients are allowed to connect to this server anonymously. While false, client connections are required to authenticate with the server.	false
User Source	Which user source contains the initial user for authenticated access. Credentials for the initial user can be found above.	Attempts to use the 'opcua-module' user sources

Advanced

Setting	Description	Default Value
Expose Tag Providers	When enabled, Ignition Tag Providers will be exposed through the UA server, allowing third party UA clients to access tags in the provider. An OPC-UA module restart is required when changing this setting.	false
Max Session Count	New in 8.1.17 The maximum number of client connections to the UA server.	100

Redundancy

Setting	Description
Backup Bind Addresses	The local addresses that the UA server will attempt to bind to while the backup pair .
Backup Endpoint Addresses	The endpoint addresses that the UA server can be reached at while the server backup in a redundant pair . The notation on this property is similar to the Endpoint property above, in that angled brackets can be used with each hostname and
Read-only When Inactive Node	When enabled, this server switches to a read-only state while its Gateway is the redundant pair .
Master Application URI	New in 8.1.10 Application URI that a redundant backup will advertise in its address belonging to the server on the redundant master. When enabling OPC UA redundancy, enter the master Gateway's URI here. URIs will generally look like the following: urn:inductiveautomation:ignition:opcua:server:#####-####-#### Note: An OPC UA server's URI can be found by reading the Ignition OPC UA ServerArray tag. In addition, the URI is reported in the Gateway's wrapper log.
Backup Application URI	New in 8.1.10 Application URI that a redundant master will advertise in its address belonging to the server on the redundant backup. When enabling OPC UA redundancy, enter the backup Gateway's URI here.

OPC UA Client Redundancy

New in 8.1.10

Ignition's OPC UA server supports non-transparent redundancy. This allows third party OPC UA clients to connect to the active node in a pair of redundant Gateways. If the master Gateway goes down, then the OPC UA client would be able to switch to the backup Gateway, following the active node as the system changes.

The steps below demonstrate how to configure OPC UA redundancy.

1. Start by having [Gateway redundancy](#) configured between two Ignition Gateways.

- Next we need to find URIs for the OPC UA servers on each node. These can be browsed from a quick client. On the master Gateway, navigate to **Config > OPC Client > OPC Quick Client**.
- In the quick client, browse down to the **Ignition OPC UA Server > Server**.
- We'll need to **read** (not subscribe) to the ServerArray. Press the **[r]** link next to **ServerArray**. This will make the server's URI appear in a panel above the tree. Copy the URI, but omit the square brackets.

✓ Read completed. [Ignition OPC UA Server]ns=0;i=2254
 Value: **urn:inductiveautomation:ignition:opcua:server:d5912ec9-5018-44f0-a55a-729a1480f938**
 Quality: Good
 Timestamp: 8/16/21, 11:43:34 AM PDT

TYPE	ACTION	TITLE
Server	refresh	+ Eclipse Milo OPC UA Demo Server
Server	refresh	- Ignition OPC UA Server
Object		+ Devices
Object		- Server
Object		+ ServerCapabilities
Object		+ ServerConfiguration
Object		+ ServerDiagnostics
Object		+ ServerRedundancy
Object		+ VendorServerInfo
Tag	[s][r][w]	+ Auditing
Tag	[s][r][w]	+ EstimatedReturnTime
Tag	[s][r][w]	+ NamespaceArray
Tag	[s] [r] [w]	+ ServerArray
Tag	[s][r][w]	+ ServerStatus

- This is the master's URI. Temporarily store the URI, as we'll need to pass it into a setting later.
- Next, switch over to the backup Gateway. Repeat steps 2-5 to obtain the backup Gateway's URI.
- With both URIs, switch back to the master Gateway.
- Navigate to **Config > OPC UA > Server Settings**.
- For the **Master Application URI setting**, enter the master's URI.
- For the **Backup Application URI setting**, enter the backup's URI.
- Press **Save Changes**.

From this point on, third party OPC UA clients can connect to the active node's OPC UA server, and will fail over when the active node switches. While OPC UA redundancy is enabled, the Ignition **OPC UA Server > Server > ServiceLevel** tag can be used to denote which server is running as the master.

Service Level	Description
255	The OPC Server is on the master Gateway, and the master is the active node. Note that this value is also used in cases where redundancy is not enabled.
254	The OPC Server is on the backup Gateway, and the backup is the active node.
1	The OPC Server is the inactive node. Meaning the other node is currently active.

 NOTE

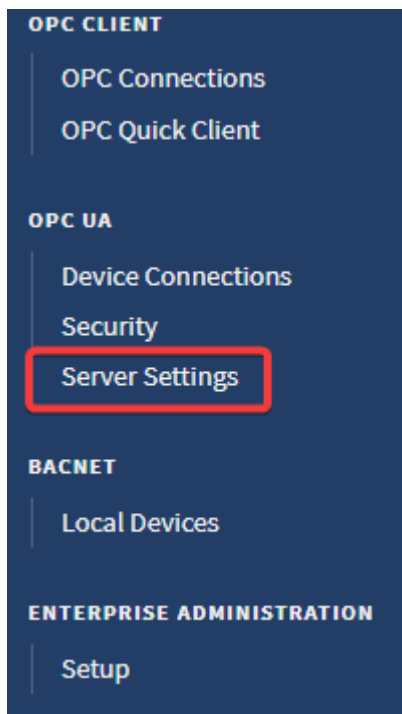
It is possible for both nodes to be active, where the master shows a value of 255 and the backup simultaneously shows a value of 254. This generally happens in cases where the two nodes are unable to communicate with each other. For example, when a network disconnect occurs between the two Gateways, or if the Gateway network connection between the two is pending approval.

Troubleshooting a Faulted Connection to Ignition's OPC UA Server

You may occasionally run into issues with Ignition's OPC UA Server connection. In these situations, there are a few things you can check to diagnose your issue.

To troubleshoot your connection to Ignition's OPC UA Server, follow the steps below:

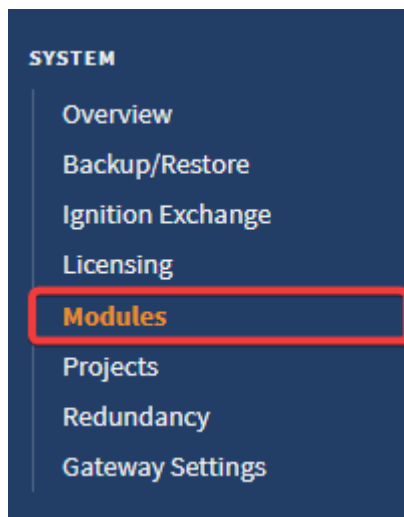
1. Go to the Server Settings page for Ignition's OPC UA Server. This page is located on the Gateway webpage **Config > OPC UA > Server Settings**.



2. Check your Endpoint Addresses setting. The default value for this setting is `<hostname>`, `<localhost>`. The IP address of the internal OPC UA Server, which can be found on the Status page of your Gateway, can also be appended.

Endpoint Configuration	
Bind Port	62541  (default: 62,541)
Bind Addresses	localhost (default: localhost)
Endpoint Addresses	<hostname>,<localhost> (default: <hostname>, <localhost>)
Security Policies	Basic256Sha256 (default: Basic256Sha256)

3. Restart the "OPC-UA" module. After making changes to your settings, restarting your module may help flush out any residual information. You can restart the module by going to your Gateway webpage **Config > System > Module**, locating OPC-UA and selecting **restart**.



OPC-UA

9.1.20-SNAPSHOT
(b2022071314)

Provides Ignition's OPC UA client and server functionality.

Trial

Running

More ▾

restart

4. Check your Bind Addresses. The default value for this setting is `localhost`. If you want to expose the OPC UA Server to external clients, you can use a value like `0.0.0.0` or the IP address of the computer.

Endpoint Configuration	
Bind Port	62541  (default: 62,541)
Bind Addresses	localhost (default: localhost)
Endpoint Addresses	<hostname>,<localhost> (default: <hostname>, <localhost>)
Security Policies	Basic256Sha256 (default: Basic256Sha256)

5. Check your Security Policies. Possible values for this setting are listed in the table under **OPC UA Server Settings**.

Endpoint Configuration	
Bind Port	62541  (default: 62,541)
Bind Addresses	localhost (default: localhost)
Endpoint Addresses	<hostname>,<localhost> (default: <hostname>, <localhost>)
Security Policies	Basic256Sha256 (default: Basic256Sha256)

If you continue running into problems after following these troubleshooting steps, contact the [Support department](#).

Third Party OPC Servers

Ignition has a built-in OPC UA server that includes all of the drivers in this section, but if your devices aren't listed here, there is another way to connect to them. Ignition can quickly and easily connect to third party OPC servers via OPC UA or OPC DA (using the OPC COM module) for devices that do not have a supported driver. This opens up the possibilities to connect Ignition to any device with servers like Kepware, Matricon, etc. With over 100 driver suites each, you are sure to be able to connect anything to Ignition.

OPC Server Connections

Name	Type	Description	Read-only	Status	
Ignition OPC-UA Server	OPC-UA	The default connection to Ignition's OPC-UA server.	false	Connected	More ▾ edit
Kepware	OPC-UA		false	Connected	More ▾ edit

→ [Create new OPC Server Connection...](#)

Note: For details about a connection's status, see the [OPC Connection Status](#) page.

OPC UA

OPC UA is the new Unified Architecture for connecting to OPC devices. Probably the best feature of this new standard is that it can be used on any OS, it is not tied to Microsoft like its predecessor is. Many companies make an OPC UA server, and between them grant access to hundreds of driver suites, giving you access to thousands of available devices through Ethernet, serial, and other types of communication. If Ignition doesn't have it already, you can still get your data in.

Connecting to an OPC-UA server is simple (locally or remote), but because of the increased security, you need access to both the OPC server and Ignition to complete the setup. For step-by-step instructions to connect to a Kepware OPC server, see [Connecting to Kepware OPC UA](#). All other OPC UA servers will have extremely similar connection steps.

NOTE

While Ignition can connect to third-party OPC UA servers, these servers must include support for at least OPC UA 1.03 datatype dictionaries in order to read structured values.

OPC COM (DA)

OPC COM (often called OPC-DA) has been the way to connect to OPC devices for many years, but the world has since out-grown it with the creation of OPC-UA. It relies on the .NET framework and because of this is restricted to Microsoft operating systems. There is no way to install an OPC COM server on Mac or Linux. There are many still in use today so it cannot be abandoned completely yet, but Ignition has an **OPC COM module** to connect to them.

! INFO

The OPC COM module only provides Client Access to OPC DA servers and cannot be used to expose Ignition data to other OPC clients

OPC COM servers are notoriously difficult to connect to, especially if you want to connect to one on a remote server. If you need to do so, using Ignition's **OPC COM Tunneller module** is highly preferred over setting up a Remote COM connection.

Connecting to Kepware OPC UA



INDUCTIVE
UNIVERSITY

Connecting to Kepware OPC- UA

[Watch the video](#)

OPC UA makes connecting to third party OPC servers quick and easy without all the headaches associated with COM. This is a detailed step-by-step guide to connecting to KEPServerEX from Ignition using OPC UA.

Connect to KEPServerEX from Ignition using OPC UA

1. In the **Config** section of the Gateway, go to **OPC Client > OPC Connections**. The OPC Connections page is displayed showing the OPC UA servers your Ignition is connected to.
2. Click on **Create new OPC Connection...**
3. Choose **OPC UA** as the connection type, and click **Next**.
4. On the Server Discovery page, enter the endpoint of the OPC UA server Ignition should connect to. For example:

Sample Format	Localhost Example, Default Port	Remote Example, Custom Port
opc.tcp://IpAddress:Port	opc.tcp://localhost:49320	opc.tcp://10.1.1.10:4444

Click on the **Next** button to continue.

Server Discovery

Endpoint URL

Example: opc.tcp://localhost:62541/discovery

[Next](#)

5. Select the Server you want and click **Next**.

Choose a Server

KEPServerEX/UA@TR-0690157-SB.ia.local

☒ **opc.tcp://127.0.0.1:49320**

[Previous](#)[Next](#)

6. A list of available Endpoints with Security Policies and Security Modes options appears.

Select Endpoint

☒ **opc.tcp://127.0.0.1:49320**

Security Policy: Basic256Sha256

Security Mode: SignAndEncrypt

[Previous](#)[Next](#)

Once an endpoint configuration has been selected, click the **Next** button.

7. On the Manage Certificate page select **Yes** for **Trust Certificate?** and click Next.

Since Ignition does not automatically approve third party OPC UA server certificates, a backup Gateway will need the certificate approved manually for any created OPC UA connection. This can be done on the **Config > OPC UA > Security** page under the **Quarantined Certificates** section.

[→ Skip to Advanced Configuration](#)

Manage Certificate

Common Name: KEPServerEX/UA Server

Organization: Unknown

Organizational Unit:

Locality:

State or Province:

Country: US

SHA-1 Fingerprint: 60df905dc826cbce78594cd590428e37c5e2bef1

Expiration: Sat Jun 24 15:36:00 PDT 2023

Trust certificate?

☐ No ☒ Yes

[Previous](#)

[Next](#)

8. Confirm your settings and click **Finish**.

[→ Skip to Advanced Configuration](#)

Confirm Connection Settings

Discovery URL: opc.tcp://127.0.0.1:49320

Endpoint URL: opc.tcp://127.0.0.1:49320

Endpoint Host Override:

Endpoint Security Mode: SignAndEncrypt

Endpoint Security Policy: Basic256Sha256

[Previous](#)

[Finish](#)

9. This takes you to the new OPC Connection screen. Fill in the **Username** and **Password** if your KEPServer connection requires it.

Most current installations of KEPServer require a login and will not connect without one. See the [No Anonymous Token Policy Found](#) section below.

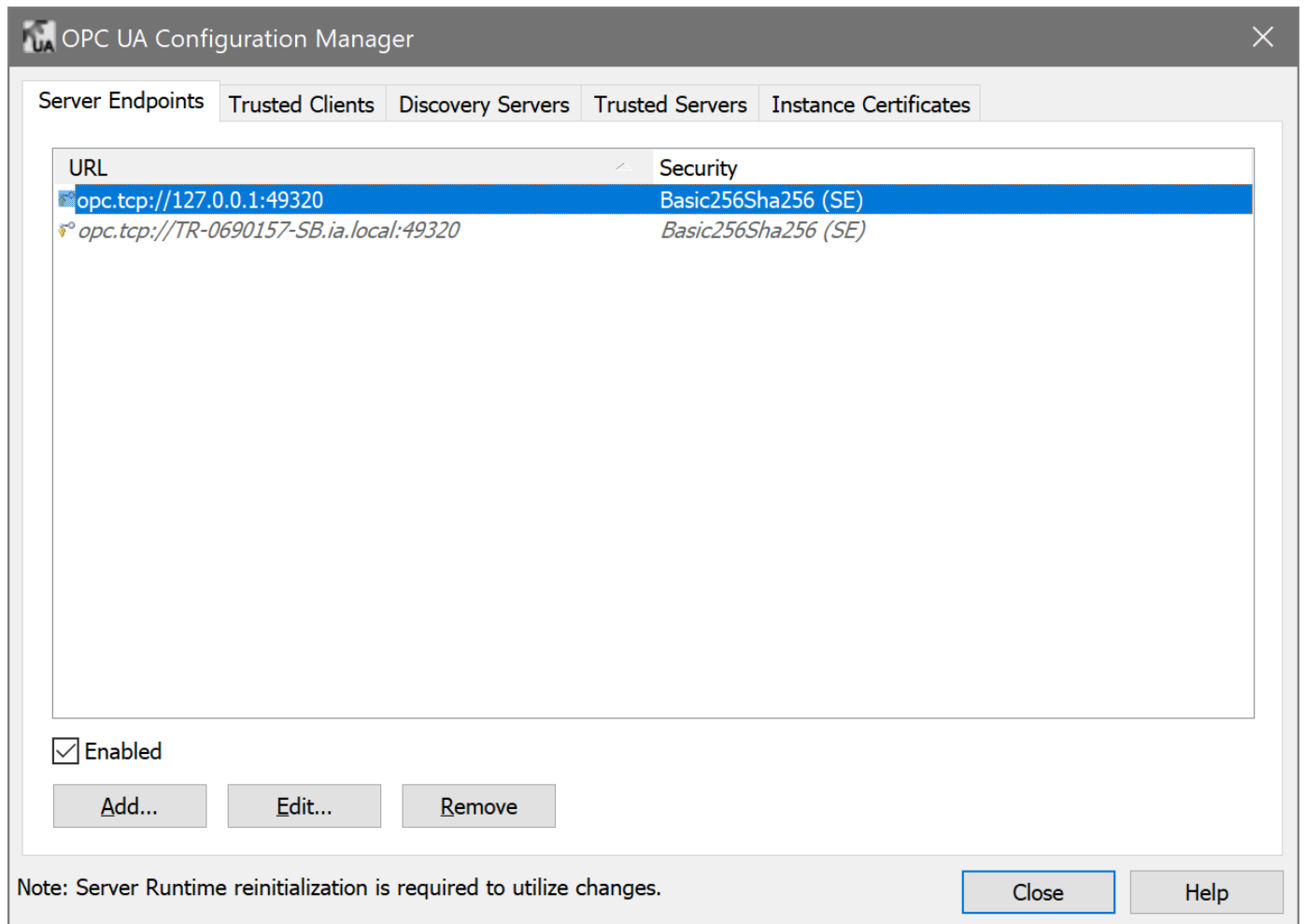
Main	
Name	KEPServerEX/UA 
Description	
Enabled	☒ (default: true)
Read Only	☐ If selected, the connection to this OPC server will be read-only; all calls to write will fail. (default: false)

Authentication	
Username	
Password	
Password	 Re-type password for verification.

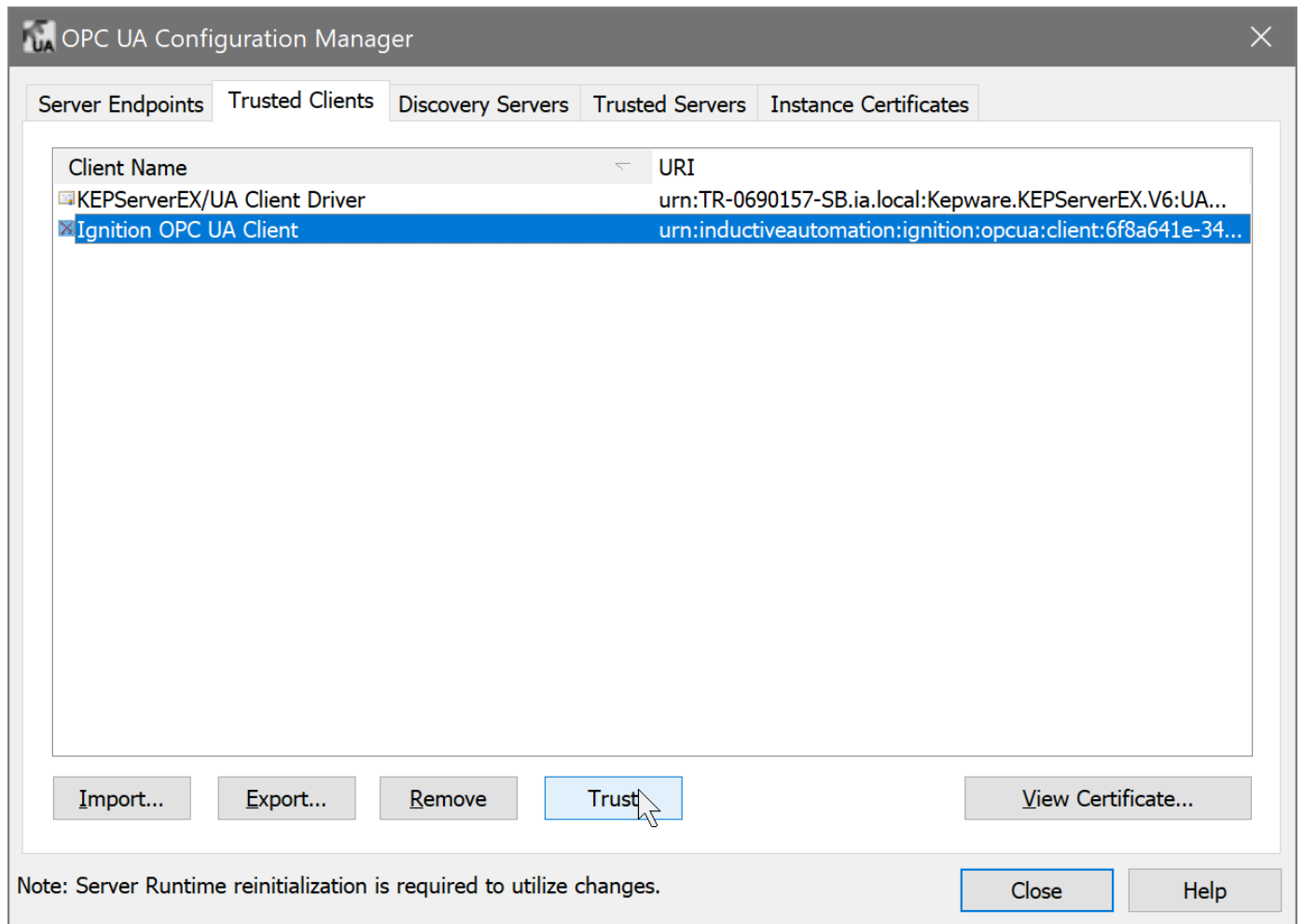
☐ Show advanced properties

Create New OPC Connection

- Click **Create New OPC Connection**.
- The connection will appear as **Faulted**. This is expected because KEPServerEX is denying access to the Ignition OPC UA Client. The next step is to have KEPServerEX trust the Ignition OPC UA Client.
- On the computer that KEPServer is installed on, right-click on the **KEPServerEX** icon on the desktop KEPServerEX is installed on, and from the menu select **OPC UA Configuration**. The **OPC UA Configuration Manager** will appear.



13. On the OPC UA Configuration Manager window, go to the **Trusted Clients** tab.
14. Click on **Ignition OPC UA Client**, click the **Trust** button, and click **Close**. Now the OPC Server Connections page shows the Status of Kepware to be Connected.



15. Again, right-click on the **KEPServerEX** icon on the desktop KEPServerEx is installed on, and from the menu select **Reinitialize**.
16. Go back to the Ignition Gateway Webpage. In the **Config** section, go to the **OPC UA > Security** page.
17. Under the Client Security tab, you will find your new connection listed as Quarantined. Click on the **Trust** button on the far right.

Client

Server

Client Security

Trusted Certificates

	Common Name	SHA-1 Fingerprint	Expiration	Action	
+	Ignition OPC UA Ser...	fd012b875173fa53e...	Jun 24, 2023	Delete	↓
+	KEPServerEX/UA Ser...	60df905dc826cbce7...	Jun 24, 2023	Delete	↓

Upload Trusted Certificate:

Browse

or Drag files here

Quarantined Certificates

	Common Name	SHA-1 Fingerprint	Expiration	Action	
+	KEPServerEX/UA Ser...	60df905dc826cbce7...	Jun 24, 2023	Trust Delete	↓

- Go back to the **Config** section of the Gateway, to **OPC Connections > Servers**. The Status of your KEPServer connection should be **Connected**.
- To test your tag connections, go to the **OPC Connections > Quick Client** in the Configure section of the Gateway. Expand the **KEPServer** object until you find tags.

Troubleshooting

If **Status** does not read **Connected**, click the **edit** link next to the server connection, scroll down to the bottom of the connection configuration page, and click **Save**. If **Status** is still reading something other than **Connected**, click the **OPC Connection Status** link at the bottom of the **OPC Server Connections** page and see if there are any useful messages to help troubleshoot the issue. Also, ensure your firewall is not blocking traffic on the port that KEPServerEX is using to communicate.

Failover

The failover Kepware OPC UA server works the same as the OPC UA server with the exception that you need to have two copies of Kepware set up, preferably on different servers. The failover Kepware OPC UA server will be used in the event the primary Kepware server goes down. To enable failover,

check the box to **Show advanced properties** in the New OPC UA Connection Settings, set the **Failover Enabled** property to 'true', and specify the **Failover Endpoint**.

The **Backup** properties should be used when a pair of redundant Ignition Gateways are trying to look at the same Kepware OPC UA server. Both the **Backup Discovery URL** and **Backup Endpoint URL** properties need to be configured.

For additional information on Failover, refer to [OPC UA Client Connection Settings](#).

No Anonymous Token Policy Found

When connecting to KepServer, some versions may not allow anonymous connections by default. This typically means you need to specify user credentials for Ignition to use in the OPC UA server connection. Alternatively, individual Kepware Projects can allow anonymous login. For more information, look into allowing "anonymous login" in [KepServer's OPC UA Configuration Manager documentation](#).

Other UA Servers

While the above example is specific to KEPServerEX, the same concepts apply to connecting to any other third party OPC server that accepts OPC UA client connections. The only difference may be in the way that the certificates are accepted on the server.

The Ignition OPC UA server sends the client certificate to the third party OPC server when it tries to make the connection, however if the OPC server is not designed to expect these certificates then there may not be a straight forward way to accept them. In these cases, you can manually download a client ticket from Ignition and supply it to the OPC server in the appropriate manner.

Download a Client Certificate Manually

1. Go to **Config** section in the Gateway Webpage.
2. Select **OPC UA > Certificate** from the left side of the page. The Manage Certificates page is displayed.
3. In the **This Gateway** tab, click the download link under **Ignition OPC UA Client**, and save the certificate somewhere to disk. This certificate is then supplied to your third-party OPC server in a way specific to that server. For more information, check the respective server's documentation.

OPC COM

Connecting to OPC Classic (COM)

NOTE

Classic OPC is based on COM, which is a technology in Microsoft Windows. Therefore, the information in this section only applies to Ignition Gateways installed on Windows. For other operating systems, OPC UA must be used.

INFO

The OPC COM module is for Client Access only, and should not be used for direct server-to-server or external third-party access.

The OPC-COM module provides the ability to connect to OPC servers that only communicate using the older COM based OPC-DA standard. If you have an OPC server that is not capable of accepting OPC UA connections and you need to talk to a PLC for which Ignition has no supported driver, you'll have to use the OPC-COM module to make your device data available in Ignition. Connections to OPC servers will be held open while the Ignition Gateway is running. All subscriptions to the server will use the same connection.

This section provides a brief walk-through of how to set-up a new Local or Remote OPC-DA server connection using the COM module. Due to the complications that Windows DCOM security settings can cause, this set-up guide is followed by the Troubleshooting OPC-COM Connections section that deals with an overview of how to deal with a faulted server connection due to DCOM security settings as well as other possibilities.

Install OPC Core Components

1. Register at www.opcfoundation.org.
 - The OPC-COM module relies on a .dll package provided by the OPC Foundation (www.opcfoundation.org) called the OPC Core Components. You can download the OPC Core Components Redistributable from the OPC Foundation's website under the downloads section. Registration with the OPC Foundation is required before you can download the package, but the registration process is free and painless.
2. Download appropriate OPC Core Components Redistributable package.

- There are two packages to choose from, the 32-bit (x86) and the 64-bit (x64), make sure you get the correct one for the version of Java and Ignition you are running. 64-bit Java and Ignition needs the 64-bit Core Components package and likewise 32-bit installations needs the 32-bit package. You may have to look around a bit for the redistributable. At last update of this page (2019), the download was listed under [Resources](#) → [Samples and Tools](#) → [classic](#).

3. Install Core Components on Ignition server.

- It should be noted that if you are going to connect to an OPC server on a remote machine, you must also install the appropriate version of the Core Components on that server as well. The version type, 64-bit or 32-bit, does not need to be the same across the two servers. Just be sure to install the version that is appropriate for the OPC Server and Windows architecture.

4. (Remote) Install Core Components on remote machine running the OPC-DA server.

- Once you have the correct package downloaded you can extract the contents of the .zip file and then run the installer. With the core components installed you can now proceed to setting up your OPC-DA server connection in Ignition.

Connecting to OPC DA Server

With the OPC Core Components now installed, the next step is creating/configuring a new OPC DA server connection.

Install OPC DA Server Connection

1. Go to the Ignition Gateway Config section.
2. Go to OPC Connections > Servers and then select **Create new OPC Server Connection....**
3. Choose the **OPC DA COM Connection** and then select whether you want to make a Local connection or if the OPC server resides on a Remote machine. For the most part, setting up a local or remote connection to an OPC DA server is the same. There are only a couple of differences for a remote connection that will be highlighted along the way.
 - **Local** - Selecting a local connection takes you to a screen that contains a list of the available and running OPC servers located on the local machine.
 - **Remote** - For a remote connection you first have to specify the host name or IP address of the machine the the OPC server resides on and then (as of Ignition 7.4) you are redirected to the available servers list.

4. Select the OPC server that you wish to connect to from the list. In the case where your server is not listed, see the **OPC server is not listed...** the Troubleshooting OPC COM Connection section.
 - **Unique Remote Connection Settings** - Remote connections have a few unique settings that you can specify. You can get to these settings by selecting the Show advanced properties check box. As of Ignition 7.4 these should all be set for you (except for the CLSID which should no longer be necessary but is still available for you to set if you wish).
 - **Remote Server** - Specifies that the server is remote and that a DCOM connection will be used.
 - **Host Machine** - The computer name or IP address of the machine on which the remote server is running.
 - **CLSID** - This is no longer required as of Ignition 7.4, but it is still made available for you. It can be used in place of the ProgId because the ProgId is really just used to lookup the CLSID in the registry. This id can be found in the registry of the machine hosting the server under: `HKEY_CLASSES_ROOT\OPCServerName\CLSID`.
5. All of the settings for the server connection are rather straight forward and each property has a description of its functionality. Most of these settings should be fine when left at their default values. The only setting that could possibly give you some trouble is the ProgId. If you selected your OPC server from the list on the **Choose OPC DA Server** page, this will be filled in for you. However, if for whatever reason your server wasn't listed and you choose the **Other Server** option, you will have to know the ProgId for your server and specify it here. The ProgId is used to look up the CLSID of the OPC Server in the Windows Registry and without this a connection cannot be made.
6. When you are finished fine tuning these settings click **Create new OPC Server Connection**. You will be redirected to the OPC Server Connections page and your new server connection should be listed. The status of your connection will read Connected if Ignition was able to successfully connect to the third-party OPC server.

Connection Is Faulted

In the case where your connection status is reporting Faulted, the troubleshooting process begins. As previously stated, configuring the DCOM settings on your machine can be a headache. The [Troubleshooting OPC COM Connections](#) section next is an attempt to ease the process of determining why your connection is faulted and how to go about fixing the issue. If after exhausting the options presented to you, you are still having issues getting you server connection up, give our Inductive Automation tech support line a call and one of our representatives will be happy to assist you.

Troubleshooting OPC COM Connections

This section provides you with a list of common OPC-COM connection problems with their possible solutions. It would be impossible to give an exhaustive list of everything that can go wrong but this should give you a good start on the troubleshooting process. If you do not see your problem listed and your connection status is faulted, try following the steps outlined in the [Ignition Server DCOM Settings](#) and [OPC Server DCOM Settings](#) sections.

Common Problems

OPC Server Is not Listed in Choose OPC DA Server List when First Creating a Connection

There are some cases in which an OPC Server that is installed will not show up in the generated list. This list is generated by the OPC Server Enumerator which is part of the OPC Core Components, so when a server you have installed on the machine does not appear in this list it is likely due to the OPC Core Components not being installed correctly.

Try reinstalling the Core Components and going through the process of creating a new server connection in Ignition again. If the server still does not appear and you have the ProgId (or the CLSID for a remote connection) for the OPC server, you can just select the **Other Server** option and then click **Next**. In this situation you will have to enter the ProgId manually on the **New OPC-DA Server** page.

With all the correct information about the OPC server we can sometimes still make a valid connection to the OPC Server even when it is not detected automatically. This however is rare. Most of the time when the server is not detected, any connection attempts Ignition makes will fail.

Connection Status Is Connected but Data Quality Is Bad or the Connection Goes Faulted after Trying to Read Tag Data

Usually this occurs when the DCOM settings for the machine on which Ignition is running are not correctly configured. DCOM connections go in both directions. Ignition must be able to send requests to the OPC server and the OPC server must also be able to callback to Ignition. If the DCOM settings on the Ignition server are not configured correctly those callbacks will fail and the server connection that initially had a status of “Connected” will either fault or all the Tags that you have configured will come back with bad quality.

This is a problem that can affect both local and remote server connections.

Follow the steps outlined in the “Ignition Server DCOM Settings” section to ensure that you have correctly configured the DCOM security settings on the Ignition server machine.

Ignition Launches Second Instance of an Already Running OPC Server and Is Unable to See Any Data

It is important to note that Ignition runs as a service under the Windows System account. This can cause some issues with OPC servers that are meant to run interactively, meaning they run under the user account that is currently logged on. When Ignition attempts to make a connection to the OPC server, it will attempt to find an instance running under the same account and if it doesn't find one it will launch its own instance under the System account. Even if there are other instances running, Ignition will choose the one that was launched under the System account for its connection.

Many OPC servers maintain an instance running under the interactive user account that has been configured by the user and maintains all of the device connection information. When Ignition launches a new instance, this configuration information is lacking and none of the desired data can be seen or accessed. To get around this problem, you must specify in the DCOM settings for the OPC server that it always identify itself with the interactive user. Essentially this will force Ignition to use the currently running instance of the OPC server.

Set the OPC Server to Run as Interactive User

1. The DCOM settings are found in the Component Services manager. Right-click the entry for your OPC server under the DCOM Config folder and select properties from the popup menu.
2. Select the Identity tab. Select the option that reads **The interactive user**, and click **OK**.
3. Close out of component services and kill any extra instances of the OPC server you see running in the Task Manager.
4. Go edit and save the OPC server connection in the Ignition Gateway.

Faulted Status with E_CLASSNOTREG Error Reported on OPC Connections Status Page

This is almost always caused by the OPC Core Components not being installed correctly. Download and install the correct version(s) for your system(s) from the OPC Foundation (www.opcfoundation.org). Remember, if you are making a remote connection you must install these components on both the Ignition server as well as the machine on which the OPC server is running.

DCOM Settings

Ignition Server DCOM Settings

Follow these steps to open up the DCOM security settings on the machine that is running Ignition:

1. Open the **Windows Component Services**, located in the Administrative Tools section of the Control Panel.

2. Browse down through the Component Services tree until you see My Computer, right-click and select **Properties**.
3. We want to focus on the COM Security tab. There are two sections, **Access Permissions and Launch** and **Activation Permissions**. Each section has an Edit Limits... and Edit Defaults... button. You must add the ANONYMOUS and Everyone accounts under each of the four areas making sure that the **Allow** option is checked for each of the permission settings. If you skip adding both of these to either the limits or defaults areas under either of the two sections there is a good chance your connection will not be successful.
4. You can also try setting the Default Authentication Level to **None** and the Default Impersonation Level to **Identify** on the Default Properties tab. This isn't always necessary but it can sometimes help.

OPC Server DCOM Settings

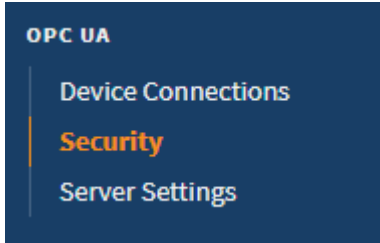
Follow these steps to open up the DCOM security settings on the machine that is running the OPC server:

1. Open up **Windows Component Services**, located in the Administrative Tools section of the Control Panel.
2. Browse down through the Component Services tree until get to the **DCOM Config** folder.
3. Locate the entry for your OPC server that you wish to make a connection to, right-click and select properties.
4. Click the **Security** tab and you will see three sections: Launch and Activation Permissions, Access Permissions, and Configuration Permissions. There are two options to choose from for each section. If you already added the ANONYMOUS and Everyone accounts to the COM Security section from the **Ignition Server DCOM Settings** section then you can go ahead and just select the **Use Default** option for each of the three areas. The second option is to edit each of the groups that have **Customize** selected. You will have to add both the **ANONYMOUS** and **Everyone** accounts with all privileges.
5. Now select the **Identity** tab. You will notice that you can choose which account you want to run the OPC server under. Select the **Interactive User** option. This ensures that if Ignition launches an instance of the OPC server, it will run under whichever user is currently logged into the system.

OPC UA Security

On the OPC UA security page you can manage **OPC UA** certificates for the client and server. Trusted certificates can be imported and quarantined certificates can be marked as trusted.

The OPC UA pages are located under the Gateway's **Config** section, under **OPC UA**:



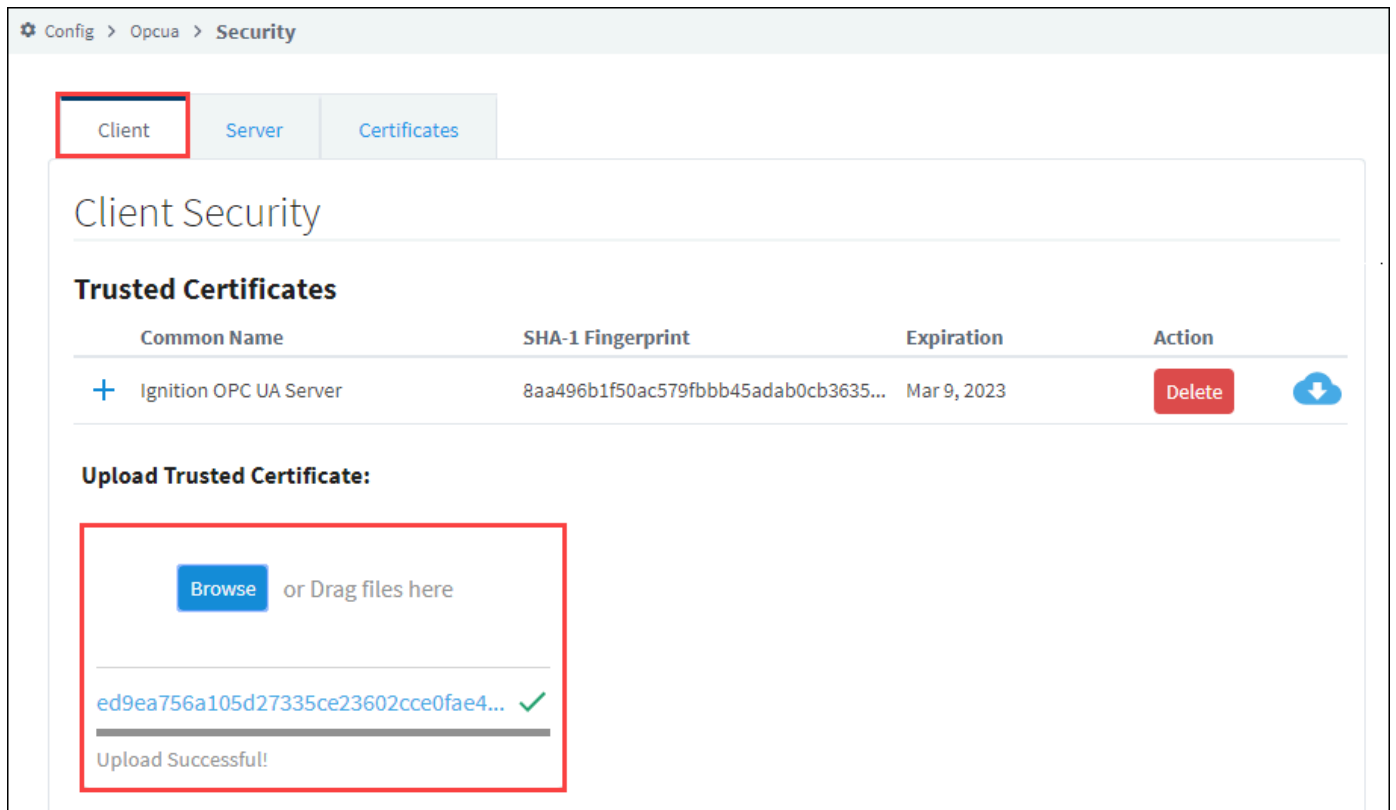
Client and Server Tabs

Both the Client and Server tabs allow you to view OPC UA security certificates. The **Client** tab contains certificates the gateway uses when acting as an OPC UA client, while the **Server** tab contains certificates the Gateway uses when acting as an OPC UA server. Both tabs have the same options in regards to managing certificates.

Upload a Trusted Certificate

The steps for uploading trusted certificates are the same whether you're on the Client tab or the Server tab. To upload a trusted Certificate, do the following.

1. On the Gateway Webpage, select **OPC UA > Security**.
2. Click the **Client** tab or **Server** tab, depending on the what certificate you're uploading.
3. Click the **Browse** button.
4. Navigate to the location of of certificate on your system and click **Open**. (Alternatively, you can drag the certificate file onto the page where it says "Drag files here.")
5. If the upload was successful, you'll see the name of the certificate and the message "Upload Successful!" The certificate will appear in the Trusted Certificates list.



Download a Trusted Certificate

To download a trusted certificate, do the following.

1. Next to the certificate name, click the **Download**  icon.
2. The certificate is downloaded to your system by your web browser.

Delete a Trusted Certificate

To delete a trusted certificate, do the following.

1. Next to the certificate name, click the **Delete** action button.
2. The certificate is deleted.

To view more information about a trusted certificate, click the **More Info**  icon.

Config > Opcua > Security

ClientServerCertificates

Server Security

Trusted Certificates

Common Name	SHA-1 Fingerprint	Expiration	Action
Ignition OPC UA Client	6af7b24fcaa398feb9433b328a2f977d2a...	Mar 9, 2023	Delete Download

	CN	O	OU	L	ST	C
Subject	Ignition OPC UA Client	Inductive Automation		Folsom	CA	US
Issuer	Ignition OPC UA Client	Inductive Automation		Folsom	CA	US

OPC UA Security Page Details

Trusted Certificates

Column Name	Description
Common Name	Name of the certificate.
SHA-1 Fingerprint	The SHA-1 (Secure Hash Algorithm 1) fingerprint is the unique identifier of the certificate.
Expiration	Date the certificate will expire.

Additional Information

Column Name	Description
CN	Common Name
O	Organization, usually the legal incorporated name of a company.
OU	Organizational Unit
L	Locality (Town or City)
ST	State
C	Country, the two-letter ISO code for the country where the organization is located.

Quarantined Certificates

If you import a certificate that is not trusted, it will appear on the Quarantined Certificates list. From here you can view the details by clicking the **More Info**  icon, **Trust** the certificate, or **Delete** it.

Config > Opcua > Security

ClientServerCertificates

Server Security

Trusted Certificates

	Common Name	SHA-1 Fingerprint	Expiration	Action	
+	Ignition OPC UA Clie...	fe37b96f9129a0980...	Sep 19, 2022	Delete	Download

Upload Trusted Certificate:

Browse

 or Drag files here

Quarantined Certificates

	Common Name	SHA-1 Fingerprint	Expiration	Action	
+	Ignition OPC UA Clie...	120fsd30980d455b3...	Sep 19, 2022	TrustDelete	Download

Certificates Tab

New in 8.1.0

The Certificates tab shows the trusted certificates for the OPC UA client and server on the gateway. From this tab the certificates can be examined by clicking the **More Info**  icon. The certificates can be downloaded by clicking the **Download** button. This will perform the same action as downloading a certificate from the **Client** tab as described above.

ClientServerCertificates

Manage Certificates

Client Certificate

Manage the certificate for the OPC UA Client running on this Gateway

	Name	Signature	Expiration	Actions
+	Ignition OPC UA Client	15b6bf9e880996ac46c23ddf535f...	Sep 6, 2025	DownloadRegenerate

Clicking the **Regenerate** button for each certificate will create a new certificate.

Regenerate Current Certificates

All certificates have a definitive live span. For example, the default life span for an Ignition-generated OPC UA certificate is three years. Any OPC UA connection, even the default loopback connection to Ignition's own server, will stop working if the certificate expires or is invalid.

Regenerating the certificates creates a new certificate with an expiration date set for three years later. If your private key is somehow compromised, regenerating a Client or Server certificate also ensures that the private key will no longer work with the Ignition Gateway.

Newly regenerated certificates are automatically trusted by the Gateway issuing them.

Note that regenerating a server certificate will require that the OPC UA **module is restarted**.

New in 8.1.8 Regenerating a client certificate will allow you to specify the duration of the new certificate. In addition, regenerating a server certificate will allow you to specify the duration as well as the DNS names and IP addresses to be included in the Subject Alternate Name (SAN) fields.

Regenerate OPC UA Server Certificate

Note: The OPC UA module must be restarted in order for this to take effect.

Validity Length (Days)

The number of days the generated certificate will be valid.

DNS Names

A comma-separated list of DNS entries to be included in the Subject Alternate Name field.

IP Addresses

A comma-separated list of IP addresses to be included in the Subject Alternate Name field.

Regenerate Server Certificate
Cancel

OPC UA Drivers

A few **Ignition drivers** are included on install in the form of modules that allow you to connect to specific types of devices over OPC UA. Each module provides the ability to connect to different types of devices.

Device Diagnostics Tags

All drivers include built-in diagnostic tags that can be used to monitor the health of the device connection. Some of these metrics can also be found on the **Device Connections Status** page on the Gateway.

Root Diagnostic Tags

Tag	Description
Connected	A Boolean representing the connection between the device and the OPC UA server. Replaced the legacy "isConnected" tag, which was functionally the same.
Hostname	The hostname of the device, as configured in the device configuration on the Gateway.
Name	The name of the device, as configured in the device configuration on the Gateway.
Port	Represents the port used to communicate with the device.
State	A string representing the state of the device configuration. Example: "Connected", "Disconnected".
Status	A string representing the status of the device configuration. Generally provides more granular information than the State tag.

Sampling Tags

New in 8.1.6

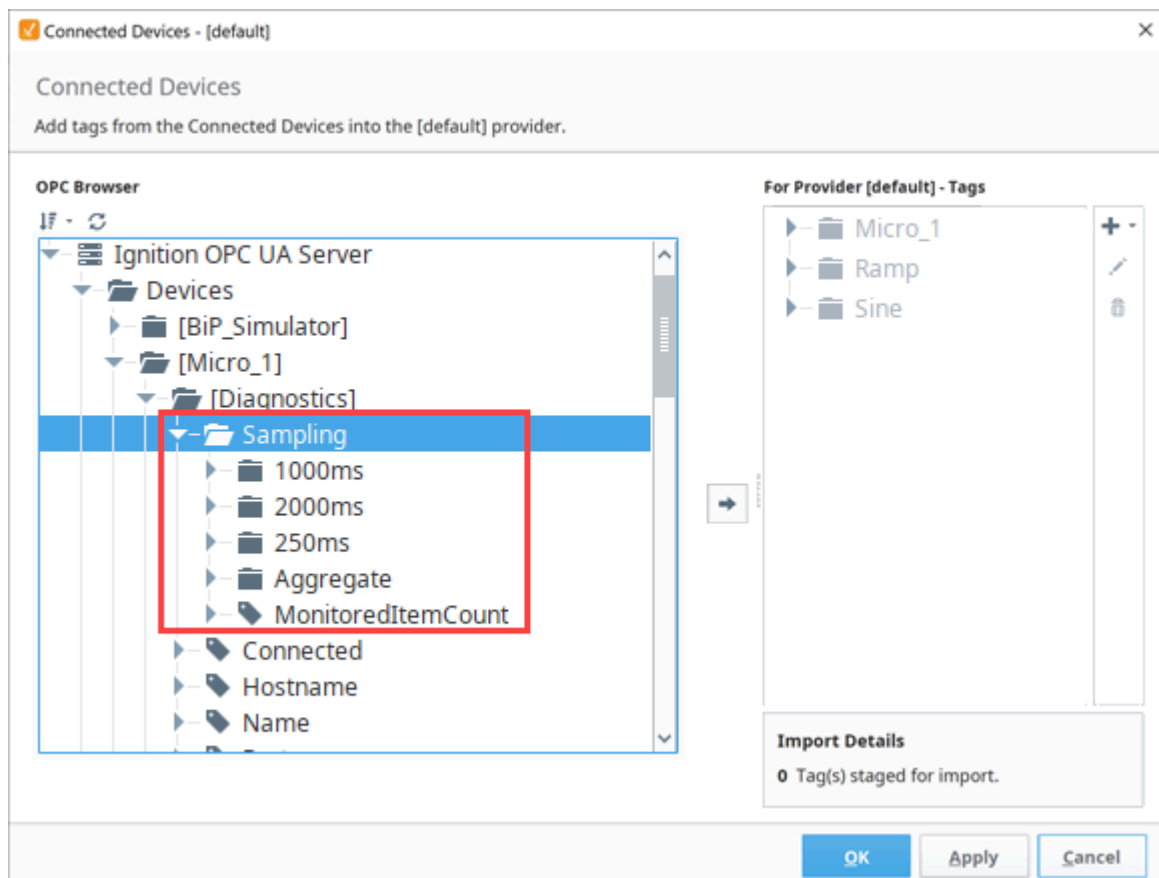
The **Sampling** folder contains metrics of the device connection's throughput. Each device contains an **Aggregate** folder, which represents overall performance of the device connection. The Aggregate folder is always present when browsing device connections that make use of one of Ignition's drivers.

Additional **Sampling Group Folders** are provided based on active subscriptions, and are grouped by sample rate. These folders contain tags that represent throughput of subscription items at set rates. For example, in the image of the **Connected Devices window** below we see several folders next to Aggregate: *250ms*, *1000ms*, and *2000ms* . These folders exist because items in the device are being subscribed to at the rates specified.

This means there is at least one item in Micro_1 that is being subscribed to at a rate of 250ms. Thus, the driver has metrics for items that are being subscribed at that rate. Because there is a 1000ms folder, we know there is at least one item being subscribed at a 1000ms rate, and so on. Thus, every additional sampling folder signifies a different sampling group, and exists because something is being subscribed to at that rate.

NOTE

Subscriptions are most commonly created by [Tag Groups](#), but can be created by other means. For example, Transaction Groups that utilize [OPC Items](#), or the [Gateway's Quick Client](#) page.



New in 8.1.10

Each device contains a **MonitoredItemCount** tag, which represents the total number of items subscribed to on the gateway. Note that this count includes device diagnostic tags.

Requests

When Ignition's drivers attempt to read values from a device, it does so with a grouping process. Updates for individual items are grouped up into **requests**. A request is a grouping of tags/items that need to be read at the same time. The number of items in a single request differs per driver, based mostly off of the protocol being used. Some drivers may have additional configurations options that impact how requests are grouped. Since requests have a fixed number of items to monitor, it's fairly common that there are more pending reads than can fit into a single request. As a result, multiple requests are often used to read values from the same device for the same sampling rate.

All requests are placed into a request queue and wait to execute. The **Sampling Group Diagnostic Tags** represent how well the device is able to respond to requests.

Sample Group Diagnostic Tags

The table below describes the various diagnostic tags under the Aggregate and Sampling Group folders. Note that the structure for each type of folder is the same, save for several tags that are not available under the Aggregate folder; those exceptions are noted in the description column.

Tags in the **Aggregates** folder summarize all requests across all groups, whereas tags in sampling group folders represent metrics for that sample group. All metrics are based off the *start* of their group, unless specified otherwise. In the case of Aggregates, they're based off the last initialization of the device connection configuration on the host Ignition Gateway.

Tag	Description	Available in Aggregate Folder
ActualSamplingInterval	Represents the actual interval the sample group was able to last execute at, in milliseconds.	No
ActualThroughput	A count representing the actual number of requests that were able to execute during the last sample interval.	Yes
IdealSamplingInterval	The ideal sample interval for the sample group in milliseconds. Generally, this will simply be the configured sample interval.	No
IdealThroughput	A count representing the ideal number of requests handled over a 1000ms period. Calculated with the following equation: $\text{RequestCount} * 1000 / \text{SamplingInterval}$	Yes
OverloadFactor	Represents how well the device is able to keep up with requests at the sample rate. Overload is calculated with the formula: $100 * (\text{QueueDuration} / \text{ActualSamplingInterval})$ If overload exceeds 100%, then requests are being sampled at a slower than ideal rate.	No
QueueDuration	Represents the average amount of time a request has spent in the request queue.	No
RequestCount	The number of requests within the sampling group.	Yes

Execution Timer Tags

The tags in the table below represent how long the request process takes. Execution times are recorded in the following manner:

1. A timer starts
2. The request is sent off to the device

3. A response from the device is received by the OPC server

4. The timer stops

NOTE

All of the following tags are available in both sample group folders, as well as the Aggregates folder. Like the Sample Group Diagnostics all metrics are based of the *start* of their group, unless specified otherwise. In the case of Aggregates, they're based off the last initialization of the device configuration.

Tag(s)	Description
NthPercentile	N% of requests were able to execute at a number of milliseconds equal to or faster than the value on the tag. Example: If 75thPercentile shows a value of 50, then 75% of all requests were able to complete within 50 or fewer milliseconds.
Count	Represents the number of requests processed since the start of the sample group.
Max	The longest duration an execution took in the sampling group since the start of sampling. Reported in milliseconds.
Mean	The average duration for all executions in the sampling group since the start of sampling. Reported in milliseconds.
Min	The smallest duration an execution took to complete since the start of sampling. Reported in milliseconds.
MeanRate	An average of throughput, since the start of the sampling group. The MeanRate tag in the Aggregates folder represents an average since the last startup of the device connection.
OneMinuteRate	A rolling average of throughput over the last minute.
FiveMinuteRate	A rolling average of throughput over the last five minutes.

Allen Bradley Ethernet

Connecting to Allen-Bradley Devices

Ignition contains drivers for several Allen-Bradley devices. These drivers can connect directly through the Gateway to devices that support Ethernet communications and to devices that have access to an Ethernet card such as an ENBT or EN2T. The device drivers in the Allen-Bradley Ethernet Module are:

- **Logix** ControlLogix and CompactLogix firmware v20.19+
- **ControlLogix** firmware v20.18 and prior
- **CompactLogix** firmware v20.18 and prior
- **PLC-5**
- **Micro800** 820, 850, 870
- **MicroLogix** 1100, 1200, 1400, 1500
- **SLC** 5/05

NOTE

The Allen Bradley Ethernet drivers only support firmware versions 16 and up. Older firmware versions may work, but are unsupported. If your device does not meet these requirements, [another OPC Server](#) can be used to connect to them.

NOTE

Reading Add On Instructions (AOI) within Add On Instructions is not supported by Allen-Bradley. Tags with this architecture cannot be browsed in RSLogix and RSLinx.

Alias tag types can be used to read and write to members inside of AOIs that reside within another AOI.

CAUTION

Direct connections to RSLinx Emulate 5000 are not supported, and may not work properly.

Verifying Device Connectivity

Device connectivity can be verified in the following places:

- In the **Configure** page of the Gateway under **OPC-UA Server > Devices**
- In the **Status** page of the Gateway under **Overview > click the Devices** box
- In the **Status** page of the Gateway under **Connections > Devices**
- In the **Status** page of the Gateway under **Connections > OPC Connections**

Allen Bradley Connection Paths Explained

All of the Allen-Bradley drivers have a **Connection Path** property that is used when the device uses a special Ethernet card (instead of having an Ethernet connection built in) or if you are using another device to act as a bridge. That is, connections to ControlLogix, CompactLogix, PLC-5, MicroLogix and SLC Allen-Bradley processors bridged through a ControlLogix Gateway require a connection path. The connection path is unique to your setup and is dependent on what modules the connection is being routed through. With there being nearly an endless number of ways to route your connection from device to device it is impossible to give an example of every possible connection path, but in general there is a pattern to how the connection path is specified.

The following is a basic outline for figuring out your connection path. For more specific information on individual device types, see the individual connection pages.

Follow the Path

A connection path is exactly what it sounds like. It is a path that when followed will lead a processor residing in a numbered slot of a chassis somewhere on site. You merely have to follow the path and build the connection path as you go. The first connection point between Ignition and the device is a ControlLogix Ethernet module such as an ENET, ENBT or EN2T module. The slot number of this module doesn't matter and there is no need to specify it in the connection path. The first entry in any connection path will be a 1, which specifies moving to the back plane. You then specify the slot of the module you wish to move to, followed by the port or channel of that module that you wish to exit through. Finally you specify the address of your entry point to the next module and the process starts all over again. This process may sound complicated at first but after some practice it will get easier.

Connection Path Steps

1. Move to the backplane.
2. Specify the slot number of the module you are moving to.
3. Specify the exit port or channel.
4. Specify address of entry point (DH+ Station Number/ControlNet Address/IP Address of Ethernet module).
5. Move to the backplane.

6. Specify processor slot number OR the slot number of the module you wish to exit through.

Connection Path Entries for Different Module Types

How you specify your exit point from a module differs depending on which module type you are using. You can only move in two directions once you are "in" a module: out to the backplane or out through the module port/channel. Ethernet modules have Ethernet ports and an IP address; ControlNet modules have ControlNet Ports and ControlNet addresses; DHRIO modules have channels and station numbers. Below is a list of different kinds of modules and what numbers you specify in the connection path when you are exiting or entering those modules. When in a module, an entry of 1 will always take you to the backplane.

ENET, ENBT, and EN2T:

Exiting

1 = Backplane

2 = Ethernet Port

Entering

IP Address

CNB:

Exiting

1 = Backplane

2 = ControlNet Port

Entering

ControlNet Address

DHRIO

Exiting

1 = Backplane

2 = DH+ Channel A

3 = DH+ Channel B

Entering

DH+ Station Number (an octal value between 0-77)

You use these numbers to specify how to move out of the module, then you specify where you are moving to by either specifying the DH+ station number, ControlNet address, or the IP address of another Ethernet module. Your connection path will always be an even number of entries due to the fact that you always move in two steps: out of a module and then in to another module. So if your connection path ends up with an odd number of entries you have missed a step somewhere and you'll have to go back and trace the path again.

Connecting to CompactLogix

NOTE

This driver is made for CompactLogix firmware up to version 20.18.



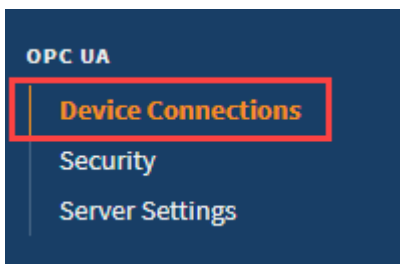
INDUCTIVE
UNIVERSITY

Connecting to Legacy
CompactLogix

[Watch the video](#)

Connect to an Allen-Bradley CompactLogix Device

1. Go to the **Config** section of the Gateway Webpage.
2. Scroll down and select **OPC UA > Device Connections**.



3. On the **Devices** page, find the blue arrow and click on **Create new Device**.
4. Select **Legacy Allen-Bradley CompactLogix**, and click **Next**.

⚙️ Config > Opcua > Devices

☒ **Allen-Bradley CompactLogix (Legacy)**
Connect to CompactLogix firmware v20 and prior processors.

☐ **Allen-Bradley ControlLogix (Legacy)**
Connect to ControlLogix firmware v20 and prior processors.

☐ **Allen-Bradley Logix Driver**
Connect to Allen-Bradley Logix family devices, including devices with firmware v21+.

☐ **Allen-Bradley MicroLogix**
Connect to MicroLogix 1100 and 1400 series PLCs.

☐ **Allen-Bradley PLC5**
Connect to PLC5s via Ethernet.

5. On the **New Device** page, leave all the default values and type in the following fields:

- Name: **CompactLogix**
- Hostname: type the **IP address** for the PLC, for example 10.20.4.55.

Config > Opcua > Devices

General

Name

CompactLogix

Description

Enabled

☒

(default: true)

Connectivity

Hostname

10.20.4.55

Timeout

2000

(default: 2,000)

Connection Path

(default:)

Concurrent Requests

2

(default: 2)

☐ Show advanced properties

Create New Device

6. Click **Create New Device**. The **Devices** page is displayed showing the **CompactLogix** device is added to Ignition. The **Status** will show as Disconnected and then Connected.

Config > Opcua > Devices

✓

Successfully created new Device "CompactLogix"

Name	Type	Description	Enabled	Status	
Sim	Simulator: Generic		true	Running	delete edit
CompactLogix	Allen-Bradley CompactLogix (Legacy)		true	Connected: Protocol: EIP - Run Mode	More ▼ edit
Sim	Simulator: Generic		true	Running	delete edit

7. To see all the Tags, go to **OPC Client > Quick Client** in the **Config** section. On the **OPC Quick Client** page, expand the **CompactLogix** folder and in the **Global** folder you can see all the tags.

Device Connection Settings

The **General** settings are common to all Allen Bradley devices, and the **Connectivity** settings are device dependent.

General Settings

Name	Description
Name	The user-defined name for this Device. The Device Name must be alphanumeric.
Description	The user-defined description for the device. This is only used as a note to differentiate between devices.
Enable Device	Only devices that are enabled appear in Connections > Devices of the Status page of the Gateway and thus have their Tags available for use. Default is true.

Connectivity Settings

Name	Description
Hostname	This is the IP Address of the Ethernet module to route through to connect a CompactLogix processor.
Local Address	New in 8.1.8 The local address to connect from when establishing a TCP connection. If left blank, then the driver will simply pick an available address.
Timeout	The timeout settings refer to the communication between the device driver and the OPC-UA server and usually can be left at their default values.
Connection Path	The Connection Path value is used to define the route to connect to the processor. The Connection Path format contains 4 numbers separated by commas. The first number is always 1 and tells the Ethernet module to route through the backplane. The rest of the numbers vary by device type, for a more in depth explanation of connection paths, see Allen Bradley Connection Paths Explained section.
Concurrent Requests	The number of requests that Ignition will try to send to the device at the same time. Increasing this number can sometimes help with your request throughput, however increasing this too much can overwhelm the device and hurt communications with the device.

Advanced

Name	Description
Disable Automatic Browse	Disables the automatic browse setting. Default is false.
Show String Arrays	Disables the Show String Arrays setting. Default is false.
Status Request Poll Rate	Controls the poll rate for status requests, in milliseconds. Default is 1,000.

Connecting to ControlLogix

NOTE

This driver is made for ControlLogix firmware up to version 20.18.



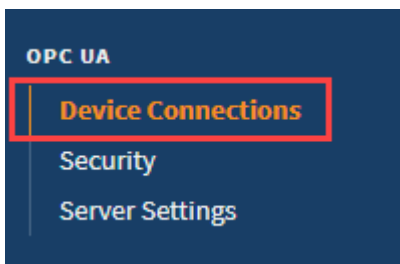
INDUCTIVE
UNIVERSITY

Connecting to Legacy
ControlLogix

[Watch the video](#)

Connect to an Allen-Bradley ControlLogix Device

1. Go to the **Config** section of the Gateway webpage.
2. Scroll down and select **OPC UA > Device Connections**.



3. On the **Devices** page, find the blue arrow and click on **Create new Device**.
4. Select **Legacy Allen-Bradley ControlLogix**, and click **Next**.

Config > Opcua > Devices

☐ **Allen-Bradley CompactLogix (Legacy)**
Connect to CompactLogix firmware v20 and prior processors.

☒ **Allen-Bradley ControlLogix (Legacy)**
Connect to ControlLogix firmware v20 and prior processors.

☐ **Allen-Bradley Logix Driver**
Connect to Allen-Bradley Logix family devices, including devices with firmware v21+.

5. On the **Devices** page, leave all the default values and type in the following fields:

- Name: **CLX**
- Hostname: Enter the **IP address** for the PLC, for example 10.20.4.50.

Config > Opcua > Devices

General	
Name	CLX 
Description	
Enabled	☒ (default: true)

Connectivity	
Hostname	10.20.4.50
Timeout	2000 (default: 2,000)
Slot Number	0 (default: 0)
Connection Path	 (default:)
Concurrent Requests	2 (default: 2)

6. Click **Create New Device**. The **Devices** page is displayed showing the **ControlLogix** device is added to Ignition. The **Status** will show as Disconnected and then Connected.

Config > Opcua > Devices

✔ Successfully created new Device "CLX"

Name	Type	Description	Enabled	Status		
CLX	Allen-Bradley ControlLogix (Legacy)		true	Disconnected: Protocol: EIP - Program Mode	More ▾	edit
Sim	Simulator: Generic		true	Running	delete	edit
PLC5	Allen-Bradley PLC5		true	Connected: Protocol: Direct	More ▾	edit

7. To see all the tags, go to **OPC Client > Quick Client** in the **Config** section. On the **OPC Quick Client** page, expand the **ControlLogix** folder and in the **Global** folder you can see all the tags.

Device Connection Settings

The **General** settings are common to all Allen Bradley devices, and the **Connectivity** settings are device dependent.

General

Name	Description
Name	The user-defined name for this Device. The name chosen will show up in OPC Item Paths and under OPC-UA Server > Devices of the Configure page of the Gateway. The Device Name must be alphanumeric.
Description	The user-defined description for the device. This is only used as a note to differentiate between devices.
Enable Device	Only devices that are enabled appear in Connections > Devices of the Status page of the Gateway and thus have their tags available for use.

Connectivity

Name	Description
Hostname	This is the IP Address of the ControlLogix Ethernet module (1756-ENET) to route through to connect a ControlLogix processor. EthernetIP protocol on TCP port 44818 (0xAF12) is used to communicate to ControlLogix processors.
Local Address	New in 8.1.8 The local address to connect from when establishing a TCP connection. If left blank, then the driver will simply pick an available address.
Timeout	After sending a request to the ControlLogix processor, the Communication Timeout setting is the amount of time in msec to wait for a response before treating it as a failure.
Slot Number	The Slot Number value is the zero based ControlLogix chassis slot number of the ControlLogix processor to connect to.
Connection Path	The Connection Path value is used to define the route of the PLC-5 processor to connect to. Currently routing through the ControlLogix Ethernet Communication Interface Module (1756-ENET) to the ControlLogix Data Highway Plus-Remote I/O Communication Interface Module (1756-DHRIO) and on to a PLC-5 processor of the DH+ network is supported.
Concurrent Requests	The number of requests that Ignition will try to send to the device at the same time. Increasing this number can sometimes help with your request throughput, however increasing this too much can overwhelm the device and hurt your communications with the device.

Advanced

Name	Description
Disable Automatic Browse	Disables the automatic browse setting. (Default is false.)
Show String Arrays	Disables the Show String Arrays setting. (Default is false.)
Status Request Poll Rate	Controls the poll rate for status requests, in milliseconds. (Default is 1,000.)

Supported Connection Methods

ControlLogix 5500 connected through 1756-ENET/A or 1756-ENET/B.

Considerations for ControlLogix Settings

When the ControlLogix processor **External Access** setting for a tag is set to **None**, the tag information is not returned to Ignition during reading of tag information.

Using RSLogix 5000, change the External Access setting to either **Read Only** or **Read/Write** for the tag to show in the Ignition browse results.

 Tag Properties - AAA_ADR_ARRAY1D

General

Name: AAA_ADR_ARRAY1D

Description:

Type: Base Connection...

Alias For:

Data Type: ADRPowerData[5]

Scope:  ADR_CACK_Bldg3

External Access: Read/Write

Style: Read/Write
Read Only
None

☐ Constant

OK Cancel Apply Help

Connecting to Logix

NOTE

This driver is optimized for Logix family devices with firmware version 21+, but supports earlier firmware versions with significantly reduced performance.



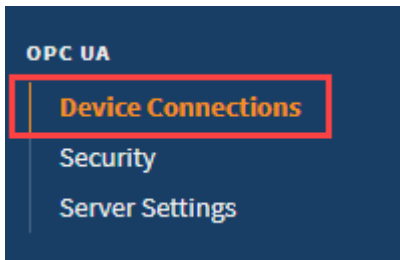
INDUCTIVE
UNIVERSITY

Connecting to ControlLogix
v21

[Watch the video](#)

Connect to an Allen-Bradley Logix Device

1. Go to the **Config** section of the Gateway Webpage.
2. Scroll down and select **OPC UA > Device Connections**.



3. On the **Devices** page, find the blue arrow and click on **Create new Device**.
4. Select **Allen-Bradley Logix Driver**, and click **Next**.

⚙️ Config > Opcua > Devices

☐ **Allen-Bradley CompactLogix (Legacy)**
Connect to CompactLogix firmware v20 and prior processors.

☐ **Allen-Bradley ControlLogix (Legacy)**
Connect to ControlLogix firmware v20 and prior processors.

☒ **Allen-Bradley Logix Driver**
Connect to Allen-Bradley Logix family devices, including devices with firmware v21+.

☐ **Allen-Bradley MicroLogix**
Connect to MicroLogix 1100 and 1400 series PLCs.

5. Fill in the following fields:

- **Name:** CLX
- **Hostname:** type the **IP address** for the PLC, for example 10.20.1.54.

Leave the default values in the remaining fields.

Config > Opcua > Devices

General

Name	CLX 
Description	
Enabled	☒ (default: true)

Connectivity

Hostname	10.20.X.XX Hostname or IP address of the controller or ethernet module.
Port	44818 Port to connect to on the remote device. (default: 44,818)
Timeout	2000 Max amount of time to wait, in milliseconds, for responses from the processor. (default: 2,000)
Max Concurrent Requests	2 Controls both the max number of concurrent unconnected requests and the max number of concurrent connected requests (and therefore CIP connections) allowed. (default: 2)
Slot Number	0 The slot number in which the processor resides. If not in a rack, usually 0. (default: 0)
Connection Path	 Optionally, the connection path to the processor. If specified, the slot number setting is ignored, so be sure to terminate the path with backplane and slot number. (default:)

6. Click **Create New Device**.

7. The **Devices** page is displayed showing the **ControlLogix** device is added to Ignition. The **Status** will show as Disconnected and then Connected.

Config > Opcua > Devices

 Successfully created new Device "CLX"

Name	Type	Description	Enabled	Status	
CLX	Allen-Bradley Logix Driver		true	ReconnectWait	delete edit
Sim_New_Programmable	Programmable Device Simulator		true	Running	More ▾ edit

8. To see all the Tags, go to **OPC Client > Quick Client** in the **Config** section. On the **OPC Quick Client** page, expand the **CompactLogix** folder and in the **Global** folder you can see all the tags.

Driver Implementation

One noteworthy implementation of the Logix driver is how boolean arrays are handled. Boolean arrays items are browsed as members of a 32-bit DWORD. Thus, a BOOL[64] in the PLC is implemented as DWORD[2] in the driver.

Below is a table representing how members of a boolean array items are mapped in the PLC, compared to the Logix driver's implementation.

PLC Mapping	Driver Implementation
boolTag[0]	boolTag[0].0
boolTag[31]	boolTag[0].31
boolTag[32]	boolTag[1].0
boolTag[63]	boolTag[1].31

New in 8.1.28 The Logix driver also supports the following datatypes:

- DT (Date/Time)
- LDT (Date/Time [ns])
- LTIME (LTime [ns])
- TIME (Decimal)
- TIME32 (Time32 [us])

Device Connection Settings

The General settings are common to all Allen Bradley devices, and the Connectivity settings are device dependent.

General

Name	Description
Name	The user-defined name for this Device. The name chosen will show up in OPC Item Paths and under OPC-UA Server > Devices of the Configure page of the Gateway. The Device Name must be alphanumeric.
Description	The user-defined description for the device. This is only used as a note to differentiate between devices.
Enable Device	Only devices that are enabled appear in Connections > Devices of the Status page of the Gateway and thus have their tags available for use.

Connectivity

Name	Description
Hostname	This is the IP Address of the ControlLogix Ethernet module (1756-ENET) to route through to connect a ControlLogix processor. EthernetIP protocol on TCP port 44818 (0xAF12) is used to communicate to ControlLogix processors.
Port	Port to connect to the remote device.
Local Address	New in 8.1.8 The local address to connect from when establishing a TCP connection. If left blank, then the driver will simply pick an available address.
Timeout	After sending a request to the ControlLogix processor, the Communication Timeout setting is the amount of time in msec to wait for a response before treating it as a failure.
Max Concurrent Requests	The number of requests that Ignition will try to send to the device at the same time. Increasing this number can sometimes help with your request throughput, however increasing this too much can overwhelm the device and hurt your communications with the device.
Slot Number	The Slot Number value is the zero based ControlLogix chassis slot number of the ControlLogix processor to connect to.
Connection Path	If specified, the slot number setting is ignored for this connection path to the processor. Be sure to terminate the path with backplane and slot number.

Advanced

Name	Description
Automatic Rebrowse	Monitor for tag additions and UDT changes and automatically initiate a re-browse when detected. If this is disabled tags will only be browsed when connecting and reconnecting. (Default is true.)
Identity Request Frequency	New in 8.1.3 Frequency, in milliseconds, that the request to read CIP Identity Object attributes occurs at. (Default is 5,000.)
CIP Connection Size	The CIP connection size to use during Forward Open requests. (Default is 500.)
CIP Connection Timeout	New in 8.1.10 Target timeout, in milliseconds, for CIP connections and RPI. (Default is 16,000)

Connecting to Micro800

New in 8.1.39

The Micro800 driver supports communication with the following Allen-Bradley Micro800 PLCs via EtherNet/IP over TCP/IP:

- **Micro820**
- **Micro850**
- **Micro870**



INDUCTIVE
UNIVERSITY

Connecting to Micro800

[Watch the video](#)

Data Types

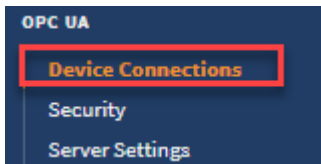
The driver supports the following data types:

- DATE (Milliseconds since Unix epoch)
- TIME (Represents duration in milliseconds)
- BOOL
- SINT
- INT
- DINT
- LINT
- USINT
- UINT
- UDINT
- ULINT
- REAL
- LREAL
- STRING (UTF-8 encoding, max limit of 80 characters)
- BYTE
- WORD

- DWORD
- LWORD

Connect to a Micro800 Device

1. Navigate to the **Config** section of the Gateway Webpage.
2. Scroll down and select **OPC UA > Device Connections**.



3. On the **Devices** page, click on **Create new Device**.
4. Select **Allen-Bradley Micro800 Driver**, and click **Next**.
5. On the New Device page, leave all the default values and fill in the following fields:
 - **Name:** Micro800
 - **Hostname:** Enter **IP address** for the device.

A screenshot of a configuration form. The 'General' section has a 'Name' field with the value 'Micro800'. The 'Connectivity' section has a 'Hostname' field with a placeholder IP address '192.168.1.1' and a tooltip that says 'Hostname or IP address of the controller or ethernet module.'

6. Click **Create New Device**.
7. The **Devices** page is displayed showing the **Micro800** device was added to Ignition. The **Status** will show as Disconnected and then Connected.

Successfully created new Device "Micro800"				
Name	Type	Description	Enabled	Status
Micro800	Allen-Bradley Micro800 Driver		true	Connected
				More ▼ edit

8. To see all the tags, go to **OPC Client > Quick Client** in the **Config** section. On the OPC Quick Client page expand the Micro800 device folder to view browsable tags.

i NOTE

Not all tags will be visible in the Quick Client. See the [Browsing](#) section for more info.

Device Connection Settings

General

General	Description
Name	The device name.
Description	The device description.
Enabled	If true, attempts connection to the device. Default is true.

Connectivity

Connectivity	Description
Hostname	Hostname or IP address of the controller or ethernet module.
Port	Port to connect to on the remote device.
Local Address	Address of network adapter to connect from.
Timeout	Max amount of time to wait, in milliseconds, for responses from the processor.
Max Concurrent Requests	Controls both the max number of concurrent unconnected requests and the max number of concurrent connected requests (and therefore CIP connections) allowed.

Advanced

Advanced	Description
Write Priority Ratio	The number of write requests to execute for every read request when an abundance of both types are queued up for execution.
Automatic Rebrowse	Monitors for tag changes and automatically initiates a re-browse when detected. If this is disabled, tags will only be browsed when connecting and reconnecting.
CIP Connection Size	The CIP connection size to use during Forward Open requests.
CIP Connection Timeout	Target timeout, in milliseconds, for CIP connections and RPI.

Browsing

The driver supports browsing global variables. However, not all global variables are exposed by the PLC via browse request, including:

- UDTs and system-defined structures
- BOOL data type arrays
- Arrays with non-zero starting indices

Despite this limitation, these global variables can still be made browsable in Ignition using the supplementary Addresses page Load Configuration File option. This page allows users to import global variables from a CSV file or .zip file downloaded from Connected Components Workbench.

 NOTE

The Micro800 driver is designed to work with individual variables exported from the Connected Components Workbench and cannot directly read Function Blocks.

 DEVICE CONNECTIONS SUPPLEMENTARY ADDRESSES PAGE

To prevent conflicting definitions with tags that can be browsed, only global variables that are not exposed via browsing the device can be specified on this page. This is determined based on the variable's name, data type, dimensions, etc. For CSV files, any global variable not meeting this condition will be automatically filtered out on import, and for manual entry, filtering will occur on

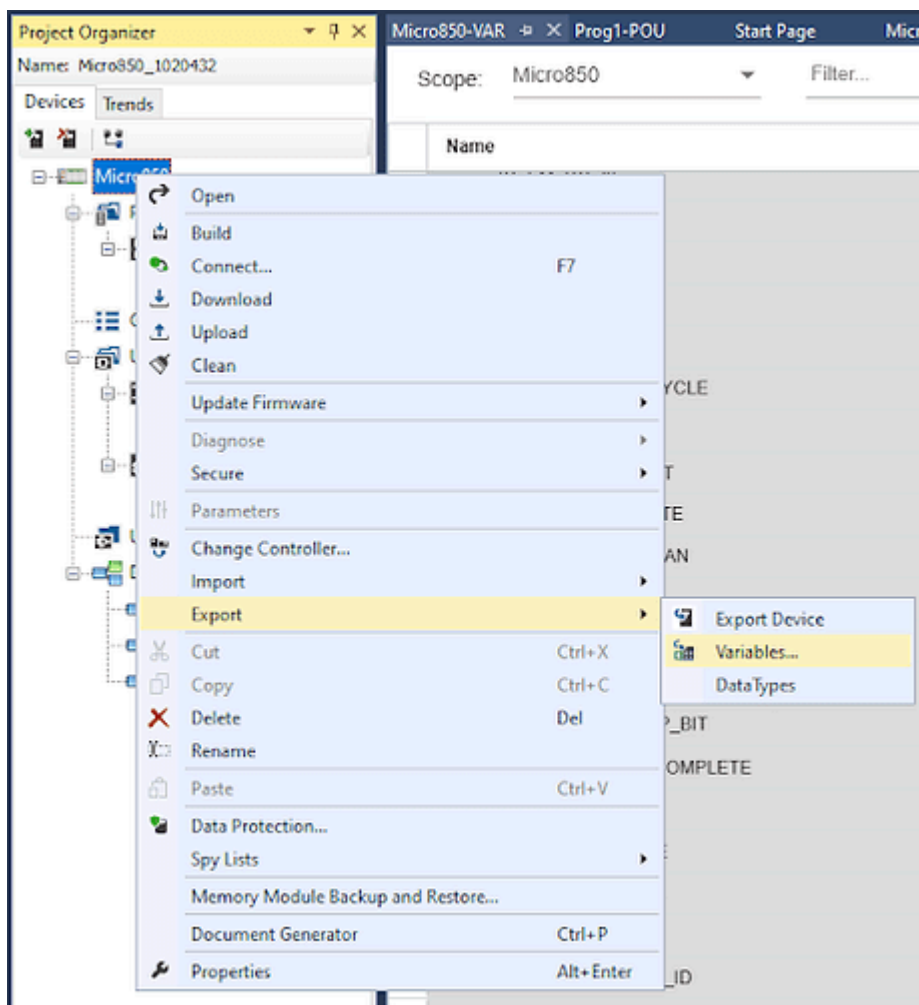
save. The

`com.inductiveautomation.ignition.drivers.micro800.util.tags.GlobalVariableCsvParser` logger can be set to TRACE to determine what was filtered out.

Importing Global Variables - CSV File

Downloading the CSV file from Connected Components Workbench

1. Open **Connected Components Workbench** and load the Micro800 project.
2. Right-click on the controller in the Project Organizer pane and select **Export > Variables**.



3. Select a suitable download location and change the file format to **Comma Separated Values**.
4. Click **Save**.

Loading the CSV file into Ignition

1. Navigate to the **Config** section of the Gateway webpage.
2. Select **OPC UA > Device Connections**.

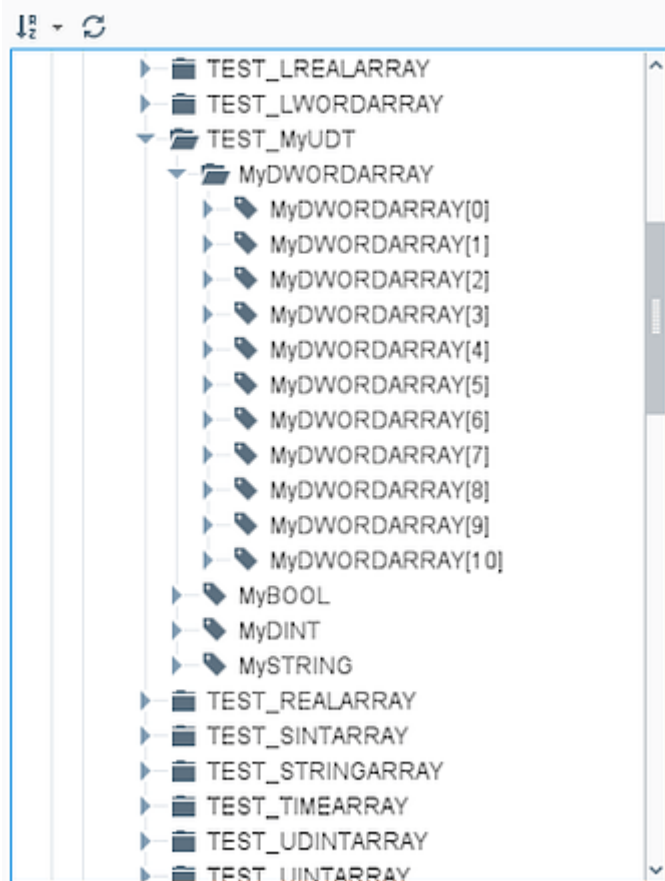
3. Locate the Micro800 device connection and click the **More** dropdown and select **Addresses**.
4. Click on **Load Configuration File** and choose the CSV file obtained from Connected Components Workbench.
5. Click **Load**. Entries for variables that can be imported should populate the page.

Device Connections / micro850

Load Configuration File 					  Remove All	
Browse Name	Data Type	Dimension	String Size		Read/Write	
TEST_MyUDT.MyBOOL	BOOL				RW	Remove
TEST_MyUDT.MyDINT	DINT				RW	Remove
TEST_MyUDT.MySTRING	STRING		80		RW	Remove
TEST_MyUDT.MyDWORDARRAY	DWORD	[0..10]			RW	Remove
NewVariable.IN	BOOL				RW	Remove
NewVariable.ABInput.Channel	UINT				RW	Remove
NewVariable.ABInput.TriggerType	USINT				RW	Remove
NewVariable.ABInput.Channel	BOOL				RW	Remove

6. Click **Save Configuration**. These variables should now be visible as nodes in the OPC Browser or Quick Client.

OPC Browser

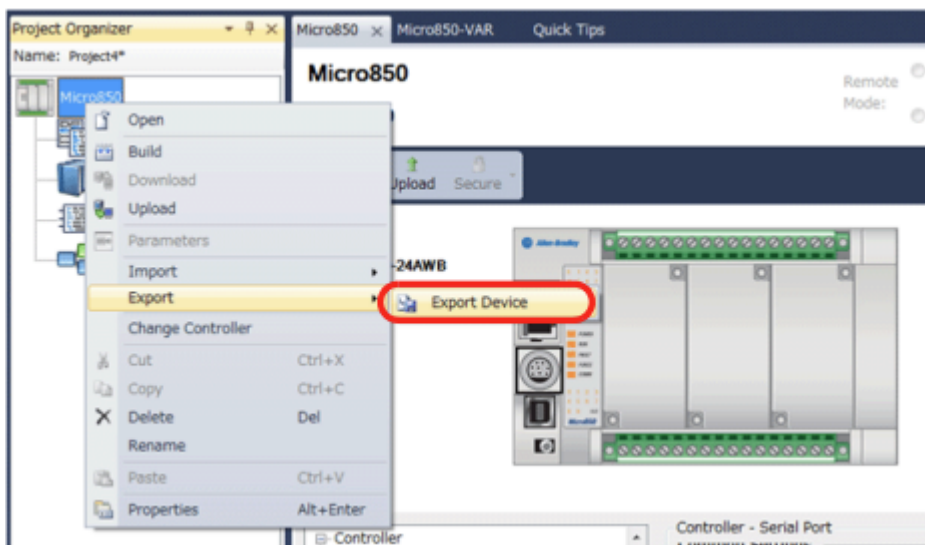


Importing Global Variables - Zip File

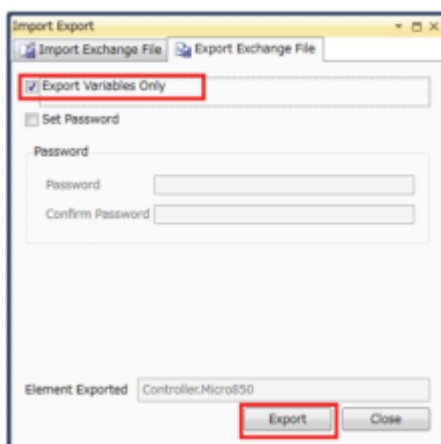
New in 8.1.40 In addition to importing with a CSV file, you can now import global variables that are not exposed via browsing with a .zip file.

Exporting Tags from Connected Components Workbench

1. Open **Connected Components Workbench** and load the Micro800 project.
2. Right-click on the controller in the Project Organizer pane.
3. Select **Export > Export Device** from the context menu.



4. In the **Import Export** popup window, select the **Export Variables Only** checkbox.
5. Click on **Export**.



6. The exported file will be in .7z format. Convert it to a .zip file using a file archiver tool.

Importing Tags into Ignition

1. Navigate to the **Config** section of the Gateway webpage.

- 2. Select **OPC UA > Device Connections**.
- 3. Click on the **More** option for the Micro800 device.
- 4. Click on **Load Configuration File** to import the .zip file.

Addressing

In most cases, you shouldn't need to worry about OPC Item Paths as they're provided through the device browse or file import. The only exception is if global variables are manually being added to the Addresses page (not recommended). In which case, the rules are:

- Paths must match with the global variable names in Connected Components Workbench.
- Paths must resolve to an atomic data type (or an array of an atomic data type), i.e. keep "drilling" down until it's not a structure.

For example, consider this UDT in Connected Components Workbench:

Name	Alias	Data Type	Dimension	Project Value	Initial Value	Comment	String Size
MyUdtInstance		MyUdt		...	...		
MyUdtInstance.NestedUdtMember		NestedUdt		...	...		
MyUdtInstance.NestedUdtMember.BoolMember		BOOL					
MyUdtInstance.NestedUdtMember.DintMember		DINT					
MyUdtInstance.NestedUdtMember.StringMember		STRING					80
MyUdtInstance.NestedUdtMember.FurtherNestedUdtMember		FurtherNestedUdt		...	...		
MyUdtInstance.NestedUdtMember.FurtherNestedUdtMember.LintMember		LINT					
MyUdtInstance.DwordArrayMember		DWORDARRAY		...	...		
MyUdtInstance.DwordArrayMember[1]		DWORD					
MyUdtInstance.DwordArrayMember[2]		DWORD					
MyUdtInstance.DwordArrayMember[3]		DWORD					

To manually expose this UDT as a browsable node in OPC Browser or Quick Client, the paths need to be specified in the Addresses page like so:

Browse Name	Data Type	Dimension	String Size	ReadWrite
MyUdtInstance.NestedUdtMember.BoolMer	BOOL			RW Remove
MyUdtInstance.NestedUdtMember.DintMen	DINT			RW Remove
MyUdtInstance.NestedUdtMember.StringMi	STRING		80	RW Remove
MyUdtInstance.NestedUdtMember.Furtherf	LINT			RW Remove
MyUdtInstance.DwordArrayMember	DWORD	[1..3]		RW Remove

Bit-Level Addressing

For bit-leveling addressing, append a period and the bit index to the OPC Item Path. This is applicable for all numerical data types. The bit index must be within range of the addressed variable's data type.

Examples:

- `MyUint.1` (First bit of MyUint)
- `MyByteArray[0].7` (7th bit of the 0th array element inside MyByteArray)
- `MyUdtInstance.MyDint.11` (11th bit of the member tag, MyDint, inside the UDT, MyUdtInstance)

Performance Considerations

Due to limitations with how the Micro800 series implements the communication protocol, there are limited opportunities for tag read/write optimizations, as implemented in other drivers. For this reason, subscriptions containing unused tags should be avoided whenever possible as well as subscriptions with unnecessarily fast rates. The **Max Concurrent Requests** setting can be adjusted to allow the Micro800 Driver to send multiple requests at the same time which can greatly improve throughput. However be wary that increasing this value too high may yield diminishing returns or negatively impact the performance of the PLC.

Connecting to MicroLogix



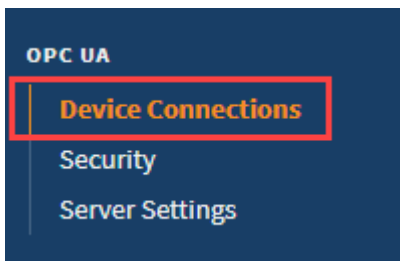
INDUCTIVE
UNIVERSITY

Connecting to MicroLogix

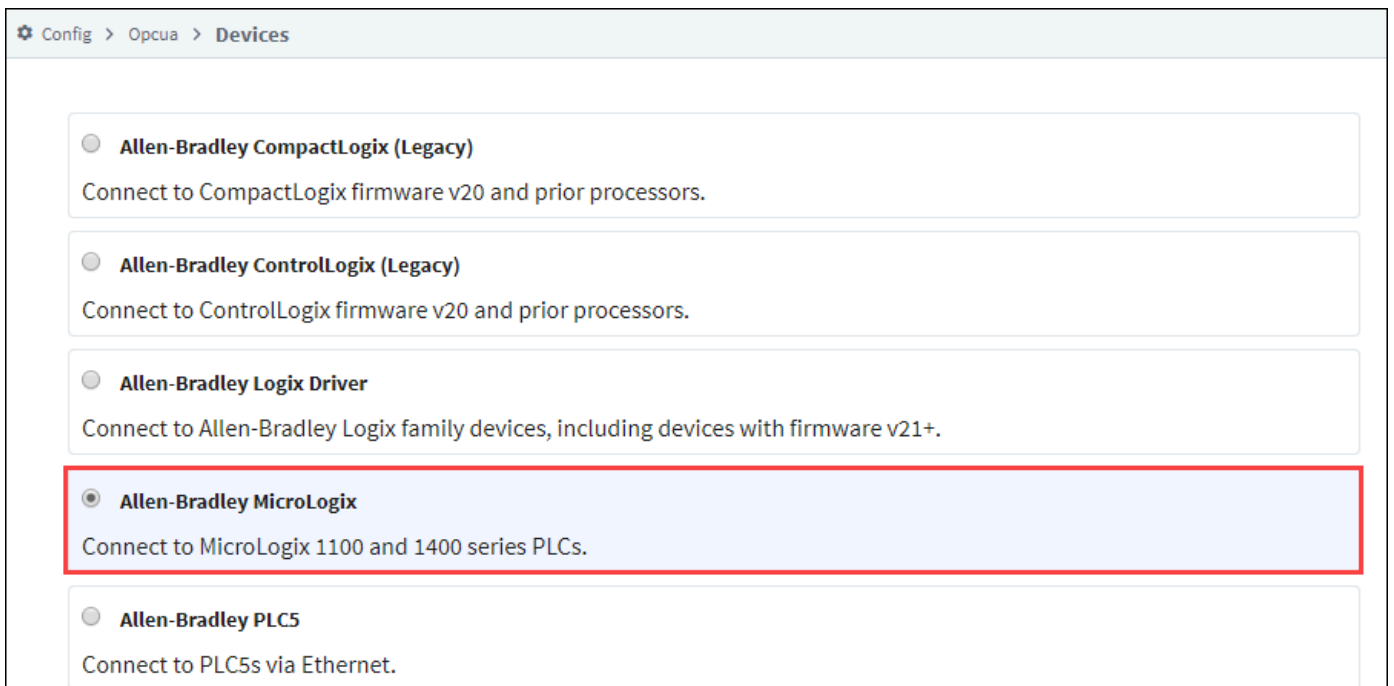
[Watch the video](#)

Connect to a MicroLogix Device

1. Go to the **Config** section of the Gateway Webpage.
2. Scroll down and select **OPC UA > Device Connections**.



3. On the **Devices** page, find the blue arrow and click on **Create new Device**.
4. Select **Allen-Bradley MicroLogix**, and click **Next**.



5. On the New Device page, leave all the default values and type in the following fields:

- **Name:** MLX
- **Hostname:** Enter **IP address** for the PLC, for example 10.20.7.77

Config > Opcua > Devices

General

Name

MLX

Description

Enabled

☒
(default: true)

Connectivity

Hostname

10.20.7.77

Timeout

2000
(default: 2,000)

Browse Cache Timeout

240000
(default: 240,000)

Connection Path

☐ Show advanced properties

Create New Device

6. Click **Create New Device**.

7. The **Devices** page is displayed showing the **MicroLogix** device is added to Ignition. The **Status** will show as Disconnected and then Connected.

Config > Opcua > Devices

✓ Successfully created new Device "MLX"

Name	Type	Description	Enabled	Status	
Sim	Simulator: Generic		true	Running	deleteedit
MLX	Allen-Bradley MicroLogix		true	Connected: Protocol: EIP	More▼edit

8. To see all the tags, go to **OPC Client > Quick Client** in the **Config** section. On the **OPC Quick Client** page expand the **MicroLogix** folder which contains all the folders with the individual tags.

Device Connection Settings

The **General** settings are common to all Allen-Bradley devices, and the **Connectivity** settings are device dependent.

General Settings

Name	Description
Name	The user-defined name for this Device. The name chosen will show up in OPC Item Paths and under OPC-UA Server > Devices of the Configure page of the Gateway. The Device Name must be alphanumeric.
Description	The user-defined description for the device. This is only used as a note to differentiate between devices.
Enable Device	Only devices that are enabled appear in Connections > Devices of the Status page of the Gateway and thus have their tags available for use.

Connectivity Settings

Name	Description
Hostname	The Hostname value is the IP Address of the MicroLogix 1100 processor, MicroLogix 1400 processor or 1761-NET-ENI Ethernet interface. EthernetIP protocol on TCP port 44818 (0xAF12) is used to communicate to the listed devices.
Local Address	New in 8.1.8 The local address to connect from when establishing a TCP connection. If left blank, then the driver will simply pick an available address.
Communication Timeout	After sending a request to the MicroLogix processor, the Communication Timeout setting is the amount of time in msec to wait for a response before treating it as a failure.
Browse Cache Timeout	When the data table layout is read from the MicroLogix processor, the Browse Cache Timeout value is the amount of time in msec to cache the results.

Advanced Settings

Name	Description
Disable Processor Browse	Disables the processor browse setting. (Default is false.)
Zero TNS Connection	Disables the Zero TNS connection setting. (Default is false.)

Supported MicroLogix Connection Methods

- MicroLogix 1100 and 1400 direct
- MicroLogix 1100 and 1400 connected through 1761-NET-ENI
- MicroLogix 1100/1400 connected through Spectrum Controls WebPort 500

 CAUTION

MicroLogix 1200 and 1500 are not fully supported. Browsing is not available on these devices, so the 'Disable Processor Browse' advanced property will need to be set to True on the device

connection.

Additionally, the MicroLogix driver can not access Input Parameters.

** NOTE**

Some Micrologix devices may get stuck in a 'Browse Pending' status. In this case setting the connection path property to 1,0 should resolve the issue.

Connecting to PLC5



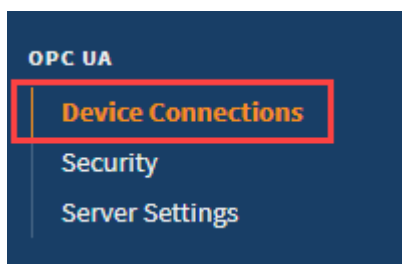
INDUCTIVE
UNIVERSITY

Connecting to PLC5

[Watch the video](#)

Connect to a Device

1. Go to the **Config** section of the Gateway Webpage.
2. Scroll down and select **OPC UA > Device Connections**.



3. On the **Devices** page, click on **Create new Device**.
4. Select **Allen-Bradley PLC5**, and click **Next**.

⚙️ Config > Opcua > Devices

☐ **Allen-Bradley CompactLogix (Legacy)**
Connect to CompactLogix firmware v20 and prior processors.

☐ **Allen-Bradley ControlLogix (Legacy)**
Connect to ControlLogix firmware v20 and prior processors.

☐ **Allen-Bradley Logix Driver**
Connect to Allen-Bradley Logix family devices, including devices with firmware v21+.

☐ **Allen-Bradley MicroLogix**
Connect to MicroLogix 1100 and 1400 series PLCs.

☒ **Allen-Bradley PLC5**
Connect to PLC5s via Ethernet.

☐ **Allen-Bradley SLC**
Connect to SLC 5/05s via Ethernet.

5. Fill in the following fields:

- **Name:** PLC5
- **Hostname:** type the **IP address** for the PLC, for example 10.20.4.56.

Leave the default values for the remaining fields.

Config > Opcua > **Devices**

General

Name	PLC5 
Description	
Enabled	☒ (default: true)

Connectivity

Hostname	10.20.4.56
Timeout	2000 (default: 2,000)
Browse Cache Timeout	240000 (default: 240,000)
Connection Path	

☐ Show advanced properties

Create New Device

6. Click **Create New Device**.

7. The **Devices** page is displayed showing the **PLC5** device is added to Ignition. The **Status** will show as Disconnected and then Connected.

Config > Opcua > **Devices**

 Successfully created new Device "PLC5 "

Name	Type	Description	Enabled	Status	
Sim	Simulator: Generic		true	Running	delete edit
PLC5	Allen-Bradley PLC5		true	Connected: Protocol: Direct	More ▾ edit

8. To see all the tags, go to **OPC Client > Quick Client** in the **Config** section. On the **OPC Quick Client** page, expand the **PLC5** folder and in the **Global** folder you can see all the tags.

Device Connection Settings

The **General** settings are common to all Allen Bradley devices, and the **Connectivity** settings are device dependent.

General Settings

Name	Description
Name	The user-defined name for this Device. The name chosen will show up in OPC Item Paths and under OPC-UA Server > Devices of the Configure page of the Gateway. The Device Name must be alphanumeric.
Description	The user-defined description for the device. This is only used as a note to differentiate between devices.
Enable Device	Only devices that are enabled appear in Connections > Devices of the Status page of the Gateway and thus have their tags available for use.

Connectivity Settings

Name	Description
Hostname	The Hostname value is the IP Address of the PLC-5 processor. The protocol that the PLC-5 processor supports is automatically detected. It will use either CSP protocol on port 2222 (0x8AE) or EthernetIP protocol on port 44818 (0xAF12).
Local Address	New in 8.1.8 The local address to connect from when establishing a TCP connection. If left blank, then the driver will simply pick an available address.
Communication Timeout	After sending a request to the PLC-5 processor, the Communication Timeout setting is the amount of time in milliseconds to wait for a response before treating it as a failure.
Browse Cache Timeout	When the data table layout is read from the PLC-5 processor, the Browse Cache Timeout value is the amount of time in milliseconds to cache the results.
Connection Path	The Connection Path value is used to define the route of the PLC-5 processor to connect to. Currently routing through the ControlLogix Ethernet Communication Interface Module (1756-ENET) to the ControlLogix Data Highway Plus-Remote I/O Communication Interface Module (1756-DHRIO) and on to a PLC-5 processor of the DH+ network is supported.

Advanced

Name	Description
Disable Processor Browse	Disables the processor browse setting. (Default is false.)
Zero TNS Connection	Disables the Zero TNS connection setting. (Default is false.)

More Information On Connection Path

The Connection Path format contains 4 numbers separated by commas. The first number is always 1 and tells the 1756-ENET module to route through the backplane. The second number is the slot number of the 1756-DHRIO module of the DH+ network the PLC-5 processor is connected to. The third number is the channel of the 1756-DHRIO module that the PLC-5 processor is connected to. Use 2 for channel A and 3 for channel B. The final and fourth number is the DH+ node number. This number is in octal and is the same as configured in the PLC-5 processor. See the **ControlLogix Ethernet Communication interface Module** User Manual for more information.

- Connection Path Format: 1,<1756-DHRIO slot number>,<1756-DHRIO channel>,<DH+ node number>

The valid range for the 1756-DHRIO slot number is between 0 and 16 but depends on the chassis size. The 1756-DHRIO channel is either 2 for channel A or 3 for channel B. The DH+ node number range is from 00 to 77 octal.

Supported PLC-5 Connection Methods

- PLC-5 L/20E, L/40E, L/80E direct
- All PLC-5 processors connected through DH+ via the 1756-DHRIO module.

NOTE

ASCII data types from a PLC5 are not supported by the Allen-Bradley PLC5 driver.

Connecting to SLC



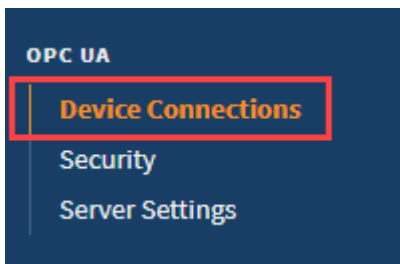
INDUCTIVE
UNIVERSITY

Connecting to SLC

[Watch the video](#)

Connect to a Device

1. Go to the **Config** section of the Gateway Webpage.
2. Scroll down and select **OPC UA > Device Connections**.



3. On the **Devices** page, click on **Create new Device**.
4. Select **Allen-Bradley SLC**, and click **Next**.

NOTE

These PLCs do not have a native Ethernet connection, therefore another device like a Net ENI or an ENBT must be used for the connection.

⚙️ Config > Opcua > Devices

☐ **Allen-Bradley MicroLogix**
Connect to MicroLogix 1100 and 1400 series PLCs.

☐ **Allen-Bradley PLC5**
Connect to PLC5s via Ethernet.

☒ **Allen-Bradley SLC**
Connect to SLC 5/05s via Ethernet.

☐ **DNP3 Driver**
Connect to a DNP3 outstation.

☐ **Modbus RTU over TCP**
Connect to devices that implement the Modbus RTU protocol over TCP.

☐ **Modbus TCP**
Connect to devices that implement the Modbus TCP protocol.

5. Fill in the following fields:

- **Name:** SLC
- **Hostname:** **IP address** for the PLC, for example 10.20.4.56.

Leave the default values in the remaining fields.

Config > Opcua > Devices

General

Name	SLC 
Description	
Enabled	☒ (default: true)

Connectivity

Hostname	10.20.4.56
Timeout	2000 (default: 2,000)
Browse Cache Timeout	240000 (default: 240,000)
Connection Path	

6. Click **Create New Device**. The **Devices** page is displayed showing the **SLC** device is added to Ignition. The **Status** will show as Disconnected and then Connected.

Config > Opcua > Devices

 **Successfully created new Device "SLC"**

Name	Type	Description	Enabled	Status	
Sim	Simulator: Generic		true	Running	delete edit
SLC	Allen-Bradley SLC		true	Connecting: Browse pending...	More ▾ edit

7. To see all the tags, go to **OPC Client > Quick Client** in the **Config** section. On the **OPC Quick Client** page, expand the **SLC** folder and in the **Global** folder you can see all the tags.

Device Connection Settings

The **General** settings are common to all Allen Bradley devices, and the **Connectivity** settings are device dependent.

General Settings

Name	Description
Name	The user-defined name for this Device. The name chosen will show up in OPC Item Paths and under OPC-UA Server > Devices of the Configure page of the Gateway. The Device Name must be alphanumeric.
Description	TThe user-defined description for the device. This is only used as a note to differentiate between devices.
Enable Device	Only devices that are enabled appear in Connections > Devices of the Status page of the Gateway and thus have their tags available for use.

Connectivity Settings

Name	Description
Hostname	The Hostname value is the IP Address of the SLC processor. The protocol that the SLC processor supports is automatically detected. It will use either CSP protocol on port 2222 (0x8AE) or EthernetIP protocol on port 44818 (0xAF12).
Local Address	New in 8.1.8 The local address to connect from when establishing a TCP connection. If left blank, then the driver will simply pick an available address.
Communication Timeout	After sending a request to the SLC processor, the Communication Timeout setting is the amount of time in milliseconds to wait for a response before treating it as a failure.
Browse Cache Timeout	When the data table layout is read from the SLC processor, the Browse Cache Timeout value is the amount of time in milliseconds to cache the results.
Connection Path	The Connection Path value is used to define the route of the SLC processor to connect to. Currently routing through the ControlLogix Ethernet Communication Interface Module (1756-ENET) to the ControlLogix Data Highway Plus-Remote I/O Communication Interface Module (1756-DHRIO) and on to a SLC processor of the DH+ network is supported.

Advanced Settings

Name	Description
Disable Processor Browse	Disables the processor browse setting. (Default is false.)
Zero TNS Connection	Disables the Zero TNS connection setting. (Default is false.)

More Information On Connection Path

The Connection Path format contains 4 numbers separated by commas. The first number is always 1 and tells the 1756-ENET module to route through the backplane. The second number is the slot number of the 1756-DHRIO module of the DH+ network the SLC processor is connected to. The third number is the channel of the 1756-DHRIO module that the SLC processor is connected to. Use 2 for channel A and 3 for channel B. The final and fourth number is the DH+ node number. This number is in octal and is the same as configured in the SLC processor. See the ControlLogix Ethernet Communication interface Module User Manual for more information.

- Connection Path Format: 1,<1756-DHRIO slot number>,<1756-DHRIO channel>,<DH+ node number>

The valid range for the 1756-DHRIO slot number is between 0 and 16 but depends on the chassis size. The 1756-DHRIO channel is either 2 for channel A or 3 for channel B. The DH+ node number range is from 00 to 77 octal.

Supported SLC Connection Methods

- SLC505 direct
- SLC505, SLC504, SLC503 connected through 1761-NET-ENI
- SLC504 connected through 1756-DHRIO
- SLC505, SLC504, SLC503 connected through Spectrum Controls WebPort 500

Allen Bradley Connection Paths

Connections to ControlLogix, CompactLogix, PLC-5, MicroLogix and SLC Allen-Bradley processors through a ControlLogix Gateway require a connection path. The connection path is unique to your setup and is dependent on what modules the connection is being routed through. However, each connection path follows a basic set of rules outlined below.

Follow the Path

A connection path is a path that when followed leads from the ControlLogix gateway to a processor residing in a numbered slot of a chassis somewhere on site. You only have to build the connection path as you go.

Setting Up the Device Connection

The first connection point between Ignition and the device is a ControlLogix Ethernet module such as an ENET, ENBT, or EN2T module. This means that while the driver type used is the same type as the PLC you want to connect to, the hostname actually needs to point to the ControlLogix Gateway. The connection path is then followed to go out of the ControlLogix Gateway and into the PLC you are connecting to.

You need to have 6 numbers/entries to specify the connection path. The way you find each number is described in the following table:

Number	Description
1st Number	Is 1 and means move to the back plane.
2nd Number	Is the slot number of the module you want to move to.
3rd Number	Is the exit port or channel of that module that you want to exit through.
4th Number	Is the address of entry point to the next module (DH+ Station Number/ControlNet Address/IP Address of ethernet module).
5th Number	Is 1 and means move to the back plane (from this 5th Number, it starts repeating as the 1st Number).
6th Number	Is the processor slot number OR the slot number of the module you want to move to.

The process of coming up with these numbers may sound complicated at first but after some practice it gets easier.

Examples

Below there are a few examples that go over the differences with using specific modules to route the connection. Note that typically, the end result PLC does not matter. So while the example may show connecting to a PLC5 through ControlNet, the the PLC5 could easily be swapped out with another PLC, and the connection path would remain the same. The big change when using different end result PLCs is what device driver to use for the connection.

NOTE

Understanding how the Connection Paths are built listed above is important to understand the specifics of using different modules in the examples below.

Connecting to a PLC Through ControlNet



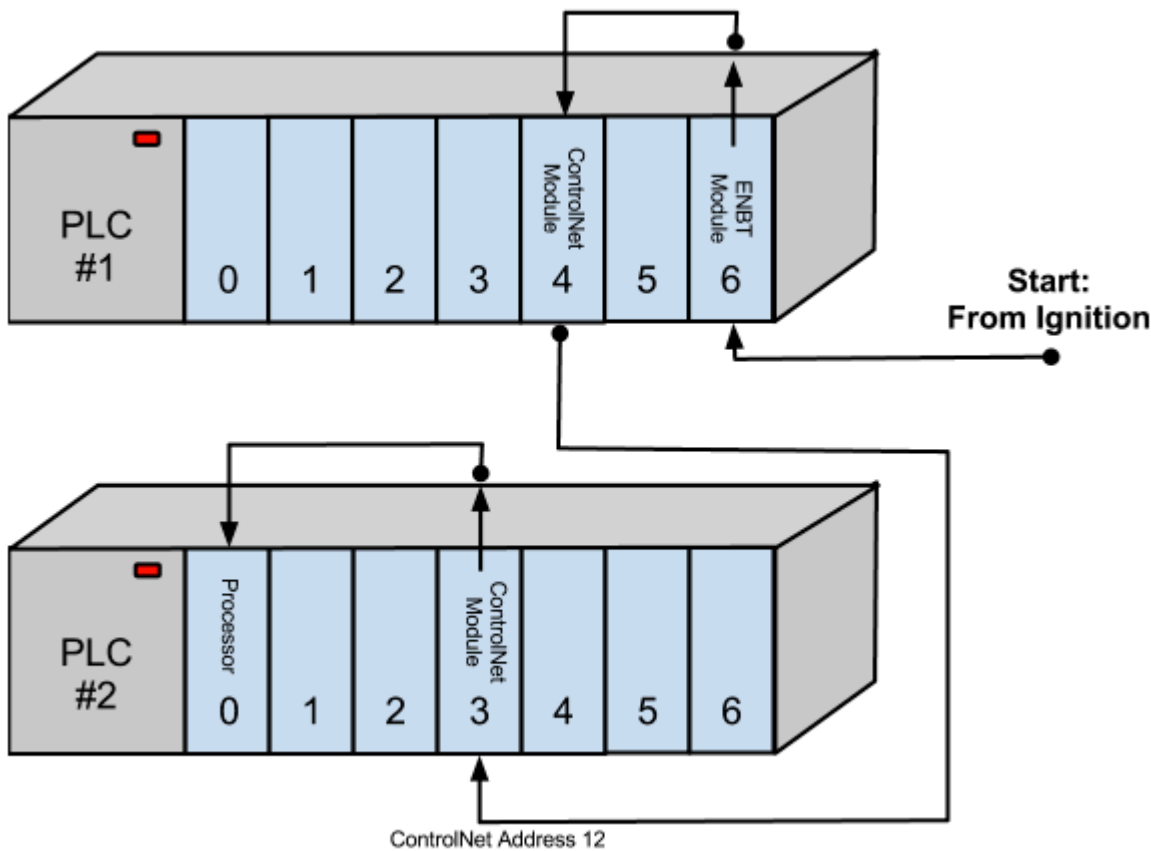
INDUCTIVE
UNIVERSITY

ControlNet Example

[Watch the video](#)

The example below connects to a PLC5 using ControlNet that has a **connection path** of **1,4,2,12,1,0** going from **PLC 1** to **PLC 2**. The device driver is the Allen-Bradley PLC5, and the hostname is the IP address of the ControlLogix Gateway (PLC #1).

Number/Entry	Description
1	Move out of ENBT Module to the backplane.
4	Move to ControlNet Module in Slot 4.
2	Move out ControlNet Port.
12	Move in ControlNet Module at ControlNet Address 12.
1	Move out of ControlNet to the backplane.
0	Move to the Processor in Slot 0.



Connecting to a PLC Using ENBT

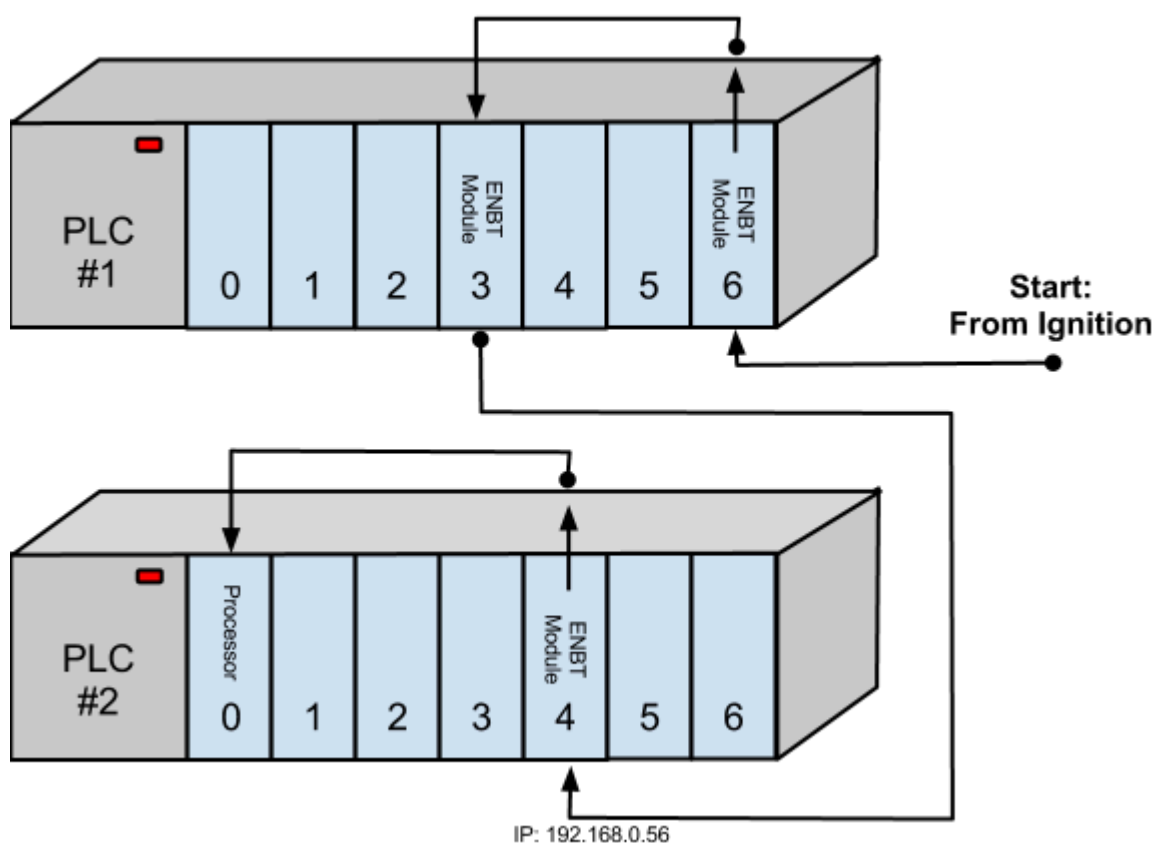


An ENBT Example

[Watch the video](#)

The example below connects to a ControlLogix using ENBT that has a **connection path** of **1,3,2,192.168.0.56,1,0** going from **PLC 1** to **PLC 2**. The device driver is the Allen-Bradley ControlLogix, and the hostname is the IP address of the ControlLogix Gateway (PLC #1).

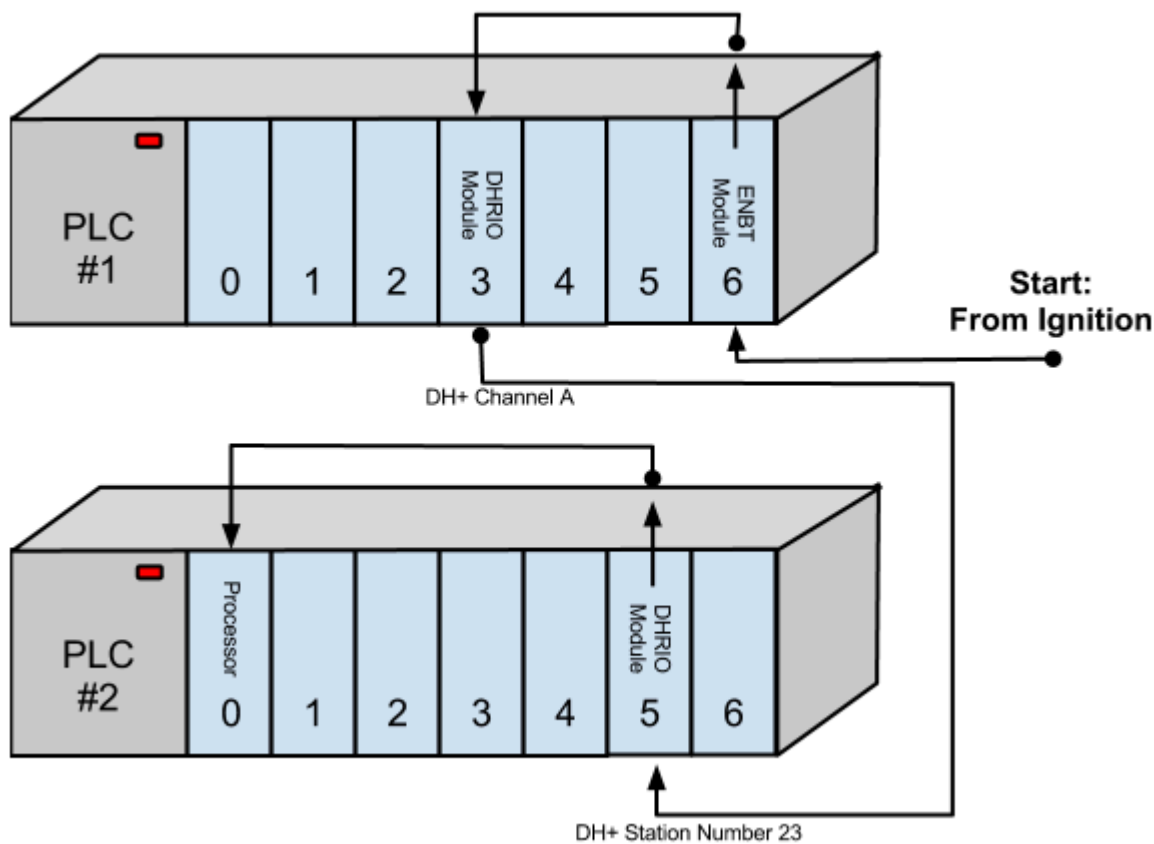
Number/Entry	Description
1	Move out of ENBT Module to the backplane.
3	Move to ENBT Module in Slot 3.
2	Move out the Ethernet Port.
192.168.0.56	Move in EBNT Module at IP Address 192.168.0.56.
1	Move out of EBNT to the backplane.
0	Move to the Processor in Slot 0.



Connecting to a ControlLogix Using Data Highway Plus (DH+)

The example below connects to a ControlLogix using Data Highway Plus that has a **connection path** of **1,3,2,23,1,0** going from **PLC 1** to **PLC 2**. The device driver is the Allen-Bradley ControlLogix, and the hostname is the IP address of the ControlLogix Gateway (PLC #1).

Number/Entry	Description
1	Move out of ENBT Module to the backplane.
3	Move to DHRIO Module in Slot 3.
2	Move out DH+ Channel A.
23	Move in DH+ Station Number 23.
1	Move out of DHRIO Module to the backplane.
0	Move to the Processor in Slot 0.



Connecting to Multiple SLCs Using Data Highway Plus (DH+)



A DH+ Example

[Watch the video](#)

The example below connects to 3 different SLC using Data Highway Plus. Each SLC connection would have their own device connection, each with a unique connection path.

The connection paths are:

- SLC X 1,3,2,21
- SLC Y 1,3,3,40
- SLC Z 1,3,3,41

The device driver is the device you want to connect to (in this case, Allen-Bradley SLC), and the hostname is the IP address of the ControlLogix Gateway (PLC #1).

SLC X

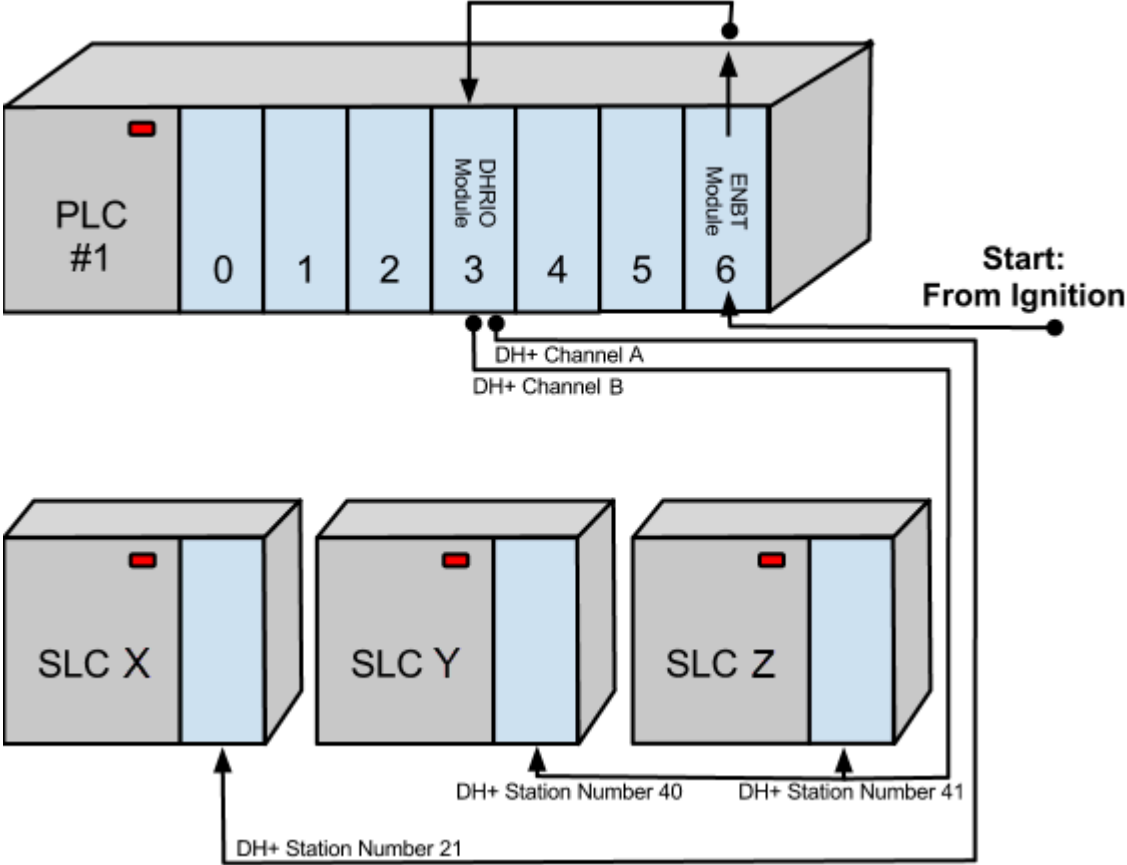
Number/Entry	Description
1	Move out of ENBT Module to the backplane.
3	Move to DHRIO Module in Slot 3.
2	Move out DH+ Channel A.
21	Move in DH+ Station Number 23.

SLC Y

Number/Entry	Description
1	Move out of ENBT Module to the backplane.
3	Move to DHRIO Module in Slot 3.
3	Move out DH+ Channel B.
40	Move in DH+ Station Number 40.

SLC Z

Number/Entry	Description
1	Move out of ENBT Module to the backplane.
3	Move to DHRIO Module in Slot 3.
3	Move out DH+ Channel B.
41	Move in DH+ Station Number 41.



Modbus

Modbus Devices

Unlike some of the other drivers (like Allen Bradley, Siemens) Modbus is a standard and not specific to a brand of device. Modbus is a protocol used for connecting to many types of devices. This is a huge benefit, but can lead to a lot of confusion because different device manufacturers may design their devices differently, and the documentation for the different manufacturers varies wildly in quality and availability. The main idea behind the Modbus standard is that you should have a regular pattern for tags inside the device, and use the same connection methods.

Our generic Modbus driver allows the Ignition OPC-UA server to communicate with any device that supports the Modbus TCP protocol or the RTU over TCP protocol. Because of this, the Modbus driver can connect directly to any devices that support Ethernet communications, even if they use the older RTU standard. If you're not sure which connection type to use, it's probably not RTU. However, it's not difficult to just try both and see which works.

Connecting to a Device

The [Connecting to Modbus Device](#) section contains step-by-step instructions on how to connect to a Modbus device. It is important to only add one Modbus device connection to Ignition for each IP address, regardless of how many devices are using that IP address. When communicating to multiple Modbus devices on the same IP address where each has a unique unit ID, either include the unit ID in the Modbus specific address or set it in the [address mapping](#) for the device.

It's easy to get connected to a Modbus device, but figuring out the addressing can be time consuming. Even with poor (or missing) device documentation, it's just a little bit of trial and error to get your addressing set.

Modbus Addresses

Getting access to your Modbus tags can be confusing because of all the different options available in the device connection. For example, you could have 0/1 based addressing or reversed word order that isn't clearly documented in the device instructions. It's helpful to manually create a few Tags in Ignition and change your connection settings until your values are correct.

One important setting to note is the **Max Holding Registers Per Request**. This setting can cause all of your test tags to go to bad quality when only one is bad. Change this setting (under the Advanced

properties) to 1 while you are testing, and set it back when you are done. If you leave it at 1, this will cause a huge strain on your system after you have added hundreds or thousands of tags from your device.

Manually Addressing

Manually creating tags in Ignition is pretty easy. If you want to look at Holding Register 1 (a common address in Modbus), the OPC Item Path is "[devicename]HR1". It's that simple! Try setting up HR0, HR1, and HR2 while you are testing to help figure out your connection settings.

Address Mapping

Creating an **Address Map** in Ignition for your device allows you to drag and drop tags just like any of the browseable device connections (like an Allen Bradley ControlLogix). You can even import and export your maps to make it quick and easy to set up many devices. Once you have your mapping set up, just drag your Tags into Ignition and start designing. Make sure to test a few tag values when you get the Address Mapping set up. Modbus devices cannot verify that your mapping is correct, it just creates a list of tags for you.

Connecting to Modbus Device

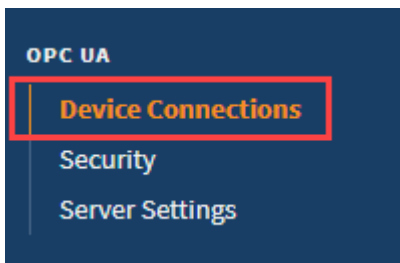


INDUCTIVE
UNIVERSITY

Connecting to Modbus Device

[Watch the video](#)

1. Go to the **Config** section of the Gateway Webpage.
2. Scroll down and select **OPC UA > Device Connections**.



3. On the **Devices** page, find the orange arrow and click on **Create new Device**. There are a few options to choose from:
 - **Modbus RTU over TCP**: Which connects to devices that implement the Modbus RTU protocol over TCP.
 - **Modbus TCP**: Which connects to devices that implement the Modbus TCP protocol.
 - **Modbus RTU** (only available if the **Serial Support Gateway** module is installed): Connects to a device via Modbus RTU over serial.

Changed in 8.1.20 Starting in version 8.1.20, the Modbus RTU driver no longer requires the Serial Support Gateway module to function.

4. In this example we'll select **Modbus TCP**, and click **Next**.

⚙️ Config > Opcua > Devices

- ☐ **Allen-Bradley SLC**
Connect to SLC 5/05s via Ethernet.
- ☐ **DNP3 Driver**
Connect to a DNP3 outstation.
- ☐ **Modbus RTU over TCP**
Connect to devices that implement the Modbus RTU protocol over TCP.
- ☐ **Modbus TCP**
Connect to devices that implement the Modbus TCP protocol.
- ☐ **Omron FINS/TCP**
Connect to devices that implement FINS over TCP.

5. On the **New Device** page, leave all the default values and type in the following fields:

- **Name:** Modbus
- **Hostname:** Type the **IP address**. For example, 10.20.8.117.

Config > Opcua > Devices

General

Name	Modbus 
Description	
Enabled	☒ (default: true)

Connectivity

Hostname	10.20.8.117
Timeout	2000 (default: 2,000)
Connection Path	 (default:)
Concurrent Requests	2 (default: 2)

☐ Show advanced properties

Create New Device

- You can check the box for **Show advanced properties** to see the additional settings, or you can keep all the defaults.
- Click **Create New Device**. The **Devices** page is displayed showing the **Modbus** device is successfully created and added to Ignition. The **Status** will show as Disconnected and then Connected.

Config > Opcua > Devices

 Successfully created new Device "Modbus"

Name	Type	Description	Enabled	Status	
Sim	Simulator: Generic		true	Running	delete edit
Modbus	Allen-Bradley CompactLogix (Legacy)		true	Connecting	More ▼ edit

Unlike other PLCs, Modbus devices do not support browsing, therefore you can not browse the Tags by going to the **OPC Connections > Quick Client** in the **Configure** section of the Gateway. To see and browse the Tags, you need to create the Tags as described in the [Modbus Addressing](#) section.

Modbus Drivers using Ethernet

The generic Modbus driver allows the Ignition OPC-UA server to communicate with any device that supports Modbus TCP protocol. The Modbus driver can connect directly to devices that support Ethernet communications. It can also connect to Modbus devices through a Gateway.

It is important to only add one Modbus device in the Ignition Device List per IP address. When communicating to multiple Modbus devices through a Gateway each with a unique unit ID, either include the unit ID in the [Modbus specific address](#) or set it in the [address mapping](#) for the device.

Supported Functions Codes

The Modbus driver supports the functions codes listed below. In some cases, a device may not allow certain function codes. To remedy this, advanced properties on the device connection can restrict or force specific functions codes. See the [Device Connection Settings](#) section on this page for more details.

Function Name	Code	Hex
Read Discrete Inputs	02	0x02
Read Coils	01	0x01
Write Single Coil	05	0x05
Write Multiple Coils	15	0x0F
Read Input Register	04	0x04
Read Holding Register	03	0x03
Write Single Register	06	0x06
Write Multiple registers	16	0x10
Mask Write	22	0x16

Device Connection Settings

General

Name	Description
Name	The user-defined name for this Device. The name chosen will show up in OPC Item Paths and under OPC-UA Server > Devices of the Configure page of the Gateway. The Device Name must be alphanumeric.
Description	The user-defined description for the device. This is only used as a note to differentiate between devices.
Enable Device	Only devices that are enabled appear in Connections > Devices of the Status page of the Gateway, and thus have their tags available for use.

Connectivity (Modbus RTU over TCP and Modbus TCP connections)

Name	Description
Hostname	Is the IP Address of the Modbus device.
Port	Is the port to use when connecting to a Modbus device. The Modbus TCP port specified in the Modbus specification is 502, but it can be changed to a different port.
Local Address	New in 8.1.8 The local address to connect from when establishing a TCP connection. If left blank, then the driver will simply pick an available address.
Communication Timeout	Is the amount of time in milliseconds to wait for a response before treating it as a failure, after sending a request to the Modbus device. Note: When working with a Modbus RTU over TCP connection, each "device" would be an individual Modbus device on the Modbus network.

Connectivity (Modbus RTU connections)

Name	Description
Serial Port	The name of the Serial Port (i.e COM1).
Bit Rate	The bit rate for the connection.
Data Bits	Data bits for the connection.
Parity	Parity configuration for the connection.
Stop Bits	Determines the stop bits for the connection.
Handshake	Determines the handshake for the connection.
Communication Timeout	Determines the maximum amount of time the connection will wait for a response.
RS-485 Mode	New in 8.1.25 Enables RS-485 mode when checked for a device using RS-485 instead of RS-232. RS-232 mode is the default mode.

Advanced

Name	Description
Concurrent Requests	New in 8.1.0 The number of requests that Ignition will try to send to the device at the same time. Increasing this number can sometimes help with your request throughput, however increasing this too much can overwhelm the device and negatively impact communications with the device.
Max Holding Registers per Request	The maximum number of Holding Registers the device can handle. Because some Modbus devices cannot handle the default of requesting 125 Holding Registers in one request, to accommodate this limitation you can change this setting.
Max Input Registers per Request	The maximum number of Input Registers the device can handle. Because some Modbus devices cannot handle the default of requesting 125 Input Registers in one request, to accommodate this limitation you can change this setting.
Max Coils per Request	The maximum number of Coils the device can handle. Because some Modbus devices cannot handle the default of requesting 2000 Coils in one request, to accommodate this limitation you can change this setting.
Max Discrete Inputs per Request	The maximum number of Discrete Inputs the device can handle. Because some Modbus devices cannot handle the default of requesting 2000 Discrete Inputs in one request, to accommodate this limitation you can change this setting.
Reverse Word Order	When reading and writing 32bit values from/to a Modbus device, the high word comes before the low word. By checking this option, the low word comes before the high word. The Modbus specification does not include a section for reading and writing 32bit values and as a result device manufacturers have implemented both methods.
Zero-based Addressing	When this option is checked, the address range for each area starts at 0. If unchecked, the range starts at 1. The Modbus specification states that Modbus addresses are to be zero based. Meaning Modbus addresses start at 0 instead of 1. To read a value from Modbus address 1024, 1023 is sent to the device. When

Name	Description
	connecting to devices that do not adhere to zero based addressing, make sure this option is not selected.
Span Gaps	When this option is checked, it spans address gaps when optimizing requests, reducing the number of requests but increasing the amount of data requested at once. If unchecked, it does not span the address gaps.
Allow Write Multiple Registers Request	Enable or disable Modbus function code 0x10, Write Multiple Registers. Some devices may not support this function code. Caution: Disabling this option will break the ability to write 32-bit and String values correctly to registers.
Force Multiple Register Writes	Force the use of Modbus function code 0x10, Write Multiple Registers, on write requests.
Allow Write Multiple Coils Request	Enable or disable Modbus function code 0x0F, Write Multiple Coils. Some devices may not support this function code.
Allow Read Multiple Registers Request	If disabled all registers will be read in individual read requests. Disable with caution.
Allow Read Multiple Coils	If disabled all coils will be read in individual read requests. Disable with caution.
Allow Read Multiple Discrete Inputs	If disabled all discrete inputs will be read in individual read requests. Disable with caution. Note: Function code 0x02 is always used to read Discrete Inputs, regardless what this property is set to.
Reconnect After Consecutive Tries	Force a reconnect after 3 consecutive timeouts. Default is true.
Max Retry Count	Maximum number of times to retry a read request after receiving a response containing an allowable Exception Code (GatewayPathUnavailable, GatewayTargetDeviceFailToRespond, SlaveDeviceBusy, or SlaveDeviceFailure). Default is 1.

String Handling

Name	Description
Reverse String Byte Order	When reading and writing string values from/to a Modbus device, the low byte comes before the high byte. By checking this option the high byte comes before the low byte. If reading a string value from a device should read ABCD but BADC appears in Ignition, then check this option.
Right Justify Strings	Strings stored in a Modbus device may contain leading spaces or trailing spaces. This can produce unwanted results so that Modbus driver removes spaces or zeros when reading string values. By default, left justify string handling is used when reading and writing strings. When you check this option, right justify string handling is used.
Read Raw Strings	Whether or not to read the entire length of a string, ignoring any null bytes encountered.

Modbus Addressing



INDUCTIVE
UNIVERSITY

About Modbus Addressing

[Watch the video](#)

Manually Addressing a Modbus Device

Modbus doesn't support tag browsing, this means you cannot view the Tags in the **Connected Devices window** in the Designer or from the **OPC Connections > Quick Client** in the Config section of the Gateway.

There are two ways you can create tags so that you can browse them:

1. By manually specifying each address

- This is done from the Designer by entering Modbus Specific Addresses into the **OPC Item Path** of an OPC Tag. See below for detailed information.

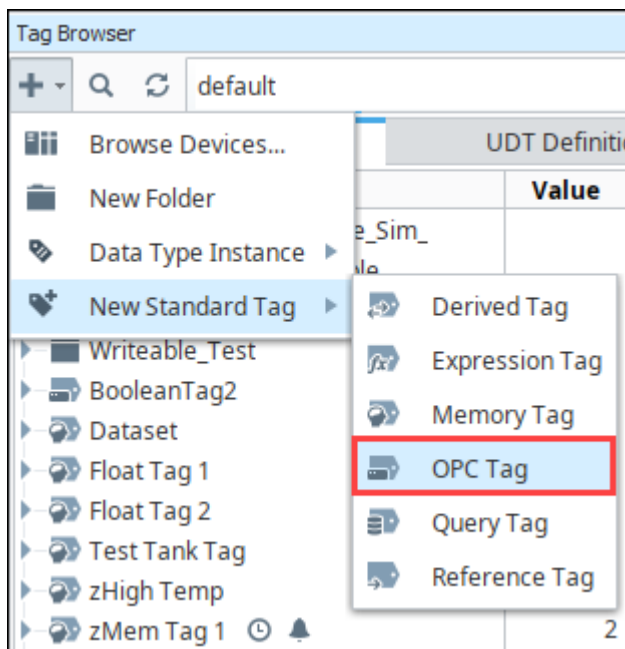
2. By specifying the address mapping

- This is done from the Gateway, see the **Modbus Address Mapping** section.

To manually specify each address

You can enter Modbus specific addresses into the OPC Item Path of an OPC Tag by using the following designators along with the Modbus address:

1. In the **Tag Browser**, click the Add (+) icon.
2. Select **New Standard Tag** then **OPC Tag**.



3. In the **Tag Editor** window, as an example, you can set the following values:

- Name: **Temp**
- Data Type: **Integer**
- OPC Server: Choose **Ignition OPC UA Server** from the dropdown.
- OPC Item Path: **[Modbus]HR1**. The **Modbus** device name goes in the square brackets. Next, you give the address to the PLC which in this case is the **HR** designator plus **1** as the Modbus address. The [Modbus Specific Addressing](#) section below, explains how you can construct these addresses.

4. Click **OK**. Now you can see the Temp tag in the Tag Browser.

Modbus Specific Addressing

Per the Modbus protocol specification, the following four basic types of addresses can be read from a device:

- Holding Registers (read/write 16 bit words)
- Input Registers (read only 16 bit words)
- Coils (read/write bits)
- Discrete Inputs (read only bits associated with device input points)

Manually Create an Address for a Single Tag

To manually enter Modbus Specific Addresses into the **OPC Item Path** of the **Tag Editor** window, use one of the following designators plus the Modbus address:

i NOTE

Other OPC servers represent each type by starting the OPC address with a number, for example, 4 for holding registers.

Designator	Description
HR	for 16 bit signed Holding Register (HR1, equivalent to 40001 in other applications)
IR	for 16 bit signed Input Register (IR1, equivalent to 30001)
C	for Coil (C1, equivalent to 00001)
DI	for Discrete Input (DI1, equivalent to 10001)

An example of using these designators with the Modbus address is to enter **HR1** in the OPC Item Path of an OPC Tag in the Tag Editor window, which is the **HR** designator plus the Modbus address **1**.

Because some devices that support Modbus protocol store data in **BCD format**, there are two additional designators. These designators convert the data from BCD format to decimal when reading data from the device and convert data from decimal to BCD when writing to the device.

Designator	Description
HRBCD	for Holding Register with BCD conversion.
HRBCD_32	for 2 consecutive Holding Registers with BCD conversion.
IRBCD	for Input Register with BCD conversion.
IRBCD_32	for 2 consecutive Input Registers with BCD conversion.

To accommodate other data encoding commonly used by Modbus supported devices, the following designators are available for Modbus specific addressing:

Float/Double

Description	Holding Register Designator	Input Register Designator
2 consecutive Registers with Float conversion.	HRF	IRF
4 consecutive Registers with Double conversion.	HRD	IRD

Integer

Description	Holding Register Designator	Input Register Designator
Holding Registers with 16 bit unsigned integer conversion.	HRUS	IRUS
2 consecutive Registers with 32 bit integer conversion.	HRI	IRI
2 consecutive Registers with 32 bit unsigned integer conversion.	HRUI	IRUI
4 consecutive Registers with 64 bit integer conversion.	HRI_64	IRI_64
4 consecutive Registers with 64 bit unsigned integer conversion.	HRUI_64	IRUI_64

To read or write string values from/to a Modbus device, the following designation is available for Modbus specific addressing:

Designator	Description
HRS	read or write consecutive Holding Registers as a string value.

There are two characters for each word and the order of which character comes first is controlled by the Reverse String Byte Order device setting as described in the [Connecting to Modbus Device](#) section. Because two characters are stored in a word, the string length must be an even number of characters.

HRS FORMAT: HRS<Modbus address>:<length>

Examples	Description
[DL240]HR1024	Read 16bit integer value from Holding Register 1024.
[DL240]HRBCD1024	Read BCD value from Holding Register 1024.
[DL240]IR512	Read 16bit integer value from Input Register 512.
[DL240]C3072	Read bit value from Coil 3072.
[DL240]IR0	Read 16bit integer value from Input Register 0.
[DL240]HRS1024:20	Read 20 character string value starting at Holding Register 1024.

! UNIT ID

You can also specify the Modbus unit ID by pre-pending it to the Modbus address. For example, to access Modbus unit ID 3 and read HR1024, the full OPC path is as follows:

```
[DL240]3.HR1024
```

Bit-level addressing

For bit-level addressing, you just append a period and the bit number you want to read and write to a bit. Your Modbus device must support the Mask Write command (your device documentation should specify if it does).

To read or write to a specific bit within a holding register, simply append the location of the bit as demonstrated in these examples:

- [DL240]HR1024.0 will read and write to the first bit of the holding register.
- [DL240]HR1024.10 will read and write to the 11th bit of the holding register.

Modbus Address Mapping



INDUCTIVE
UNIVERSITY

**About Modbus Address
Mapping**

[Watch the video](#)

Because it can be tedious to manually enter OPC Tag information one-by-one, the driver offers an address mapping feature. This feature allows entering blocks of common addresses and the driver will create the individual addresses and display them in the [Connected Devices window](#).

Another benefit of address mapping is that the addresses inside a device can have a different numbering scheme than the Modbus address. The Direct Automation DL240 is a perfect example of this. Address V2000, capable of holding a 16 bit integer, is Modbus Holding Register 1024. In addition, the DL240 addressing is in octal meaning there are no 8 or 9s. The sequence of addresses are: V2000, V2001, V2002, V2003, V2004, V2005, V2006, V2007, V2010, V2011.... V3777.

Address Mapping Properties

Name	Description
Prefix	A prefix applied to each mapped address as they appear in the Connected Devices window. Must compose of letters, numbers, and underscore characters. The following values are reserved, and may not be used: • HR • IR • C • DI
Start and End	Numerical values will be assigned to each mapped addresses. These properties determine the range of the numerical assignments. Follows the Prefix. The difference between these two values determines how many mapped addresses will be created. New in 8.1.42 Range must be set between 0 and 65,535. The Gateway will throw an error if start and end values are out of range.
Step	When enabled, adjacent addresses will be combined. This is commonly used to combine two words (16-bit addresses) into a double word (32-bit addresses). Please see the Floating Point or 32-bit Address Mapping for more details.
Radix	The base number of unique digits for modbus addresses. Determines what format addresses in the device are incremented and labeled (HR0, HR1,...HR9, HR10, HR11). Common values are 10 (decimal system) or 16 (hexadecimal).
Unit ID	The Unit Id for the device to use. When several Modbus devices are connected to a single IP address, the step determines which device the mapping should be applied against. A value of 0 means the first device, and should be used when only a single Modbus device is connected. See Address Mapping Multiple Devices for more details.
Modbus Type	The table each mapped address should run against, as well as the type and size of each address.

Name	Description
Modbus Address	The address in the device that mapping will begin at. Since the Modbus Type property denotes which table the address will run against, the value here does not need to start with the entity number. Thus, when attempting to start a mapping at the 100th Holding Register, the value on this property would be 100 (assuming one-based addressing), not 40100; the Modbus Type property determines what the leading number is.

Simple Mapping Demonstration

Temperature readings are being stored in 10 16-bit addresses: 40,010 - 40,019. There is a single Modbus device at the IP address (unit ID 0), and the addresses are decimal. The mapped addresses should appear in the **Connected Devices window** as "Temp1", "Temp2", and so-on. The following configuration would be used:

- **Prefix:** Temp
- **Start:** 1
- **End:** 10
- **Step:** False
- **Unit ID:** 0
- **Modbus Type:** Holding Register (Int16)
- **Modbus Address:** 10

NOTE

Address 40,010 (HR10) uses Modbus Address 10 as a starting point and not 40,010. The leading 4 is automatically entered for you when you select Holding Register from the dropdown. The same is true for all other types of Tags: Inputs are 30,000, Discrete are 10,000, etc.

Prefix	Start	End	Step	Unit ID	Modbus Type	Modbus Address	
Temp	1	10	☐	0	Holding Register (Int16)	10	[delete]

Radix

10

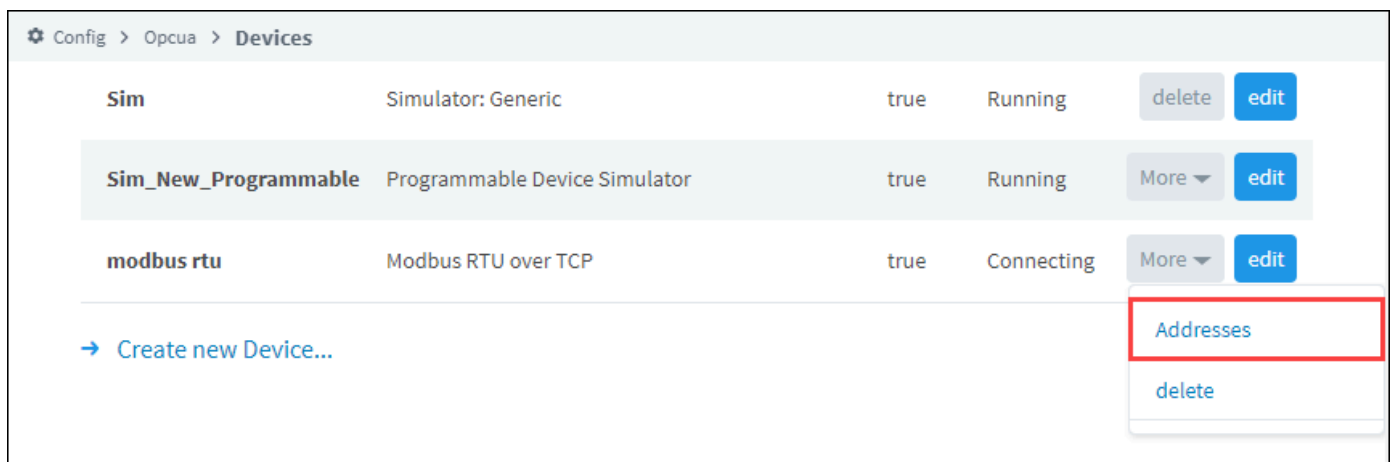
The above configuration results in the items Temp1 through Temp10 appearing in the **Connected Devices window**.

NOTE

We are using Unit ID 0 in the demonstration above, but your Unit ID can vary depending on the Modbus device you are using.

Specify the Address Mapping

1. Go to the **Config** section of the Gateway Webpage.
2. Scroll down to **OPC UA > Device Connection**.
3. Click on the **More** button and select **Addresses** to the right of your Modbus device.



The screenshot shows the 'Config > Opcua > Devices' section of the Gateway Webpage. It displays a table of device connections. The 'modbus rtu' device is highlighted, and the 'More' button is clicked, showing a dropdown menu with 'Addresses' selected.

Device Name	Device Type	Enabled	Status	Actions
Sim	Simulator: Generic	true	Running	delete edit
Sim_New_Programmable	Programmable Device Simulator	true	Running	More edit
modbus rtu	Modbus RTU over TCP	true	Connecting	More edit

[→ Create new Device...](#)

Dropdown menu for 'modbus rtu':

- [Addresses](#)
- [delete](#)

4. Click on **Add Row**.

⚙️ Config > Opcua > Devices

Address Configuration

No file chosen

Prefix	Start	End	Step	Unit ID	Modbus Type	Modbus Address
Radix	10					

There are currently no address mappings configured...

5. Enter the mapping as follows:

- Prefix: **V**
- Start: **2000**
- End: **3777**
- Modbus Type: **Holding Register (Int16)**
- Modbus Address: **1024**
- Radix: **8** (8 causes the address to be in octal, also known as base 8)

Config > Opcua > Devices

Address Configuration

Choose File No file chosen

Import Configuration

Export Configuration

Prefix	Start	End	Step	Unit ID	Modbus Type	Modbus Address
V	2000	3777	☐	0	Holding Register (Int16)	1024 [delete]

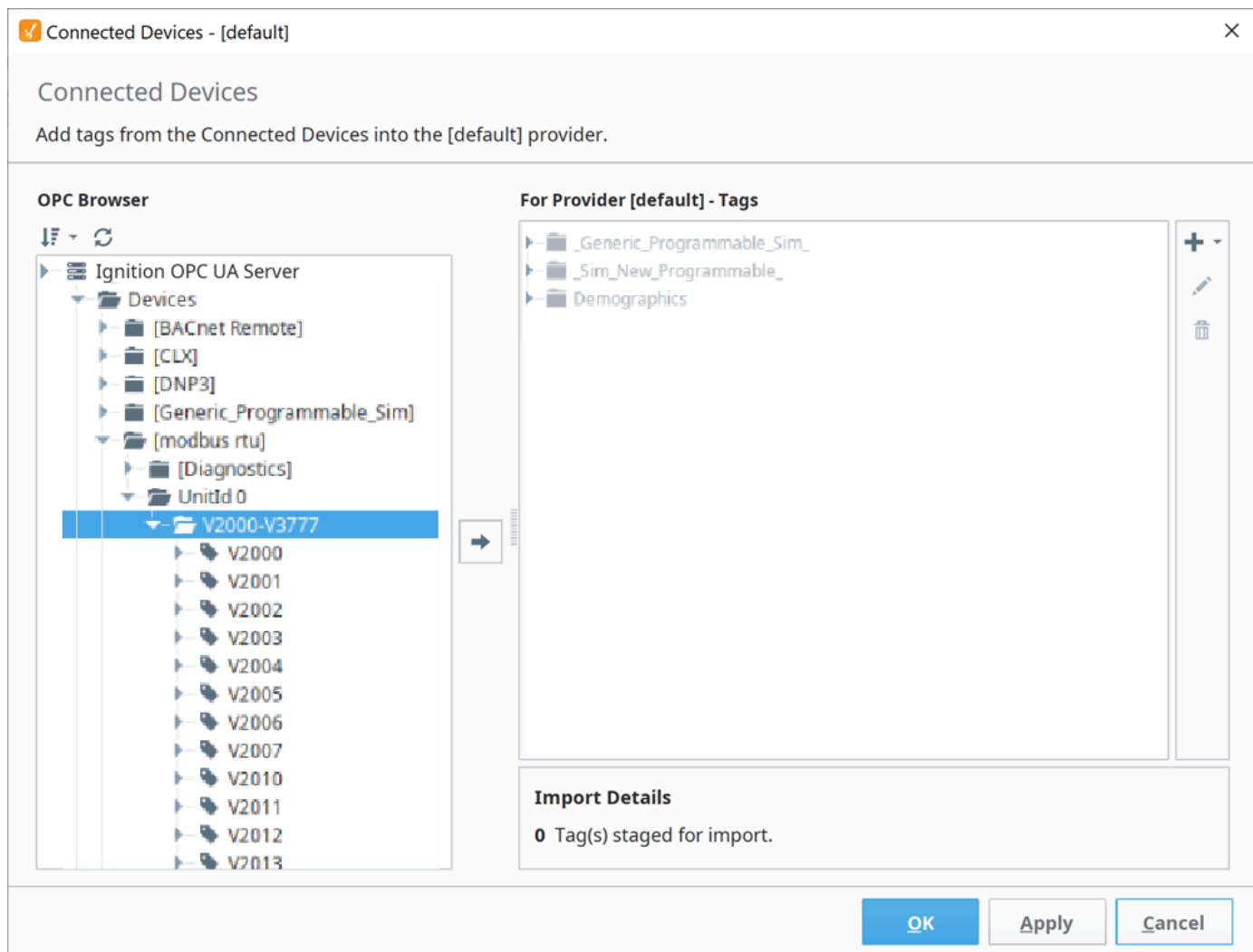
Radix 8 

These settings map the **Modbus address range V2000 to V3777** in octal to **Modbus Holding Register addresses 1024 to 2048**.

NOTE

The mappings for string data types cannot be entered. Strings can only be read or written using Modbus Specific Addressing.

- Click **Save**.
- Go to the Connected Devices window in Designer or the OPC Connections > Quick Client (on Gateway), and open the **Modbus** folder. You can now see all the Modbus addresses from V2000 to V3777 in octal.



Here is an example of mapping for all of the Modbus DL240 addressing.

Prefix	Start	End	Unit ID	Modbus Type	Modbus Address
V	2000	3777	0	Holding Register (Int16)	1024
X	2000	477	0	Discrete Input	2028
Y	0	477	0	Coil	2048
TAO	0	127	0	Input Register (BCD16)	0

Prefix	Start	End	Step	Unit ID	Modbus Type	Modbus Address
V	2000	3777	☐	0	Holding Register (Int16)	1024 [delete]
X	0	477	☐	0	Discrete Input	2048 [delete]
Y	0	477	☐	0	Coil	2048 [delete]
TAO	0	127	☐	0	Input Register (BCD16)	0 [delete]
Radix 8						

Address Mapping Multiple Devices

It is not recommended to communicate to multiple Modbus devices through a Modbus Gateway where Gateway has the same address. Therefore, do not add multiple Modbus devices with the same IP address.

Only add one Modbus device to the Ignition OPC UA Server device list for Gateway and specify the different unit IDs in the address mapping. The unit ID is specified for each entry in the address mapping for the Modbus device. Notice in the example below, the Prefix, Start, End, Modbus Type and Modbus Address can be the same for two entries provided that the Unit IDs are different.

Prefix	Start	End	Unit ID	Modbus Type	Modbus Address
V	2000	3777	0	Holding Register (Int16)	1024
V	2000	3777	1	Holding Register (Int16)	2024

Prefix	Start	End	Step	Unit ID	Modbus Type	Modbus Address
V	2000	3777	☐	0	Holding Register (Int16)	1024 [delete]
V	2000	3777	☐	1	Holding Register (Int16)	2024 [delete]
Radix 8						

Now when browsing the Modbus device, the unit ID will show as a folder and the OPC Tag path includes the unit ID. This only happens when more than one unit ID is specified in the address mapping otherwise the unit ID is eliminated.

Floating Point or 32-bit Address Mapping

Modbus only supports reading and writing to memory types of bits and 16-bit words. This is not very useful when reading from or writing to float point or 32-bit integers.

To work around this problem, the Modbus driver is designed to read two consecutive 16-bit words and encode it into the desired data type.

Mapping Float Point Addresses

The Modbus address mapping below shows how to map float point addresses starting at 1024 and ending at 1030. With the box in the **Step** column checked, the addresses on the Ignition side will index by 2. In this case, R1024, R1026, R1028 and R1030 will be created.

Because **Modbus Type of Holding Register (Float)** is selected, the driver will read two consecutive 16-bit words and convert it to a floating point value. It also indexes the Modbus Address by 2 for each entry. In this case, R1024 reads from Modbus addresses 1024 and 1025 and converts them into a floating point value. When writing, the reverse of converting a floating point value into two 16-bits words is done before sending them to the device.

Prefix	Start	End	Step	Unit ID	Modbus Type	Modbus Address
R	1024	1030	☒	0	Input Register (Float)	1024
Radix 10						

The following shows what is displayed in the Connected Devices window. Notice that the numbering is indexed by two and that it matches the Modbus address. With some devices, this allows the addresses displaying in the Connected Devices window to match the addresses in the device.

Example

OPC Browser

- Ignition OPC UA Server
 - Devices
 - [modbus rtu]
 - R1024-R1030
 - R1024
 - R1026
 - R1028
 - R1030

Connected Devices - [default]

Connected Devices

Add tags from the Connected Devices into the [default] provider.

OPC Browser

Ignition OPC UA Server

- Devices
 - [BACnet Remote]
 - [CLX]
 - [DNP3]
 - [Generic_Programmable_Sim]
 - [modbus rtu]
 - [Diagnostics]
 - UnitId 0
 - R1024-R1030
 - R1024
 - R1026
 - R1028
 - R1030
 - [Modbus]

For Provider [default] - Tags

Import Details

0 Tag(s) staged for import.

OK

Apply

Cancel

Import/Export Address Mapping

The mapping configuration can be exported to a comma separated values (CSV) file. The CSV file can later be imported in other Ignition installations or similar devices. You can find a few examples of CSV files on our website. To see the examples, go to:

<https://inductiveautomation.com/downloads/extra-material>

Scroll down to **Modbus Templates** and double-click on the template files to see an example of the CSV file.

Siemens

The Siemens drivers in Ignition support basic connections to S7 devices. Ignition connects to these PLCs via TCP/IP using the S7 protocol. Similar to Modbus and some Allen Bradley connections, the Siemens S7 devices do not support tag browsing. You can create S7 tags manually in Ignition, or use Ignition's tag import/export to create all of your tags quickly in Excel or another spreadsheet program. Currently, Ignition has drivers for the following Siemens PLCs:

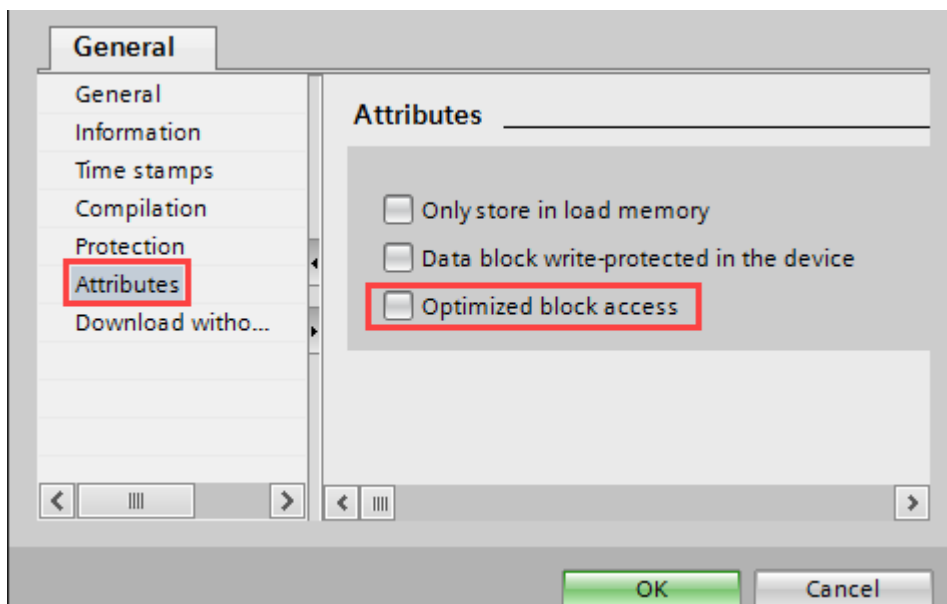
- S7-300
- S7-400
- S7-1200
- S7-1500

Some S7-1200 and S7-1500 devices have an onboard OPC UA server that can be enabled to connect over the [OPC UA server](#) instead of using the Siemens driver. OPC UA allows you to browse tags, while the driver requires you to address each tag manually.

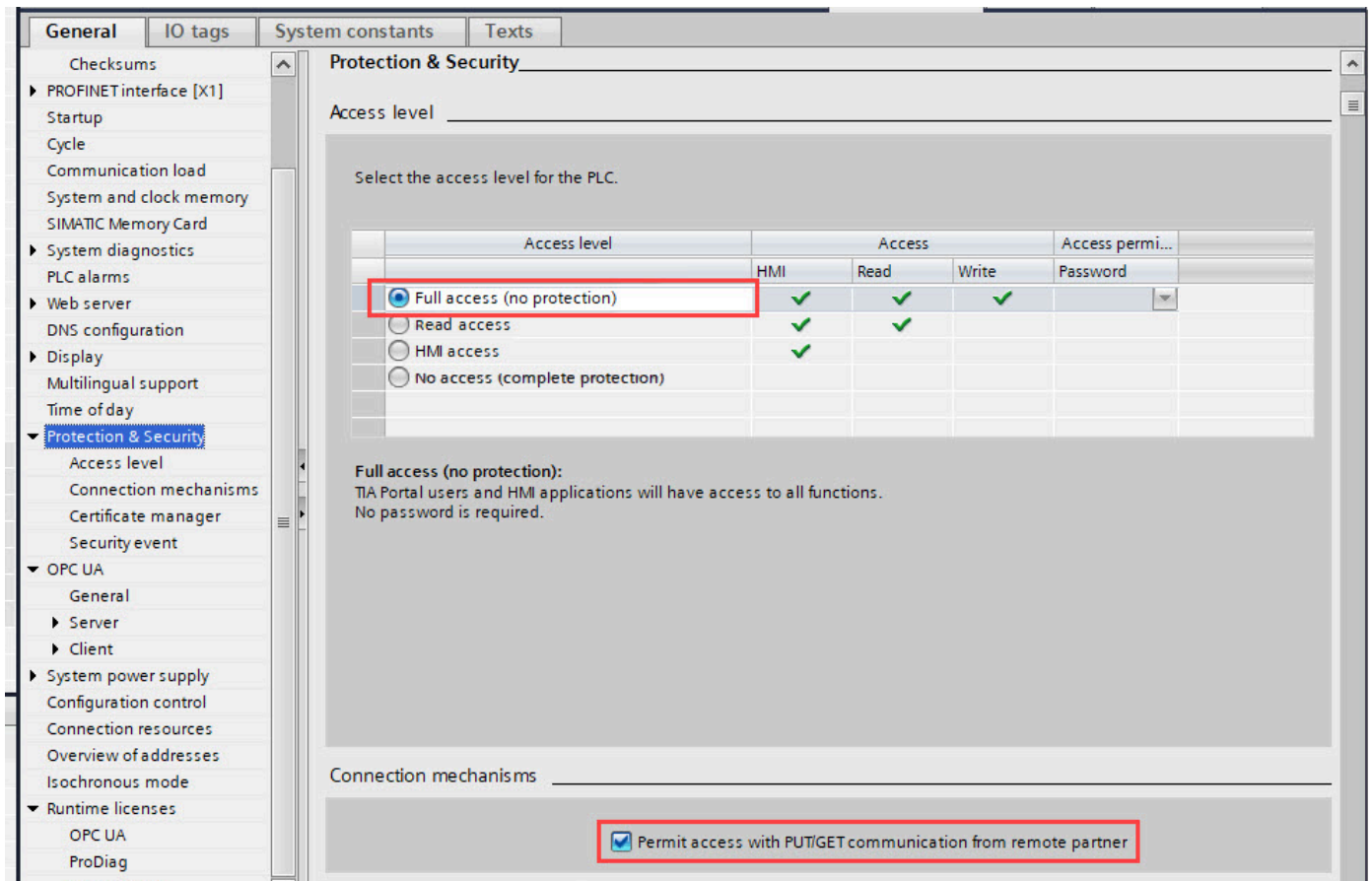
Considerations for 1200 and 1500 Devices

The requirements listed below apply when connecting to either the S7-1200 and S7-1500 drivers:

1. Only global DBs can be accessed.
2. Reading and writing is not supported for timer (TM) and counter (CT) areas.
3. The **Optimized block access** attribute option must be turned off.



4. When configuring the Protection & Security settings, make sure the **Full access (no protection)** and **Permit access with PUT/GET communication options** are selected.



NOTE

This change may require the hardware configuration to be recompiled and downloaded. Additionally, disabling "optimized access" for the data block(s) you are attempting to read may also improve the reliability of tag reads.

Connecting to a Siemens Device



INDUCTIVE
UNIVERSITY

Connecting to S7 Devices

[Watch the video](#)

1. Go to the **Config** section of the Gateway Webpage.
2. Scroll down and select **OPC UA > Device Connections**.
3. On the Devices page, click on **Create new Device**.
4. On the Add Device Step 1: Choose Type page, select **Siemens S7-1200**, and click **Next**.

5. There are four modules to choose from:
 - **Siemens S7-1500**: Connects to Siemens S7-1500 PLCs over Ethernet.
 - **Siemens S7-1200**: Connects to Siemens S7-1200 PLCs over Ethernet.
 - **Siemens S7-300**: Connects to Siemens S7-300 PLCs over Ethernet.
 - **Siemens S7-400**: Connects to Siemens S7-400 PLCs over Ethernet.
6. On the New Device page, leave all the default values and type in the following fields:
 - Name: **S71200**
 - Hostname: type the **IP address**, for example 10.20.4.71.
7. You can check the box for **Show advanced properties?** to see the additional settings, but you can keep all the defaults.
8. Click **Create New Device**. The Devices page is displayed showing the Siemens device is successfully created and added to Ignition. The Status will show as Disconnected and then Connected.

Siemens devices do not support browsing, therefore you can not browse the tags by going to the **OPC Connections > Quick Client** in the **Configure** section of the Gateway. To see and browse the tags, you need to create the tags manually as described in the **Configuring Siemens Addressing** section.

Configuring Siemens Addressing



INDUCTIVE
UNIVERSITY

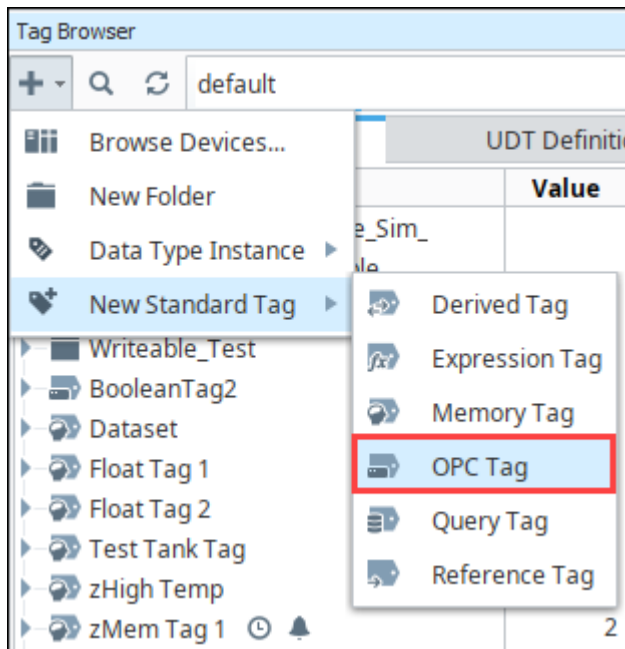
About Siemens Addressing

[Watch the video](#)

The S7 protocol does not support tag browsing. Therefore, you must configure all tags in the Designer. This can be done either manually, as needed, or by importing bulk using the tags CSV import functionality.

Manually Specify Each Address

1. From the Designer, in the Tag Browser, click the **Add (+)** icon.
2. Select **New Standard Tag**, then **OPC Tag**.



3. In the Tag Editor window, as an example, you can set the following values:

- Name: **Tag**
- Data Type: **Int4**
- OPC Server: choose **Ignition OPC-UA Server** with the edit button on the right.
- OPC Item Path: **[S71200]IW0**, the S71200 device name goes in the square brackets then you give the address to PLC which in this example is **IW0** (Word at Offset 0 in the Inputs area). The [Address Syntax](#) section explains how you can construct these addresses.

4. Click **OK**. Now you can see the Temp tag in the Tag Browser.

Address Syntax

You need a device name plus a tag address to create a tag. The device name is a known, but the tag address needs to be configured. Once you have both the device name and tag address you enter them in the in the **OPC Item Path** field of the **Tag Editor** window using the **[device_name]address format**, where **device_name** is the name of the device and **address** is the configured tag address, which is described here.

Tag addresses are made up of three different components: Area, Data Type, and Offset.

Area Syntax

Area	Syntax
DataBlocks	DBn
Inputs	I
Outputs	Q
Flags	M
Timers	T
Counters	C

Data Type Syntax

Data Type	Syntax	Signedness
Bit	X	N/A
Byte	B	Unsigned
Char	C	Signed
Word	W	Unsigned
Int	I	Signed
DWord	D	Unsigned
DInt	DI	Signed
Real	REAL	Signed
String	STRING or STRING.LEN	N/A
Date_And_Time	New in 8.1.29 DT Note: Due to limitations on Siemens devices, this data type is only supported on the following devices: • S7-300 • S7-400 • S7-1500	N/A
LInt	New in 8.1.29 LI Note: Due to limitations on Siemens devices, this data type is only supported on the following device: • S7-1500	Signed
LReal	New in 8.1.29 LREAL	Signed

Data Type	Syntax	Signedness
	Note: Due to limitations on Siemens devices, this data type is only supported on the following devices: • S7-1200 • S7-1500	

EXTRACTING BYTES IN DATE_AND_TIME ITEMS

The Siemens driver does not support Ignition tag's DateTime type. In these cases it is recommended to extract each byte in the Siemens DATE_AND_TIME value item, storing each into a separate Ignition tag. Then use an expression tag to combine each byte into a human readable datetime.

Example

```
// The following assumes each byte of the DATE_AND_TIME value is under a tag
// named like "BASIC_DATA_AND_TIME", and that each tag is a String type.

"20" + //First part of the year, assuming in 21st century
toHex({[.]BASIC_DATA_AND_TIME[0]}) + "-" + //Second part of
the year, reading from an Ignition tag.
numberFormat(toHex({[.]BASIC_DATA_AND_TIME[1]}), "00") + "-" + //Month
numberFormat(toHex({[.]BASIC_DATA_AND_TIME[2]}), "00") + "-" + //Day
numberFormat(toHex({[.]BASIC_DATA_AND_TIME[3]}), "00") + ":" + //Hours
numberFormat(toHex({[.]BASIC_DATA_AND_TIME[4]}), "00") + ":" + //Minutes
numberFormat(toHex({[.]BASIC_DATA_AND_TIME[5]}), "00") + ":" + //Seconds
numberFormat(fromBinary(left(numberFormat(toBinary({[.]BASIC_DATA_AND_TIME[6]}
+ {[.]BASIC_DATA_AND_TIME[7]}), "0000000000000000"), 12)), "000") + " " +
//Milliseconds, it includes the first 4 bits of byte 7
case( //Day of the week, last 4 bits of byte 7
right(numberFormat(toBinary({[.]BASIC_DATA_AND_TIME[7]}), "00000000"), 4),
"0000", "Sunday",
"0001", "Monday",
"0010", "Tuesday",
"0011", "Wednesday",
"0100", "Thursday",
"0101", "Friday",
"0111", "Saturday",
"Error - No Day Found")
```

Area + Data Type + Offset Examples

To form an address, you combine syntax for the desired Area and Data Type with an Offset into that area.

Address	Description
IB0	Byte at Offset 0 in the Inputs area.
IW0	Word at Offset 0 in the Inputs area.
DB500,DI8	DInt at Offset 8 in DataBlock 500.
ISTRING24.50	A String of length 50 starting at offset 24 in the Inputs area.
IX20.3	Bit 3 of the Byte at Offset 20 in the Inputs area.
T0	Timer at offset 0 (No DataType is specified for Timers).
C0	Counter at offset 0 (No DataType is specified for Counters).

Offsets

It is important to note that offsets are absolute. **IW0** and **IW1** share a byte. To get two consecutive, non-overlapping words you need to address **IW0** and **IW2**.

Bits

Bits are addressed by using the Bit data type (X) and appending `.bit` to the end, where bit is in the range [0-7]. When addressing a bit at a given offset, that offset is always treated as a byte.

Strings

Strings are assumed to be in the S7 string format and have a max length of 210.

Timers

Timers are scaled up to a DWord and converted from S5 time format so they can represent the time in milliseconds without requiring any multipliers. When you write to a timer it is automatically converted from milliseconds into S5 time format for you. A data type is not specified when accessing timers.

Counters

Counters in the PLC are stored in BCD. The driver automatically converts to/from BCD for you and exposes any counter Tags as UInt16 values. A data type is not specified when accessing counters.

Device Settings

General

Setting	Description
Name	The name of the Device Connection.
Description	A description for the Device Connection. The description will appear on the Devices page on the Gateway.
Enabled	Whether or not the connection is active. Disabling this setting terminates communication with the device.

Connectivity

Setting	Description
Hostname	The hostname or IP address of the device.
Local Address	New in 8.1.8 The local address to connect from when establishing a TCP connection. If left blank, then the driver will simply pick an available address.
Timeout	The request timeout, specified in milliseconds. The default is 2,000.

Advanced

Setting	Description
Port	The port to use when connecting to the device. The default is 102.
PDU Size	Number of bytes to fit into PDU block of a single packet. Increasing this number can improve request throughput only if the processor supports a higher PDU Size. Varies from 240 and up to 960 depending on the device. The default is 240.
Rack Number	The number of the rack that the device is positioned in. The default is 0.
CPU Slot Number	The slot number assigned to the CPU. The default is 2.
Reconnect After Consecutive Timeouts	After several consecutive timeouts, the Device Connection will attempt to reconnect to the Device. Default is true.

UDP and TCP Driver

Ignition's UDP and TCP drivers allows Ignition to communicate to various devices like barcode scanners, and scales. These are not catch-all drivers that will talk to any PLC that communicates over TCP, but rather very basic drivers that will communicate over TCP or UDP. The drivers are configured to connect to one or more ports on a given IP address and bring in any data there (ASCII characters, etc) as a value of a tag. Both drivers can act as strictly passive listeners: meaning they do not write back or make any requests to the device. This is very powerful for the devices that are designed to function in this way (like barcode scanners) because the device just needs to constantly update data on a port for this driver to work.

The TCP driver has the option of writing back to the device. When configured for Writeback, a tag is exposed in Ignition's OPC server that will handle writing: any writes made to the tag are sent to the device.

Structure in the Address Space

A device using the UDP or TCP driver appears in the **Devices** folder of the OPC-UA server with the name it was configured to use. Browsing the device will yield one folder per port configured to listen on. Browsing the port folder will yield one variable node containing the entire message received as well as an additional variable node per field configured. A device configured with a field count of four would have five nodes total: one for the message and four for the fields.

Connecting to a Device



INDUCTIVE
UNIVERSITY

Connecting to TCP Device

[Watch the video](#)

Instead of connecting to a device directly, this driver will connect to a port (often on the host computer or a computer connected directly to the device), and that device will be configured to post data to that same host/port. Rules are configured that dictate how the incoming data is interpreted. You can configure multiple ports for each device connection.

Connect to a Barcode Scanner or Scale

You can connect to a barcode scanner or scale by using Ignition's UDP and TCP driver.

1. Go to the **Config** section of the Gateway Webpage.
2. Scroll down and select **OPC UA > Device Connections**.
3. On the Devices page, click on **Create new Device**.
4. On the Add Device Step 1: Choose Type page, scroll all the way down and select **TCP Driver**, and click **Next**.
5. On the New Device page, leave all the default values and type in the following fields.
 - Name: The name you specify here will appear under the **Devices** folder on the Quick Client page in the Gateway.
 - Port(s): **12345**, as an example
 - Address: type the **IP address**, for example 10.20.3.456

Config > Opcua > Devices

General	
Name	Scale
Description	
Enabled	☒ (default: true)

Connectivity	
Port(s)	12345 The port(s) to connect to.
Address	10.20.3.456 The IP address to connect to.
Inactivity Timeout	0 The number of milliseconds without receiving data from the source before a disconnect/reconnect is made. Set to 0 to disable. (default: 0)

6. You can check the box for **Show advanced properties?** to see the additional settings, but you can keep all the defaults.

7. Click **Create New Device**. The Devices page is displayed showing the Scale device is successfully created and added to Ignition. The **Status** will show as **1/1 Connected**.
8. Go to the **OPC Connections > Quick Client** in the Config section of the Gateway Webpage.
9. Under the **Devices > [Scale] > 12345** folder you will see the Last Receive Time and the Message folders. Next to each tag, under the **Action** column, you will see `[s][r][w]`.
10. Click on `[s]`, which means Subscription. You will be able to see the Value of the tag displayed on this OPC Quick Client page.

Device Properties

The properties on the **New Device** page of the Gateway for the TCP and UDP devices are as follows:

General

Property	Description
Name	Name of the device using this driver. This name will appear in the Devices folder when browsing the OPC UA server.
Enables	When selected, the device is enabled. When not selected, disabled devices will not make a connection attempt.

Connectivity

Property	Description
Port(s)	On the UDP driver, this is the port(s) to listen on. On the TCP driver, this is the port(s) to connect to. Separate multiple ports with a comma.
Address	On the UDP driver, this is the IP address to listen to. On the TCP driver, this is the IP address to connect to.
Local Address	New in 8.1.8 The local address to connect from when establishing a TCP connection. If left blank, then the driver will simply pick an available address.
Connect Timeout	New in 8.1.17 The timeout, in milliseconds, when opening a socket connection to a remote host. Set to 0 to disable.
Inactivity Timeout	The number of milliseconds without receiving data from the source before a disconnect/reconnect is made. Set to 0 to disable.

Message

Property	Description
Message Delimiter Type	Sets the method used to determine how much or what data length constitutes a full message. • Packet Based: Assumes that whatever arrives in one packet, regardless if length or content, is the message. • Character Based: Content is appended to a message buffer until the given character or set of characters arrives, at which point the contents of the buffer are considered the message. • Fixed Size: Content is appended to a message buffer until some fixed number of bytes is received, at which point the contents of the buffer are considered the message.
Message Delimiter	If the message delimiter type is Character Based, this will be the character or set of characters used to identify a message. If the type is Fixed Size, this will be the size used to identify a message. If using Character Based, see Message Delimiters below.
Field Count	The number of fields within a message must be fixed. This property dictates how many fields will be present in each message. When the number of fields received does not match the designated count, all nodes will receive quality BAD_CONFIG_ERROR.
Field Delimiter	This is the character(s) that are used as field delimiters. For example, the message a b c d with a field delimiter of " " would be split into four fields: a, b, c, and d. The field count would have to be set at 4.

Advanced

Both drivers have unique Advanced Properties.

TCP Driver

Property	Description
Write Timeout	New in 8.1.24 Duration, in milliseconds, to wait for writes to complete before returning. The default time is 5 seconds.
Writeback Enabled	Enable writeback capabilities for the device.
Writeback Message Delimiter	The delimiter expected by the device signaling the end of the incoming message. See Message Delimiters below.

UDP Driver

Property	Description
Message Buffer Size	The size of the message buffer in bytes.
Multicast	If the connection should be enabled for multicast.

⚠ TAG POLLING SPEED

Tag values update no faster than the scan class a tag is assigned to. If additional messages are received over TCP or UDP at a faster rate, the tag will not show these additional values.

Generally it is recommended to use these drivers with updates that come through at a 1 second rate or slower. The absolute maximum rate for the TCP driver is 25ms, although most systems cannot reliably execute this quickly. The UDP driver does not have a maximum rate, but a practical maximum rate will depend on hardware, and likely will be significantly slower than 25ms as well. Most "fast" scan rates fall in the 100ms-200ms range.

Message Delimiters

A couple of properties allow you to specify a message delimiter. Some devices may be expecting a control character of some sort, such as a carriage return. The table below lists characters that you can enter into a message delimiter property to signify certain control characters.

Characters	Description
\t	tab
\b	backspace
\n	newline
\r	carriage return
\f	form feed

UDP and TCP Device Tags

When configured, both drivers have explicit sets of tags they use to communicate. The tags are listed below.

TCP Tags

Name	Description
Last Receive Time	A datetime representing when the last message was received.
Message	A string interpretation of the last received message.
MessageBytes	New in 8.1.2 A binary representation of the last received message.
Writeable	Only available when the TCP device connection has Writeback Enabled . Writing to this tag from Ignition will send a write request to the device. The content will be sent as a string.
WritableBytes	New in 8.1.2 Only available when the TCP device connection has Writeback Enabled . Writing to this Tag from Ignition will send a write request to the device. The content will be sent as binary data.

UDP Tags

Name	Description
Last Receive Time	A datetime representing when the last message was received.
Message	A string interpretation of the last received message.
MessageBytes	New in 8.1.2 A binary representation of the last received message.

DNP3

DNP3 is a communication protocol commonly used in water and electric utilities. It connects a central **master station** to multiple remote devices (outstations), creating a distributed network without overwhelming the communication infrastructure. This page covers DNP3 connection information and general protocol requirements. DNP3 is similar to the Modbus protocol in that it is more device agnostic, but it is newer, more robust, and because of that, more complex. Check your device documentation to verify DNP3 support.

NOTE

The DNP3 driver does not support DNP3 Secure Authentication.

DNP3 Drivers

Ignition supports two DNP3 drivers:

- [DNP3 Driver](#)
- [Legacy DNP3 Driver](#)

In versions prior to 8.1.36, the DNP3 driver is now labeled as **DNP3 Driver (Legacy)** in the Gateway. Starting in 8.1.36, the new DNP3 driver is installed by default on new systems, while the Legacy driver remains available as an optional add-on. Although the Legacy driver still functions, it no longer receives updates or support.

Scripting Functions

The DNP3 and Legacy DNP3 drivers use the following syntax for their respective scripting functions to avoid namespace conflicts:

- DNP3 Driver scripting functions: `system.dnp`
- Legacy DNP3 Driver scripting functions: `system.dnp3`

Supported Operating Systems

Both the DNP3 and Legacy DNP3 drivers are compatible with:

- Windows x64
- Linux x64
- Linux arm32
- Linux arm64

i NOTE

The native libraries for both drivers are not currently supported on macOS.

Upgrading from the Legacy DNP3 Driver

To manually upgrade from the Legacy DNP3 driver:

1. Delete or rename the Legacy DNP3 device.
2. Create a new DNP3 device and use the old Legacy DNP3 device name in the new DNP3 device Name field.
3. Fix tag references as necessary.

i NOTE

The DNP3 driver operates differently from the Legacy driver. Review the [DNP3](#) and [Legacy DNP3](#) connection settings carefully after upgrading.

Internal Indicators

Each response from an outstation includes an Internal Indication (IIN) bit field. This field reports various states or error conditions within the device. These IIN bits are exposed as read-only points. Refer to the table below for descriptions of what the exposed IIN bits indicate.

Indicator	Description
Broadcast	A broadcast message was received.
Class 1 Events	Additional Class 1 event data is available.
Class 2 Events	Additional Class 2 event data is available.
Class 3 Events	Additional Class 3 event data is available.
Need Time	The outstation requires time synchronization.
Local Control	One or more outputs are in local (manual) control mode.
Device Trouble	An abnormal condition exists within the device.
Device Restart	The outstation has restarted.
No Func Code Support	The requested function code is not implemented.
Object Unknown	The requested object is unknown or unsupported.
Parameter Error	There was an error with one or more request parameters.
Event Buffer Overflow	The outstation's event buffer has overflowed.
Already Executing	A previously requested operation is still executing.
Config Corrupt	The outstation's configuration is corrupted.

Terminology

Term	Definition
Unsolicited Response	A message sent from the outstation to the master without a direct request. This behavior is enabled when the master allows unsolicited reporting of points.
Integrity Poll	A request from the master that gathers all current event data (Classes 1–3), followed by the static data (Class 0) for each configured point.
DNP3TIME	A timestamp format in UTC, measured as the number of milliseconds elapsed since January 1, 1970. As of 2008, this format is the required time base for all DNP3 timestamps.

Aliased Points

Aliased points let you assign meaningful names and folder paths to DNP3 points. They are especially useful for addressing points that were not returned during the initial integrity poll.

 NOTE

The DNP3 driver does not support aliased points. This feature is available on the [Legacy DNP3 driver](#) only.

Property	Description
Point Address	The group, variation, and index that fully describe a point. A full address includes: • An integer prefixed with g for group (e.g., g40) • An integer prefixed with v for variation (e.g., v2) • An integer prefixed with i for index (e.g., i5) Example: g30v1i20
Browse Path	A / separated folder path where the point should appear. Example: Facility1/Voltage
Description	(Optional) A user-defined description of the point.

NOTE

Tags created from these aliases may need to have their OPC Item Path updated if the **Point Address** is later changed in the **Aliased Points** configuration.

Buffered Events

Buffered events allow missed events to be preserved and replayed after a disconnect. This feature can be enabled by setting the Queue Size property on the corresponding **Tag Group** to a value greater than 1.

The Queue Size determines how many events the driver will buffer per tag. The driver retains only the most recent events, dropping older ones when the buffer is full. For example, if the Queue Size is set to 20 and 30 value changes occur while the device is disconnected, the 10 oldest events will be dropped.

When the connection is restored, the driver plays back buffered events in order from oldest to newest. This updates the corresponding Ignition tag for each event, which can trigger value changes in systems such as:

- Alarms (including recent events and Alarm Journal records)
- Tag History (when the History **Sample Mode** is set to **On Change**)
- Tag Event Scripts

Once all buffered events are replayed, the tag resumes displaying its live value.

Support of this Feature

This Buffered Events feature can only be used by devices that support Sequence of Events (SoE), and have the option enabled for points Ignition is subscribed to. Note that the data type for SoE may differ from the data type that is received by normal subscription. This is generally due to a configuration on the Default Variation property in the device.

Browsing DNP3 Points



INDUCTIVE
UNIVERSITY

About DNP3 Addressing

[Watch the video](#)

When the driver connects to an outstation, it performs an integrity poll. Any DNP3 objects returned are mapped to OPC items using their group, variation, and index values.

To see these points, open the [Connected Devices Window](#) in the Designer.

Type Name	Group	Supported Variations
SingleBitBinaryInput	1	1 - Packed format 2 - With flags
DoubleBitBinaryInput	3	1 - Packed Format 2 - With flags
Binary Output	10	1 - Packed Format 2 - With flags
Counter	20	1 - 32-bit with flags 2 - 16-bit with flags 5 - 32-bit 6 - 16-bit
FrozenCounter	21	1 - 32-bit with flags 2 - 16-bit with flags 5 - 32-bit with flags and time 6 - 16-bit with flags and time 8 - 32-bit 10 - 16-bit
AnalogInput	30	1 - 32-bit with flags 2 - 16-bit with flags 3 - 32-bit 4 - 16-bit 5 - Float with flags 6 - Double with flags
FrozenAnalogInput	31	1 - 32-bit with flags 2 - 16-bit with flags 3 - 32-bit with time of freeze 4 - 16-bit with time of freeze 5 - 32-bit 6 - 16-bit 7 - Float with flags 8 - Double with flags

Type Name	Group	Supported Variations
AnalogOutput	40	1 - 32-bit with flags 2 - 16-bit with flags 3 - Float with flags 4 - Double with flags
OctetString	110	0 - 225

DNP3 Driver

New in 8.1.36

The DNP3 driver uses the **DNP3** protocol for event-based polling, unsolicited messaging, and explicit reads to acquire data. For the previous driver, see [Legacy DNP3 Driver](#).

NOTE

The DNP3 driver does not support DNP3 Secure Authentication.

Data Acquisition Methods

The DNP3 driver offers three distinct methods for acquiring data: event-based polling, unsolicited messaging, and explicit reads.

Event-Based Polling

Event-based polling is the primary method of data acquisition for the DNP3 driver. Each event class (1, 2, and 3) can be configured with polling intervals. At each interval, the driver polls for all events that occurred since the previous poll. If multiple events exist for a point, they are preserved and processed sequentially. OPC UA clients with MonitoredItems set to a sampling interval of 0 will receive the full sequence of events without sampling delay. The sampling interval does not affect the class polling intervals.

Unsolicited Messaging

Unsolicited messaging is an event-driven method of acquiring data in the DNP3 driver. Instead of polling each event class at configured intervals, the outstation automatically reports event updates whenever changes occur. The client's sampling interval does not influence the outstation, but it does affect OPC UA MonitoredItem and Subscription behavior. OPC UA clients with MonitoredItems configured with a sampling interval of 0 will receive the full reported sequence of events for each point. Unsolicited messaging must be configured for each event class in the driver settings. Not all outstations support unsolicited messaging, so review your device's capabilities before enabling it.

Explicit Reads

Explicit reads allow points to be read directly using the DNP3 Read function code, returning the static or current value of each point at each poll. Events are not read or processed when using explicit reads. This method is typically used for points addressed by group/variation/index (gvi) syntax. When MonitoredItems are configured, points are polled at the client's requested sampling interval.

Connecting to a Device



INDUCTIVE
UNIVERSITY

Connection to DNP3 Devices

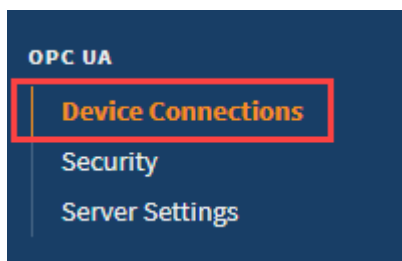
[Watch the video](#)

CAUTION

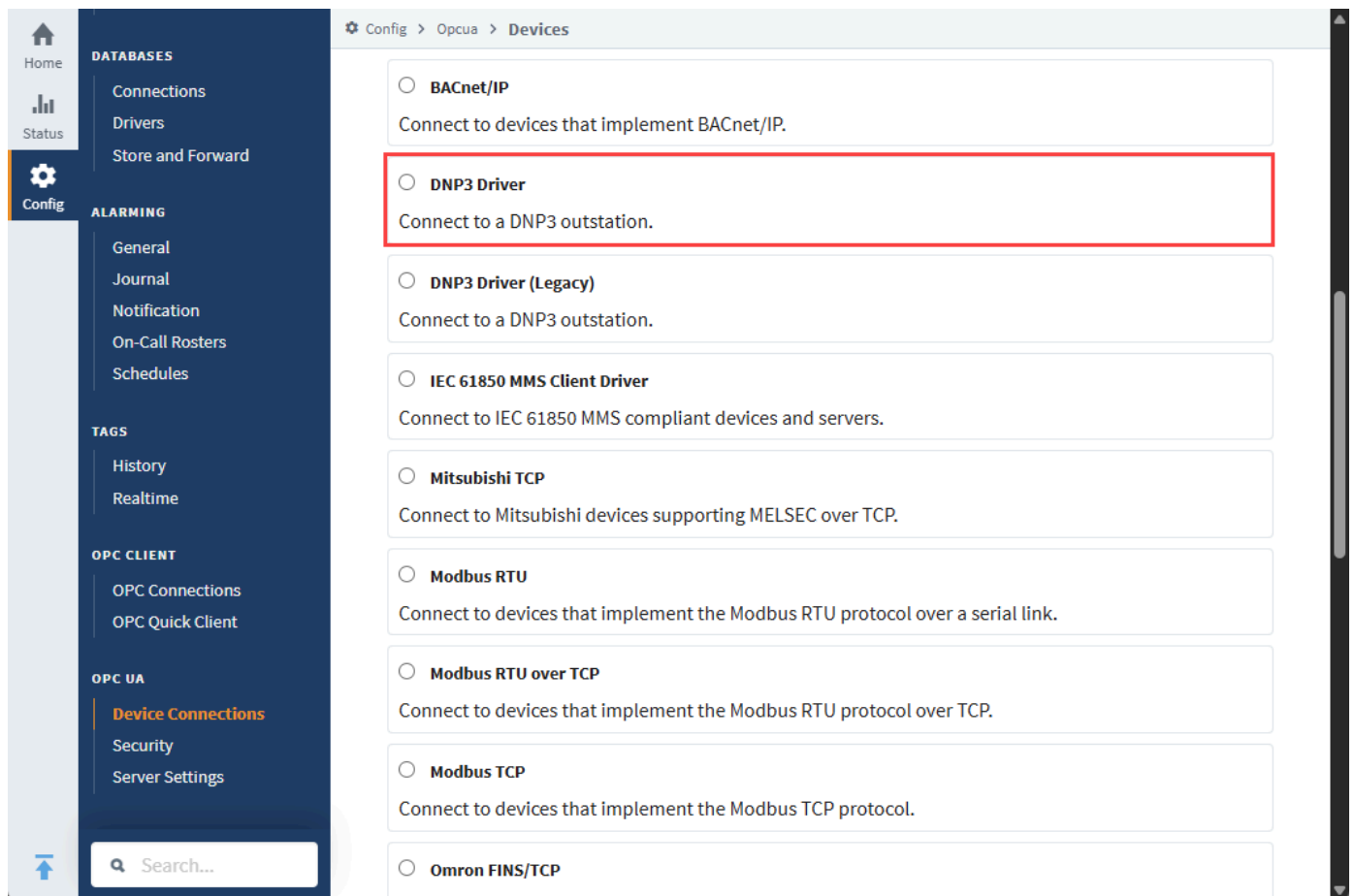
This module requires the OPC UA module to be installed and enabled. If the OPC UA module is not installed or is disabled, this module will fault with a warning in the logs and display the state as "Faulted Missing Dependency" on the Config > Modules page.

The DNP3 driver connects to devices that support Ethernet communication through a master station. Create a separate device connection for each outstation (remote device) by setting the source and destination addresses in the device connection settings.

1. Go to the **Config** section of the Gateway Webpage.
2. Select **OPC UA > Device Connections**.



3. Click **Create new Device**.
4. Choose **DNP3 Driver**, then click **Next**.



5. On the **New Device** page, leave all the default values and type in the following fields:

- **Name:** DNP3
- **Hostname:** IP address or hostname of the device.

6. (Optional) Click **Show advanced properties** to see additional settings.

7. Click **Create New Device**.

The **Status** will change from Disconnected to Connected or Idle.

Connection Settings

General

Property	Description
Name	The name of this DNP3 device connection.
Description	An optional description for the connection. You can use this field to add any helpful comments about the connection.
Enabled	If checked (True), the connection is enabled. If unchecked (False), the connection is disabled.

Connectivity

Property	Description
Hostname	The hostname or IP address of the outstation device.
Port	The TCP port used to connect to the outstation. Default is 20000.
Source Address	The DNP3 data link address of the local (master) device. Default is 3.
Destination Address	The DNP3 data link address of the remote (outstation) device. Default is 4.
Response Timeout (ms)	The time to wait for a response from the outstation, in milliseconds. Default is 5000.
Keep Alive Timeout (ms)	The maximum inactivity time, in milliseconds, before a link-layer keep-alive message is sent. Default is 3,600,000.

Data Acquisition

Property	Description
Integrity Poll Period (ms)	How often to perform an integrity poll (requesting Class 1/2/3/0 data). Disabled if set to 0. Default is 3,600,000.
Class 1 Poll Period (ms)	How often to poll Class 1 event data. Disabled if set to 0. Default is 10,000.
Class 2 Poll Period (ms)	How often to poll Class 2 event data. Disabled if set to 0. Default is 10,000.
Class 3 Poll Period (ms)	How often to poll Class 3 event data. Disabled if set to 0. Default is 10,000.
Unsolicited Event Classes	Specifies which event classes are enabled for unsolicited reporting. Leave blank if unsolicited messaging is not being used. Default is blank.

Analog Operation

Property	Description
Command Mode	The command mode for analog outputs. DIRECT_OPERATE or SELECT_BEFORE_OPERATE. Default is DIRECT_OPERATE.
Read After Operate	If enabled, issues an explicit read after operating analog outputs. Useful if the outstation does not generate analog output events. Default is false.
Read After Operate Delay (ms)	The delay between operating an analog output and reading it back, in milliseconds. Default is 0.

Binary Operation

Property	Description
Command Mode	The command mode for binary outputs. DIRECT_OPERATE or SELECT_BEFORE_OPERATE. Default is DIRECT_OPERATE.
Read After Operate	If enabled, issues an explicit read after operating binary outputs. Useful if the outstation does not generate binary output events. Default is false.
Read After Operate Delay (ms)	The delay between operating a binary output and reading it back, in milliseconds. Default is 0.
Trip Close Code	Specifies the trip/close action for binary outputs: NUL, CLOSE, or TRIP. Default is NUL.
Op Type	The type of operation for binary outputs. LATCH or PULSE. Default is LATCH.
Count	The number of operations to perform for binary outputs. Default is 1.
On Time (ms)	The amount of time the output remains active (on), in milliseconds. Default is 0.
Off Time (ms)	The amount of time the output remains inactive (off), in milliseconds. Default is 0.

Advanced

Property	Description
Auto Time Sync	New in 8.1.39 Automatically synchronizes the Gateway's time with the outstation when the NEED_TIME indication is set. Default is true.
Octet Strings Are Strings	New in 8.1.44 When enabled, Octet String points are delivered as String types instead of ByteString types. Default is false.

Logging

Property	Description
Application Layer Debug Logging	Enables debug logging for the DNP3 application layer. Default is false.
Physical Layer Debug Logging	Enables debug logging for the DNP3 physical layer. Default is false.

Sequence of Events

Sequence of events can be preserved using both event-based polling and unsolicited messaging. It's recommended to use a dedicated Tag Group with the following non-default settings:

- **OPC UA Queue Size:** Increase this value based on the maximum number of queued events you expect during normal polling intervals or communication outages. Ignition typically has far more memory available than most outstations, which often have small, fixed-size buffers. The ideal queue size depends on your outstation's capabilities and should be determined from its technical documentation.
- **OPC UA Sampling Interval:** Set the Sampling Interval to 0 to enable event-based reporting for MonitoredItems instead of sample-based updates. This allows multiple rapid events to be reported accurately while maintaining OPC UA compliance.

If sequence preservation is critical, ensure your device supports Sequence of Events (SoE) reporting.

Legacy DNP3 Driver

The Legacy DNP3 driver, previously known as the DNP3 Driver before **8.1.36**, uses the **DNP3** protocol for explicit reads and unsolicited messaging to acquire data. The Legacy DNP3 driver will no longer receive continued support, updates, or fixes. For the updated driver, see **DNP3 Driver**.

NOTE

The Legacy DNP3 driver does not support DNP3 Secure Authentication.

Data Acquisition Methods

The Legacy DNP3 driver offers two methods for acquiring data: explicit reads and unsolicited messaging.

Explicit Reads

Explicit reads are the primary method of data acquisition for the Legacy DNP3 driver.

The DNP3 Read function code is used to read data points directly. Each polling operation retrieves only the current value of each point; event data is not retrieved or processed. Explicit reads are typically used when points are addressed using the group/variation/index (gvi) syntax.

Unsolicited Messaging

Unsolicited messaging is an event-driven method of acquiring data. Instead of polling at regular intervals, the outstation automatically reports changes when they occur. The client's sampling interval does not affect the outstation, but it does affect OPC UA MonitoredItem and Subscription behavior. OPC UA clients with MonitoredItems configured with a sampling interval of 0 will receive the full reported sequence of events for each point.

Not all outstations support unsolicited messaging, so review your device's documentation before enabling it.

Connecting to a Device



INDUCTIVE
UNIVERSITY

Connection to DNP3 Devices

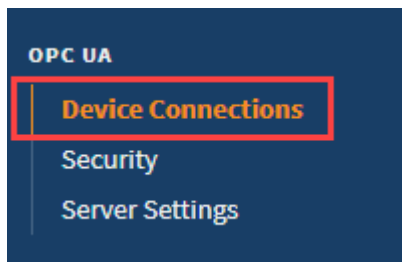
[Watch the video](#)

CAUTION

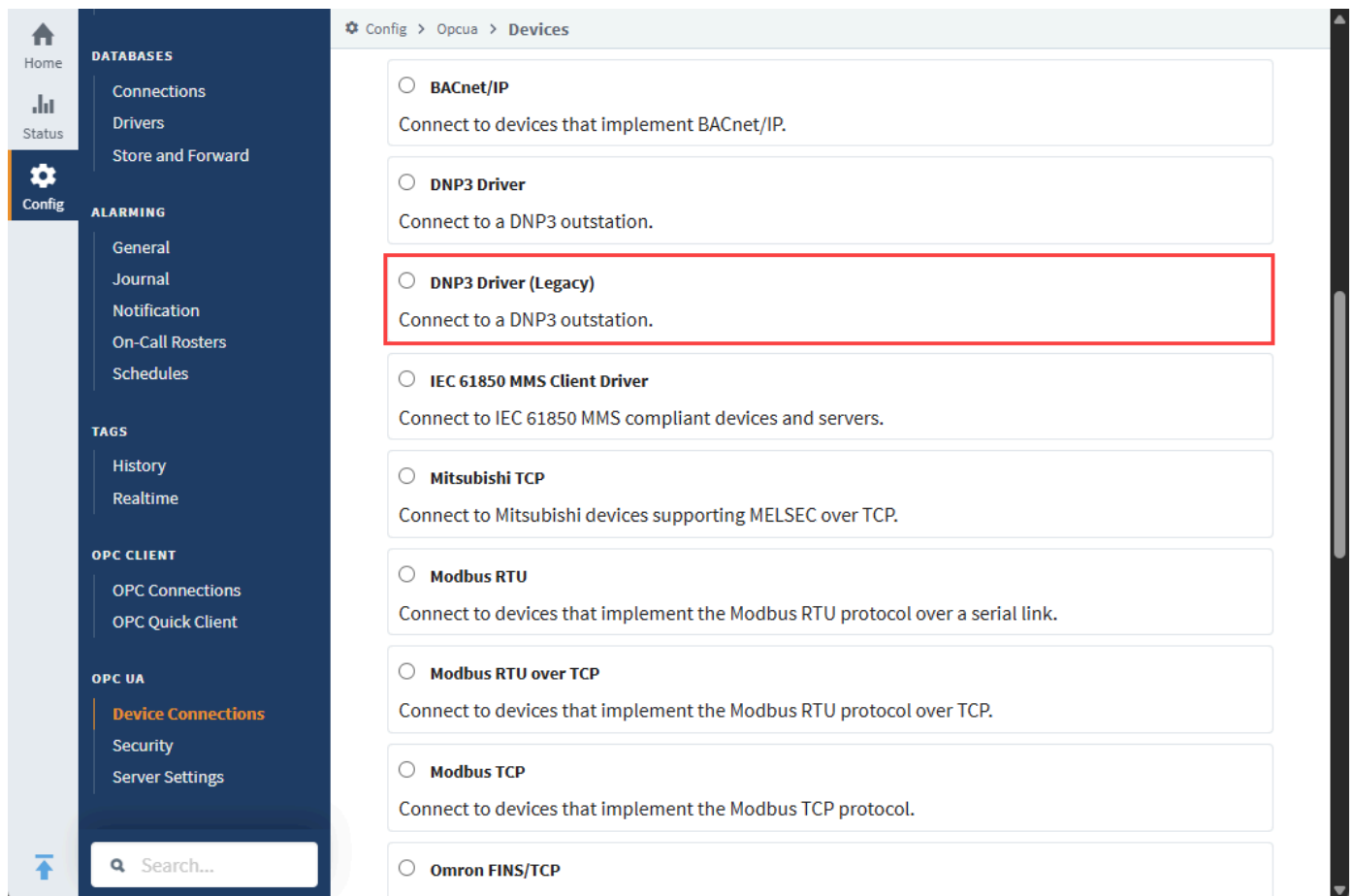
This module requires the OPC UA module to be installed and enabled. If the OPC UA module is not installed or is disabled, this module will fault with a warning in the logs and display the state as "Faulted Missing Dependency" on the Config > Modules page.

The Legacy DNP3 driver connects to devices that support Ethernet communication through a master station. Create a separate device connection for each outstation (remote device) by setting the source and destination addresses in the device connection settings.

1. Go to the **Config** section of the Gateway Webpage.
2. Select **OPC UA > Device Connections**.



3. Click **Create new Device**.
4. Choose **Legacy DNP3 Driver**, then click **Next**.



5. On the **New Device** page, enter:

- **Name:** Legacy DNP3
- **Hostname:** IP address or hostname of the device

6. (Optional) Click **Show advanced properties** to configure additional settings.

7. Click **Create New Device**. The **Status** will change from Disconnected to Connected or Idle.

Connection Settings

General

Property	Description
Name	The name of this DNP3 device connection.
Description	An optional description for the connection. You can use this field to add any helpful comments about the connection.
Enabled	If checked (True), the connection is enabled. If unchecked (False), the connection is disabled.

Main

Property	Description
Hostname	The IP address or hostname of the device.
Port	The TCP port to use when connecting to the outstation. Default is 20000.
Local Address	New in 8.1.8 The local address to connect from when establishing a TCP connection. Leave blank to auto-select.
Source Address	The address of the local (master) station. Default is 3. Note: Source and Destination addresses must be reversed between Ignition and the device. - Ignition settings: Source Address = 3, Destination Address = 4 - Device settings: Source Address = 4, Destination Address = 3
Destination Address	The address of the remote (outstation) device.
Integrity Poll Interval	The interval at which to perform an integrity poll, in milliseconds. Default is 3,600,000.
Direct Operate Enabled	If true, the Direct-Operate function code is used for writes; if false, Select-Operate is used. Default is true.
Unsolicited Messages Enabled	If true, the outstation may send unsolicited messages. Default is false. Not all devices support this mode. Before enabling, check your device's documentation.

Advanced

Property	Description
Message Fragment Size	The maximum size of an application layer message fragment. Default is 249 bytes.
Message Timeout	Response timeout duration, in milliseconds. Default is 5,000.
Retries	The number of retries after timeout. Default is 0.
Time Synchronization Enabled	New in 8.1.17 If enabled, automatically synchronizes Ignition's time to the outstation when requested. Default is true.
Default Outstation Conformance Level	The DNP3 Application Layer subset level to use when communicating with the outstation.

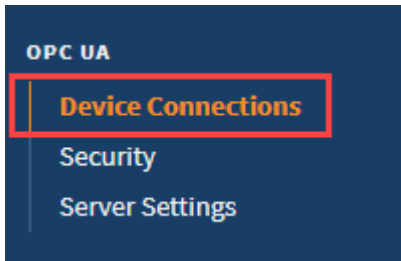
Default Value Types

Property	Description
Analog Input Points	The default value type to use when reading analog input points. Default is INTEGER.
Analog Input Frozen Points	The default value type to use when reading frozen analog input points. Default is INTEGER.
Analog Output Points	The default value type to use when reading or writing analog output points. Default is INTEGER.
Counter Points	The default value type to use when reading counter points. Default is INTEGER.
Counter Frozen Points	The default value type to use when reading frozen counter points. Default is INTEGER.
Binary Input Points	The default value type to use when reading binary input points. Default is WITH_FLAGS.
Double-Bit Binary Input Points	The default value type to use when reading double-bit binary input points. Default is WITH_FLAGS.
Binary Output Points	The default value type to use when reading binary output points. Default is WITH_FLAGS.

Omron NJ Driver

Connect Ignition to an Omron NJ Device

1. Go to the **Config** section of the **Gateway** Webpage.
2. Scroll down and select **OPC UA > Device Connections**.



3. On the **Devices** page, click on **Create new Device**.
4. Select **Omron NJ Driver**, and click **Next**.
5. On the **New Device** page, leave all the default values and type in the following fields:
 - Name: **Omron**
 - Hostname: type the IP address, for example 74.125.224.72
 - Check the box for **Show advanced properties?** to see the additional settings, but you can keep all the defaults.
6. Click **Create New Device**. The **Devices** page is displayed showing the **Omron** device is successfully created and added to Ignition.
7. On the **Devices** page, click the Tags link next to the newly created device. The **Manage Tags** page is displayed, allowing you to configure which variables in the device will show up as Tags in Ignition.

Device Settings

General

Property	Description
Name	Device name.
Description	Description of the device connection.
Enabled	Default is set to true.

Main

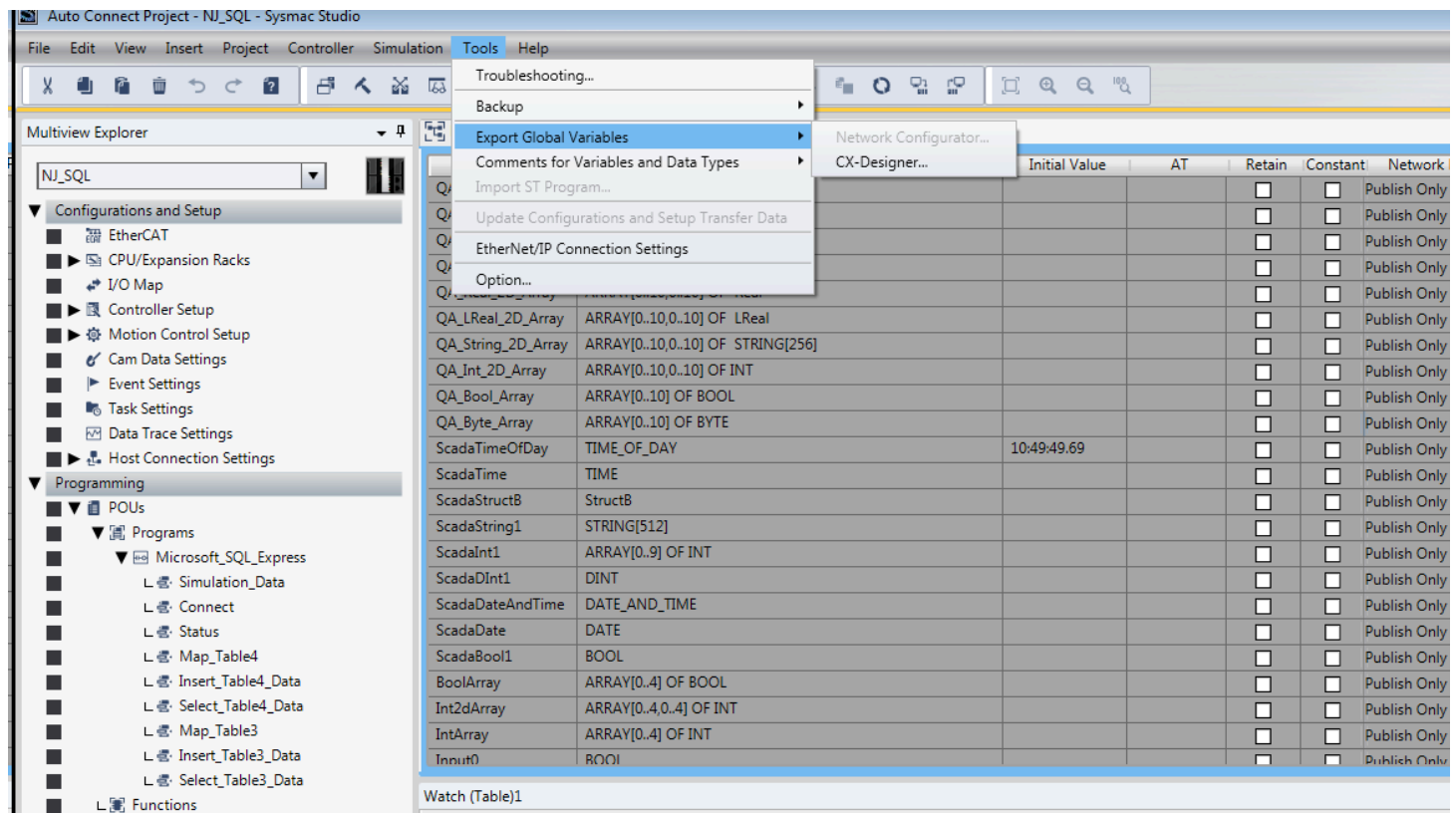
Property	Description
Hostname	The IP Address of the device.
Local Address	New in 8.1.8 The local address to connect from when establishing a TCP connection. If left blank, then the driver will simply pick an available address.
Timeout	The request timeout, specified in milliseconds. The default is 2,000.
Concurrency	The number of concurrently issued requests allowed. There is a 1:1 correlation between concurrency and the number of CIP connections used. The default is 2.

Advanced

Property	Description
Connection Size	The CIP connection size to use. The default (and maximum) is 1,994 bytes.
Date/Time Offset	New in 8.1.1 Offset in hours for Date/Time values. Default is 0. The driver needs an advanced setting for an offset to be applied when reading or writing date/time values.
Slot Number	The slot number in the backplane in which the CPU is located. The default is 0.

Exporting from the Device

To export variables from Sysmac Studio, navigate to the global variables and select **Tools > Export Global Variables > CX-Designer**.



The variables will be saved to the clipboard in tab-separated format. You can now paste the contents into an empty text file for use with importing into the Ignition Gateway.

Managing Tags

In order to browse Tags in the Designer, you must first create a mapping for the device in the Gateway. The **Manage Tags** page can be accessed by navigating to the Omron device and clicking the **addresses** link.

Name	Type	Description	Enabled	Status		
Omron NJ Device	Omron NJ Driver		true	Idle	More ▾	edit
→ Create new Device...					addresses	
					delete	

Importing Tags

Once on the **Manage Tags** page, you can manually enter the Tags, or import them from a tab-separated file. When importing, first choose a file, then click the **Load Configuration File** button to append Tags to the table.

< back

Device Connections / New

Load Configuration File 📄

🔄

📄

Remove All 🗑

Browse Name	Data Type	Characters	Array Elements	ReadWrite
Add Row			Save Configuration	<< < > >>

Once you save any changes made to the Tag mapping, you can view the Tags in the **Connected Devices window** of the Designer.

Addressing

In the **Tags** table of the **Manage Tags** page, we have four columns of configuration per Tag:

- **Browse Name** - The corresponding address of the variable found in the Omron device. Struct members are separated with periods.
- **Datatype** - The datatype of the variable found in the Omron device.
- **Chars** - The maximum number of characters that a String Tag will contain.
- **Elements** - Denotes whether the Tag is considered a scalar or array. See below for more detail on specifying the number of elements to read from the device.
- **R/W** - Specifies read / write access permissions on the Tag.

Support for the following data types is included:

- TIME_NSEC
- DATE_NSEC
- TIME_OF_DAY_NSEC
- DATE_AND_TIME_NSEC

Scalars

Leaving the **Elements** column blank will result in a scalar Tag. When reading from the device, only one element will be requested.

Arrays

Specify the number of elements in an array in the form of **0..N**. The initial index 0 is always included, so an array mapped with 0..9 elements is a 10 element array.

Browse Name	Data Type	Characters	Array Elements	ReadWrite
Scadaint1	INT ▼		0..9	R ▼ Remove

ⓘ OPTIONAL FORMAT

Array elements may also be specified with a single integer representing the last offset. For example, an integer value of **5** is equivalent to **0..5**.

ⓘ NOTE

The number range specified in the Elements field can deviate from the range specified in the device's program. Thus, if an array was configured with a range of 0 - 4, but the mapping on the Ignition Gateway is set to 3 - 7, then the resulting items would be offset as follows:

PLC Program	Tag in Ignition
0	BoolArray_3_
1	BoolArray_4_
2	BoolArray_5_
3	BoolArray_6_
4	BoolArray_7_

This is because the driver always assumes that the lowest configured element on the mapping page matches up with the lowest element in the PLC program. As seen above, this can cause some confusion if the mapping on the Ignition Gateway is configured with a different range.

For this reason, it is **highly recommended** to configure the Elements field on the Ignition Gateway to match the range used in the PLC program.

Note, that this also applies to **Multidimensional Arrays**.

Multidimensional Arrays

Multidimensional arrays are specified in the same way as arrays with each group of indices separated by a comma. For example, entering 0..3,0..3 into the **Array Elements** field specifies two groups of indices.

Strings

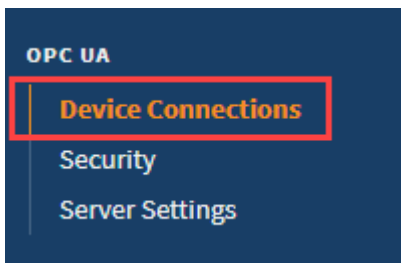
The number of characters for String variables is specified in the **Chars** field. String arrays are mapped using both the **Chars** and **Elements** fields.

Omron FINS Driver

Ignition supports Omron FINS devices. This driver supports both UDP and TCP transport protocols.

Connect Ignition to an Omron FINS Device via TCP

1. Go to the **Config** section of the **Gateway** Webpage.
2. Scroll down and select **OPC UA > Device Connections**.



3. On the **Devices** page, click on **Create new Device**.
4. Select **Omron FINS/TCP**, and click **Next**.
5. Fill in the following fields:
 - Name: **FINS TCP**
 - Hostname: Enter the **IP address**, for example 74.125.224.72

Leave the default values in the remaining fields.

6. Click **Create New Device**.

The **Devices** page is displayed showing the **FINS TCP** device is successfully created and added to Ignition.

Connect Ignition to an Omron FINS Device via UDP

1. Go to the **Config** section of the **Gateway** Webpage.
2. Scroll down and select **OPC UA > Device Connections**.

3. On the **Devices** page, click on **Create new Device**.
4. Select **Omron FINS/UDP**, and click **Next**.
5. Fill in the following fields:
 - Name: **FINS UDP**
 - Bind Address: Enter the **IP address** of your Gateway, for example 74.125.225.10
 - Remote Address: Enter the **IP address** of your Omron FINS device, for example 74.125.224.72
6. Under FINS Settings set the following:
 - FINS Source Node: The last octet of your IP address, for example 10 using the bind address above.
 - FINS Destination Network: The address number of your FINS device typically configured in the routing table for your device. The valid range is 0-128.
 - FINS Destination Node: The node number of your FINS device. This is typically the last octet of the IP address of the device, for example 72 in the example above.
 - Fins Destination Unit: The unit number of your FINS device.
7. Click **Create New Device**.

The **Devices** page is displayed showing the **FINS UDP** device is successfully created and added to Ignition.

Device Settings

General

Property	Description
Name	Device Name
Description	Device description
Enabled	Default is set to true.

Connectivity

Property	Description	TCP or UDP
Hostname	The hostname or IP Address of the device.	TCP Only
Port	The port your Omron device is listening on. The default is 9600.	TCP Only
Local Address	Address of network adapter to connect from.	TCP Only
Bind Address	The hostname or IP address of the Gateway	UDP Only
Bind Port	The port you wish to create a UDP connection on for the Gateway. The default is 9600.	UDP Only
Remote Address	The hostname or IP address of the device.	UDP Only
Remote Port	The port your Omron device is listening on. The default is 9600.	UDP Only
Timeout	The request timeout, specified in milliseconds. The default is 2,000.	UDP & TCP

FINS Settings

Property	Description	Valid Value
FINS Source Network	The address number of the source network (the Ignition Gateway).	Valid value is an integer between 0 and 127. The default setting is 0.
FINS Source Node	The number of the source node (the Ignition Gateway), typically the last octet of the IP address.	Valid value in an integer between 0 and 254. The default setting is 0.
FINS Source Unit	The unit number of the source device (the Ignition Gateway).	Valid value is an integer between 0 and 255. The default setting is 0.
FINS Destination Network	The address number of the destination network (the FINS device).	Valid value is an integer between 0 and 127. The default setting is 0.
FINS Destination Node	The number of the destination node (the FINS device), typically the last octet of the IP address and set via dials on the front of the device.	Valid value in an integer between 0 and 254. The default setting is 0.
FINS Destination Unit	The unit number of the destination device (the FINS device).	Valid value is an integer between 0 and 255. The default setting is 0.

Request Optimization

Property	Description
Concurrent Requests	The number of requests that Ignition will try to send to the device at the same time. Increasing this number can sometimes help with your request throughput, however increasing this too much can overwhelm the device and hurt communications with the device.
Max Request Size	The maximum size of a request in bytes.
Max Gap Size	The maximum address “gap” to span when optimizing requests to contiguous address locations.
Write Priority Ratio	The number of write requests to execute for every read request, when an abundance of both are queued for execution.

Import Addresses

There is a user interface in the Gateway to create import addresses. After import, you'll be able to browse the Tags in the Designer.

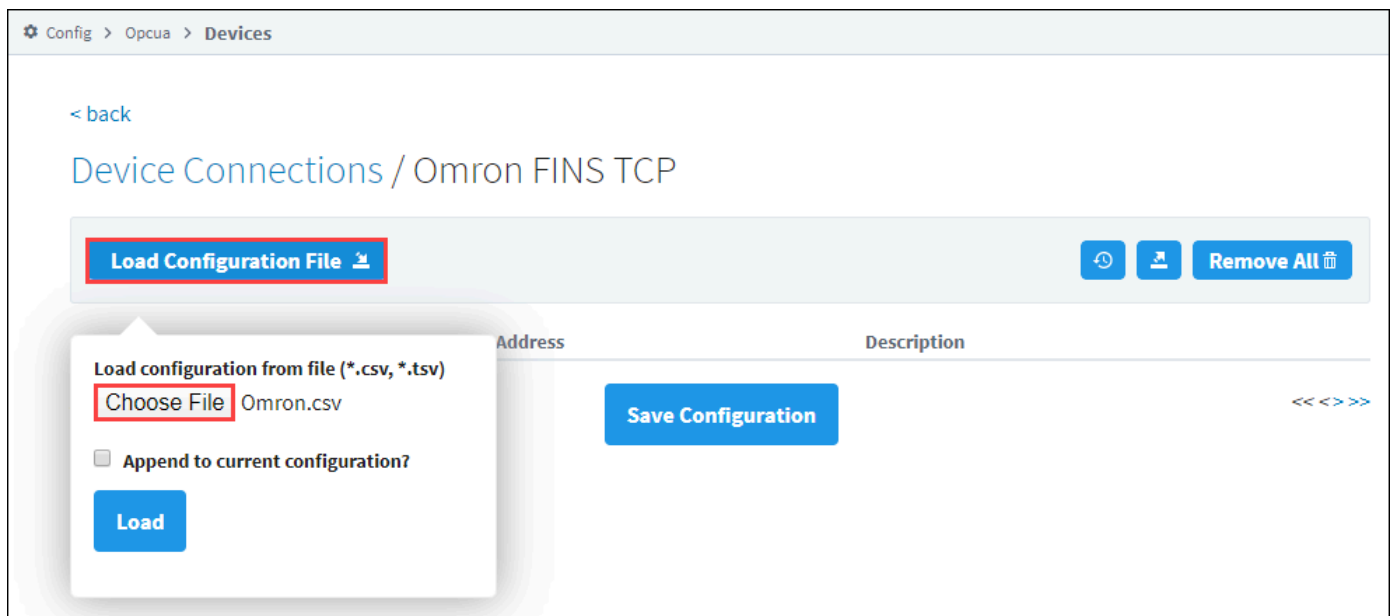
1. Go to the **Config** section of the **Gateway** Webpage.
2. Scroll down and select **OPC UA > Device Connections**.
3. On the **Devices** page, scroll to the **OMRON FINS** device. Click the **More** dropdown and choose **addresses**.

OMRON FINS UDP	Omron FINS/UDP	true	UNBOUND	More	edit
OMRON NJ	Omron NJ Driver	true	Idle	addresses	delete

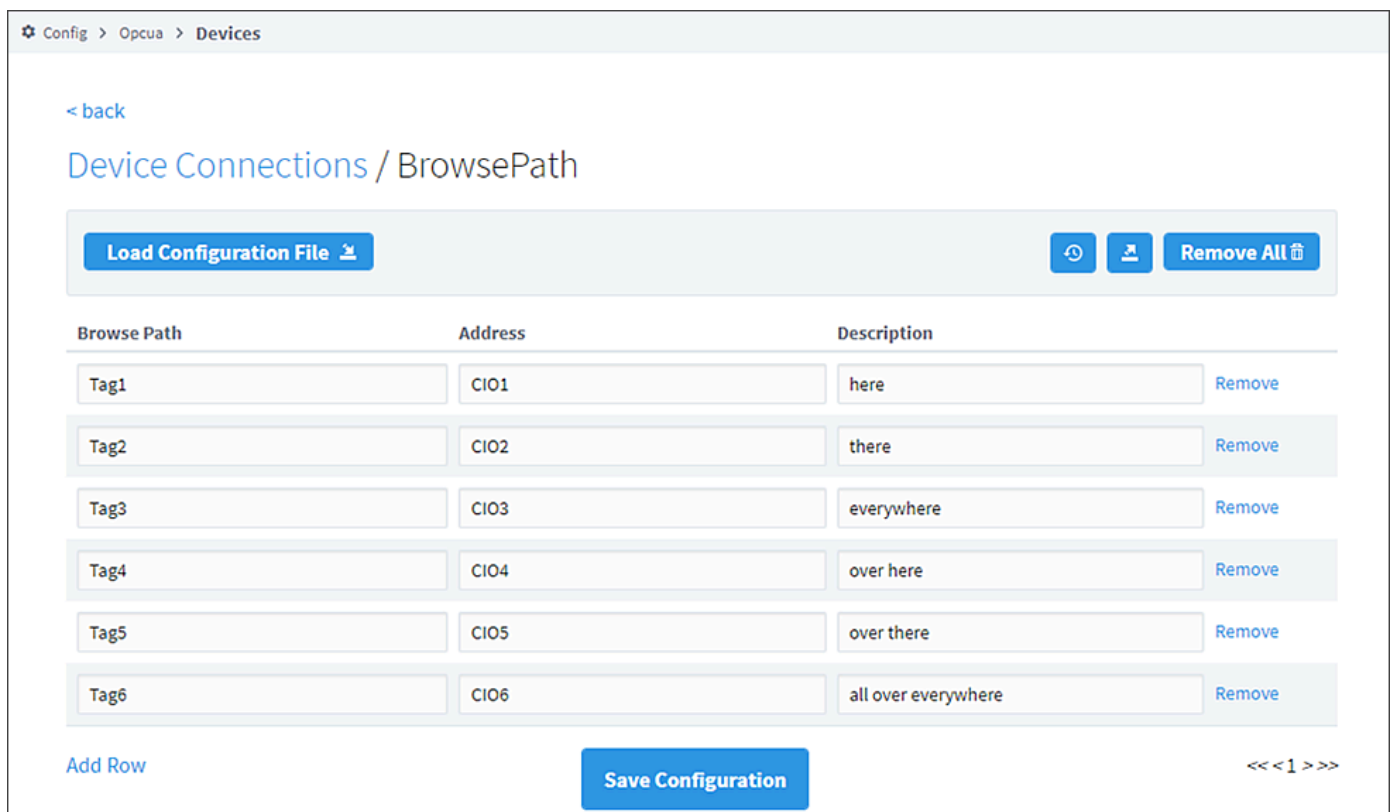
4. On the next page, load the addresses from a CSV, TSV, or XML file. Click the **Load Configuration** button and choose a file.

New in 8.1.2 Release 8.1.2 added support for XML configuration files.

5. Click **Load**.



You'll see the values load onto the screen.



6. Click **Save Configuration**.

7. Once you save any changes made to the Tag mapping, you can view the Tags in the **Connected Devices window** of the Designer.

Addressing

You also have the option to manually address tags via an Ignition OPC tag in the Ignition Designer. See [Ignition OPC UA Server Node Configuration](#) for more information.

Data Areas

Area	Ignition Code	Omron Symbol Code	Native Size
CIO Area	CIO		Int16/UInt16
Work Area	WR or W	W	Int16/UInt16
Holding Bit Area	HR or H	H	Int16/UInt16
Auxiliary Bit Area	AR or A	A	Int16/UInt16
Timer Area	TIM or T	T	Int16/UInt16 or Bool
Counter Area	CNT or C	C	Int16/UInt16 or Bool
Index Register Area	IR	IR	Int32/UInt32
Data Register Area	DR	DR	Int16/UInt16
DM area	DM or D	D	Int16/UInt16
EM area	EM or Exx	E	Int16/UInt16

EM Area

Use `EM` for the current bank, or `Exx` for bank `xx`, where `xx` is 2 hex digits. Valid banks are E00 through E18 (25 total).

Data Types

- Bit
- Bool
- Int16
- Int32
- Int64
- UInt16
- UInt32
- Float

- Double
- String
- BCD16
- BCD32

Data Type Modifiers

Modifier	Order
@BE	Big Endian Byte Order (default when not specified)
@LE	Little Endian Byte Order
@HL	High-Low Word Order
@LH	Low-High Word Order (default when not specified)

Examples

Syntax Example

NOTE

Items in curly brackets {} are optional.

Below is an example OPC Item path, representing how to manually specify an address in the device.

```
ns=1;s=[DeviceName]Area{<DataType>}Offset{.Bit}
```

When typing Addresses in using the Device's Address page on the gateway, the syntax looks like the following:

```
Area{<DataType>}Offset{.Bit}
```

More Examples

Item Path (OPC UA NodeId)	FINS Request
CIO0	An Int16 at offset 0 in CIO area.
CIO<Int32>1	An Int32 at offset 1 in CIO area.
CIO<Int64>3.0	Bit 0 of an Int64 at offset 3 in CIO area.
CIO<Int64@HL>3	An Int64 at offset 3 in CIO area in High-Low word order.
TIM0	Present Value (16-bit) of Timer at offset 0.
TIM<Bool>0	Completion Status (Bool) of Timer at offset 0.
CNT0	Present Value (16-bit) of Counter at offset 0.
CNT<Bool>0	Completion Status (Bool) of Counter at offset 0.
DR0	An Int32 at offset 0 in DR area (32-bit by default).
DR<Int64>0	An Int64 at offset in DR area.
CIO<Int16[3]>0	An Int16 array of size 3 at offset 0 in CIO area.
CIO<Int16[3]>0[1]	Element 1 of an Int16 array of size 3 at offset 0 in CIO area.
CIO<String10>0	String of up to length 10 (string length in bytes) at offset 0 in CIO area.
E001000	An Int16 from EM bank 0 at offset 1000.
E00<Int16>1000	Also an Int16 from EM bank 0 at offset 1000, but with the an explicit DataType to make the address look less confusing.
E001000	An Int16 from EM bank 0 at offset 1000.
TIM0	Present Value (16-bit) of Timer at offset 0.
TIM<Bool>0	Completion Status (Bool) of Timer at offset 0.

Ignition OPC UA Server Node Configuration

You can also create OPC Nodes in Ignition's OPC UA server manually via a CSV file on the Ignition Gateway itself with the following settings:

- File Name: **tags.csv**
- File Location: **\$IgnitionHome/data/opcua/devices/\$deviceName**
- File Format: Comma separated value file with the following columns:
 - Browse Path
 - Address
 - Description

Example File

```
Tag1, CI00, "First tag"
Foo/Tag2, CI01, "Second tag, in a folder"
Foo/Bar/Tag3, CI02, "In a couple of folders"
```

Use quotes on the description when it contains a comma (or whenever you like).

When adding or making changes to this file, a restart of the device is necessary for the changes to be picked up. This is done by editing the device and clicking **Save**.

Programmable Device Simulator

The Programmable Device Simulator allows you to add simulator devices that act as if they are connected to a real device. It allows users to read and write Tags without any network or PLC connection.

The Programmable Device Simulator provides functionality that allows you to create your own simulator program with outputs you define. Users are able to create Tags that can be used in their own development process, thus providing the flexibility to simulate devices with a Tag structure they are building for their project. There are a number of built-in functions that will result in unique Tag outputs rather than static values. We also provide the ability to set specific values at different points of the program should the functions not provide exactly what you need.

NOTE

The Programmable Device Simulator replaced the previous built-in simulators in earlier versions of Ignition, and combined the Dairy Demo, Generic and SLC simulator device connections into a single driver.

Connecting to a Programmable Device Simulator

Connecting to a Programmable Device Simulator and loading the programs are simple and will give you some values that change on their own. These simulator devices can be used for realtime values, history and alarms.

1. From the Gateway webpage, go to the **Config** section.
2. Scroll down and select **OPC UA > Device Connections**.
3. Click on **Create new Device....**
4. Select **Programmable Device Simulator**, and click **Next**.
5. Give the device a name (i.e., GenSim), and click **Create New Device** button.

General	
Name	GenSim 
Description	
Enabled	☒ (default: true)

Simulator Program Settings	
Repeat Program	☒ (default: true)
Base Rate (ms)	1000 (default: 1,000)

Create New Device

6. The window will refresh and you'll see your device was successfully created with a status of "Running". Now you can set the program for the simulator by clicking **More > edit program**.

Config > Opcua > Devices

✔ Successfully created new Device "GenSim"

Name	Type	Description	Enabled	Status	
GenSim	Programmable Device Simulator		true	Running	More  edit edit program delete

[→ Create new Device...](#)

7. Create a program by either clicking the Add Instruction link to add a new line of instruction to the program, or loading a predefined program from the **Load Program** dropdown (i.e., Generic Program) and clicking the **Load Simulator Program** button.

[< back](#)

GenSim

Load Program

Choose One ▾

Choose One

SLC Program

Generic Program

Dairy Simulator

Load from CSV

Export Instruction File

Clear All

Time Interval	Browse Path	Value Source	Data Type
---------------	-------------	--------------	-----------

Add Instruction

<< < > >>

Save Program

8. Once you have some instructions, click the **Save Program** button. These instructions determine what Tags get made, and what their values are.

< back

✓ Program loaded successfully.

GenSim

Load Program

Generic Program ▼

Load Simulator Program

Export Instruction File

Clear All

Time Interval	Browse Path	Value Source	Data Type	
0 	Ramp/Ramp0	ramp(0.0, 10.0, 100.0, true)	Double ▼	Remove
0	Ramp/Ramp1	ramp(5.0, 123.0, 230.0, true)	Double ▼	Remove
0	Ramp/Ramp2	ramp(1.0, 2.0, 50.0, true)	Double ▼	Remove
0	Ramp/Ramp3	ramp(-10.0, 10.0, 300.0, true)	Double ▼	Remove
0	Ramp/Ramp4	ramp(0.0, 500.0, 1500.0, true)	Double ▼	Remove
0	Ramp/Ramp5	ramp(0.0, 700.0, 2000.0, true)	Double ▼	Remove
0	Ramp/Ramp6	ramp(0.0, 800.0, 2500.0, true)	Double ▼	Remove
0	Ramp/Ramp7	ramp(0.0, 900.0, 3000.0, true)	Double ▼	Remove
0	Ramp/Ramp8	ramp(0.0, 110.0, 3500.0, true)	Double ▼	Remove
0	Ramp/Ramp9	ramp(0.0, 5000.0, 4000.0, true)	Double ▼	Remove

Add Instruction

<< < 1 2 3 4 5 6 7 8 > >>

Save Program

Simulator Settings

Setting	Description
Repeat Program	If True, the program will repeat indefinitely, meaning that once the last defined Time Interval has been reached, the program will start over again at Time Interval 0.
Base Rate (ms)	The number of milliseconds determining how quickly the program should move between each Time Interval. The initial duration of the Base Rate is defined in the Program Device Simulator settings should you need this value to persist after restarts.

Program Instructions

The simulator gets its values from a Program made up of various instructions. The program runs through the instructions in order and executes. Each Program Device Simulator contains only one program. Each instruction has a Time Interval and Tag path. If there is more than one Time Interval for a Tag, it will step through those values in order. You can alternatively have a single time Interval and a function in the Value Source field.

Term	Description
Time Interval	Represents a “step” or point in time for the program. A program will start at interval 0, and set the Value Source on the Tag. The program will wait for a number of milliseconds (determined by the Base Rate, defined below) before moving to the new Time Interval.
Browse Path	The full path to the Tag you want to create in the simulator. Forward slashes are used to specify folders, as well as act as a delimiter for additional folders in the path.
Value Source	Determines how the value for the node at the Browse Path is generated. Either a static value or a function will be used to generate the value.
Data Type	The Data Type for a node. Valid types are: • boolean • int16 • uint16 • int32 • uint32 • int64 • float • double • string • datetime • uint64 (Not all values full supported by Ignition at this time)

Value Source Functions

Users are allowed to define a 'function' for the value source. This is not a Python or Expression function. Rather, the value in the cell is a string that looks like a function definition; eg., `foo(x,x,x)`. The driver will attempt to parse the string, and then derive a value based on the function. Valid functions are in the following table.

Function	Descriptions
sine(min, max, period, repeat)	Sine wave. The value moves between the min value and max value, and back to the min. The total duration of this wave is based on the period parameter. If no parameters are specified, then the function should produce a wave that starts at 0, reaches an upper bound of 100, and returns to 0. This series should occur over the default period ((10 * Base Rate) * 1000 ms = 10,000 ms).
cosine(min, max, period, repeat)	Cosine wave.
square(min, max, period, repeat)	Square wave.
triangle(min, max, period, repeat)	Triangle wave.
ramp(min, max, period, repeat)	Ramp signals starts from the min value going up to max value based on the period parameter. When the value reaches its upper limit, it is reset to the min value.
realistic(setPoint, proportion, integral, derivative, repeat)	A realistic number generator driven by a PID system.
random(min, max, repeat)	Random values starting at the min value and working up to the max value. Changed in 8.1.38 For Boolean tags, the min and max parameters are unused. They are still required, but ignored.
list(value1, value2, etc..., valueN, repeat)	Accepts any number of parameters and walks through each value in this list.
qv(value, qualityCode)	Sets a value and quality code. Allows users to simulate a value with bad quality.
readonly(value)	Sets a static value for read only purpose.

Function	Descriptions
Each of the functions above expect certain parameters.	

Parameter	Description	Default Value
min	Minimum value for the function.	0
max	Maximum value for the function.	100
period	Represents a period of time as a number of Time Intervals. The default value of 10 represents "10 Time Intervals."	10
setPoint	The setpoint of the PID Control Loop. Also known as the desired position.	100
proportion	The proportional value of the PID Control Loop.	1.2
integral	The integral value of the PID Control Loop.	0.06
derivative	The derivative value of the PID Control Loop.	0.25
qualityCode	Allows users to set the quality on a Tag (assuming the function used contains a quality code parameter). Valid values are: Good, Bad, Uncertain.	Good
repeat	Does the function repeat for each time slice of the program, or does it report a single value when the instruction is run.The default value of true will ensure that a function will continue to run as long as the program itself is running. A value of false will only update a value when it reaches the specified Time Interval in the program.	true

Example Instruction Files

Simple Function Example

TimeInterval	BrowsePath	ValueSource	DataType
0	TagA	sine(0,100,10,true)	Double

The table above shows us function usage with parameters. Because the example is using default values, it is functionally equivalent to the following:

TimeInterval	BrowsePath	ValueSource	DataType
0	TagA	sine()	Double

Running this program should yield the following results (assuming repeat is enabled):

1. A Tag at root named “TagA” will be created.
2. Since only a single Time Interval was defined, the program ends quickly. The delta time between the start of the program and the end should be around one Base Rate interval.
3. The simulator will use a internal clock to ensure that the value reported is representative of the actual period for the function requested and if history is enabled, a full wave would be graphed.

Advanced Program Example

TimeInterval	BrowsePath	ValueSource	DataType
0	TagA	sine()	Double
0	myfolder/TagB	0	Int32
3	myFolder/TagB	4	Int32
4	another_folder/TagC	100	Int32
20	TagA	ramp()	Double
30	myfolder/TagB	55	Int32

The program above should yield the following results:

1. TagA is initialized with the Sine function.
2. TagB (which is located under **myFolder** in the simulator's structure) is initialized with a value of 0.
3. TagC isn't defined at the start, but that doesn't matter; the program creates a TagC initially because it will exist in the device. If this is the first time the program ran, then TagC started with a value of 0. If it is the second time through, it maintains the value at the end of the program until it reaches an interval where something changes.
4. Time Interval 1. The program doesn't state that any changes should be made, so we're done evaluating this Time Interval.
5. Time Interval 2. Again, nothing to do here, so we wait another Base Rate duration.
6. Time Interval 3, the value on TagB to 4.
7. Time Interval 4. The value of TagC changes to 100. If this program has executed at least once before, then TagC could already have a value of 100, since this is the only entry in the program that changes the value on Tag C. However, it is possible that the user (or something else, such as a binding or script) wrote another value to this Tag. At this point, our program is setting the value back to 100
8. Time Interval 20, TagA switches to the Ramp function. This whole time it has been using the Sine function to generate some moving numbers, but now we're telling it to use the Ramp function, which change the value (starting with the minimum value for the function) and uses a different method to determine its value.
9. Time Interval 30. The value of TagB changes to 55.
10. We're now at the end of the program. If Repeat was enabled, we'll move back to Time Interval 0 and start the whole process over again. If not, then the program ends for all Tags.

Preloaded Programs

The Programmable Device Simulator comes with preloaded programs for Generic, Dairy and SLC simulators, as well as the ability to load from a CSV.

CAUTION

When loading any of the default programs (Generic, Dairy, or SLC), the item paths on the tags generated will contain a **_Meta:** prefix. This prefix exists for legacy purposes to ensure that users who replace simulators created in 7.x with the new programmable simulator will not have to recreate tags.

However, this also means that if you later export these tags as a CSV, the **_Meta:** prefix is removed since it is not clearly shown in the browse path. This means that once you are done adding or removing tags in the exported CSV and want to import again, the old tags may no longer work. To fix the old tags, you can either recreate the OPC tags or manually add **_Meta:** before the value set in the Browse Path column.

Example Browse Path

```
ns=1;s=[Testing]_Meta:Writeable/Agitator1Mode
```

Generic Simulator Program

The Generic Simulator Program provides a variety of Tags that offer different data types and value generation styles. For example, there are ramps, sine waves, and random values. Additionally, there is a set of static writable Tags whose values will persist while the device is running.

Dairy Simulator Program

The Dairy Simulator Program has a ControlLogix like structure with Compressor, Tank, Motor Tags and more. The folders are split into an Overview and Refrigeration section with Tags multiple levels deep that mimic a UDT in a ControlLogix device.

SLC Simulator

The SLC Simulator Program creates a simple device whose address structure mimics a basic SLC structure. These Tags are readable and writeable.

Load From CSV

The Load from CSV option allows you to select a CSV file to load a predefined program from. The CSV file must have four columns in a specific order (Time Interval, Browse Path, Value Source, Data Type), with each row representing a different instruction in the program.

Meta Tags

Each Programmable Device Simulator will have Meta Tags that allow users to interact with the program from the Designer/Session/Client. All Tags will be listed under the parent folder **[Controls]** within the device. Altered Meta Tag values do not persist after restart.

Term	Description
Base Rate	The number of milliseconds determining how quickly the program should move between each Time Interval. The initial duration of the Base Rate is defined in the Program Device Simulator settings should you need this value to persist after restarts.
Pause	Setting this Tag to 'True' will stop the program at its current Time Interval. Setting it back to 'False' will cause the program to resume from that point.
Program Counter	A count that tracks the Time Interval of the currently running program. This counter continues to increment for the duration of the program and resets when a program repeats.
Repeat	If True, the program will repeat indefinitely, meaning that once the last defined Time Interval has been reached, the program will start over again at Time Interval 0.
Reset	If True, the program is immediately reset to Time Interval 0 and the value will immediately change back to ' False.'

Export Instruction File

The ability to create custom instructions in the driver was added providing an opportunity to create a custom simulator device. You have the option of exporting the instruction file to a CSV formatted file and loading it once it's modified. You also have the ability to add and remove individual instructions. **Clear All** removes all instructions in the program.

Generic

Load Program

Generic Program

Load Simulator Program

Export Instruction File

Clear All

Time Interval	Browse Path	Value Source	Data Type	
0	Ramp/Ramp0	ramp(0.0, 10.0, 100.0, true)	Double	Remove
0	Ramp/Ramp1	ramp(5.0, 123.0, 230.0, true)	Double	Remove
0	Ramp/Ramp2	ramp(1.0, 2.0, 50.0, true)	Double	Remove
0	Ramp/Ramp3	ramp(-10.0, 10.0, 300.0, true)	Double	Remove
0	Ramp/Ramp4	ramp(0.0, 500.0, 1500.0, true)	Double	Remove
0	Ramp/Ramp5	ramp(0.0, 700.0, 2000.0, true)	Double	Remove
0	Ramp/Ramp6	ramp(0.0, 800.0, 2500.0, true)	Double	Remove
0	Ramp/Ramp7	ramp(0.0, 900.0, 3000.0, true)	Double	Remove
0	Ramp/Ramp8	ramp(0.0, 110.0, 3500.0, true)	Double	Remove
0	Ramp/Ramp9	ramp(0.0, 5000.0, 4000.0, true)	Double	Remove

Add Instruction

<< < 1 2 3 4 5 6 7 8 > >>

Save Program

BACnet

BACnet

Ignition provides a driver for BACnet (a data communication protocol for building automation and control networks). The first step before setting up devices and using the driver is to download the module. Go to the [Ignition downloads](#) page then refer to [Installing or Upgrading a Module](#).

The driver implements BACnet/IP over UDP. Any access to devices on other network media types (Ethernet (ISO-8802-3), MSTP, ARCNET, etc...) must be done through a gateway/router device.

Setup Process

The process for setting up BACnet differs from other drivers in that you have to first set up a local device. After that you can configure a remote device.

1. Set up a [local device](#).
2. Set up a [remote device](#).

Creating Tags



INDUCTIVE
UNIVERSITY

BACnet Addressing

[Watch the video](#)

The BACnet driver offers browsing support, making Tag creation as easy as dragging-and-dropping from the Tag Browser. However understanding how the OPC Item Path is read by the driver can be useful. Using the example below, there are three important parts of the item path we can identify.

```
ns=1;s=[MyRemoteDevice]AnalogInput100.Present_Value
```

- [MyRemoteDevice] is the Remote Device Name
- AnalogInput is the Object Type
- 100 is the Object Identifier

Remote Device Name

The name of the Remote Device, as known by the OPC server.

Object Type

The BACnet object type. Currently supported types are:

- Device
- AnalogInput
- AnalogOutput
- AnalogValue
- LargeAnalogValue
- BinaryInput
- BinaryOutput
- BinaryValue
- MultiStateInput
- MultiStateOutput
- MultiStateValue

New in 8.1.17 Ignition now supports Elapsed_Active_Time and Time_Of_Active_Time_Reset properties on BinaryInput, BinaryOutput, and BinaryValue objects. These tag types contain an ElapsedActiveTime folder with the following properties:

Name	Type	Read/Write
Elapsed_Active_Time	Long	Writable
Time_Of_Active_Time_Reset	Document	Read Only

Object Identifier

The numerical Identifier of the object.

Status_Flags and Document Tags

The Status_Flags Tags on each object are turn a JSON document, which requires parsing before you can see the values on the flags.

Tag Browser			
Tag	Value	Data Type	Traits
Tags			
Data Types			
Present_Value OPC	-147.6	Float	
Status_Flags OPC	{"Value":[0],"ValidBits":[15]}	Document	
System			
All Providers			

While the individual flags can be accessed by manually creating Tags for each flag, in some cases you may want to handle the parsing with an expression instead of creating separate OPC Tags. This can easily be done with the jsonGet functions. Assuming an Expression Tag was created in the same directory as a Status_Flags Tag, we could use the following expression to extract the first bit of Status_Flags (which represents In_Alarm):

```
getBit(jsonGet({[.]Status_Flags}, 'Value[0]'), 0)
```

Writing BACnet Null Values

New in 8.1.4 As of release 8.1.4, you can write a None value (via Python) to a Tag that's subscribing to an address in a BACnet device.

Writing to Unsupported Objects

New in 8.1.18

Scripting functions ([system.bacnet.readRaw](#) and [system.bacnet.writeRaw](#)) now allow users to read from or write to any BACnet object or property. Because the functions accept and return types used by the underlying BACnet4j library instead of translating to OPC UA and into types supported by tags, these functions provide access to objects not explicitly supported by the BACnet driver.

⚠ CAUTION

Please use caution when interacting directly with any BACnet device, as writing to an object may have unintended consequences and could result in data loss.

BACnet Local Device

Before you can start communicating with a remote device you must first configure a local device, representing Ignition's presence as a BACnet device on a network.

Local Devices are configured by specifying a local bind address, port, broadcast address, BACnet network, and BACnet device number. A Local Device can communicate with many remote devices as long as they are reachable on the same IP network.

Device Configuration Strategies

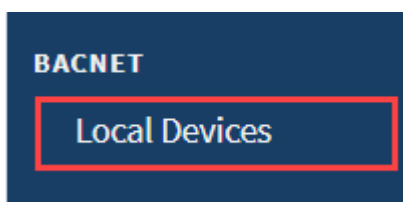
One strategy for configuring local devices is to configure one per network adapter that will be used for communication between the Ignition Gateway and remote BACnet devices.

The local device bind address is configured by default to a wildcard address of 0.0.0.0. This wildcard address is required to receive broadcast packets from any of the remote devices you intend to communicate with. We've found that some devices will only respond to the BACnet Who-Is request with an I-Am that is sent to a broadcast address regardless of whether the original Who-Is was sent unicast or broadcast.

Another situation where you may need multiple local devices is when registration as foreign device with a BACnet Broadcast Management Device (BBMD) is necessary to bridge traffic between networks. Each Local device instance can only be registered with one BBMD.

Configure a Local BACnet Device

1. Go to the **Config** section of the **Gateway** Webpage.
2. Scroll down and select **BACnet > Local Devices**.



3. On the **Devices** page, click on **Create new Device**.
4. Select **BACnet/IP**, and click **Next**.

5. Fill in the following fields:

- Name: **BACnet Local**
- Bind Address: Enter the **IP address** or leave the wildcard 0.0.0.0 default address.
- Bind Port: Enter **local port** or leave default of 47,808.
- Broadcast Address: Enter the **IP address**, for example 10.10.###.##

6. Leave the default values in the remaining fields.

7. Click **Create New Local Device**. You'll see the message "Successfully created new Local Device "BACnet Local."

 Config > Bacnet > Local Devices

 **Successfully created new Local Device "BACnet Local"**

Name
BACnet Local

delete

edit

[→ Create new Local Device...](#)

Now you can configure a driver for the **remote device**.

Device Settings

General

Local Device Setting	Description
Name	Device name.
Bind Address	Changed in 8.1.31 The local IP address to bind to. The default wildcard binding address is 0.0.0.0, which will process broadcast traffic for the given bind port over all interfaces.
Bind Port	This is the port the Device you are setting up will listen on. The local port to bind to. Default is 47,808.
Broadcast Address	A network address shared by other BACnet devices to send and receive UDP data.
Network Prefix Length	Default is 24.
Device Number	Default is 1,000.
Network Number	Default is 1.
Foreign Device Registration Enabled	If true, register as a foreign device with the configured BBMD (BACnet Broadcast Management Device). Default is false.
BBMD Address	Address of the BACnet Broadcast Management Device (BBMD) to register with. If you're planning on having this Local Device configuration work in conjunctions with a Remote Device via BBMD, then the address here should be the address of the BBMD on the same subnet as the Remote Device.
BBMD Port	Port the BACnet Broadcast Management Device to register with is listening on. Default is 47,808.

Advanced

Ignition Redundancy Setting	Description
Backup Device Number	New in 8.1.29 Device number used by the local device on the Ignition redundant backup. Should differ from the device number used on the Ignition redundant primary. Default is 1,001.

BACnet Remote Device

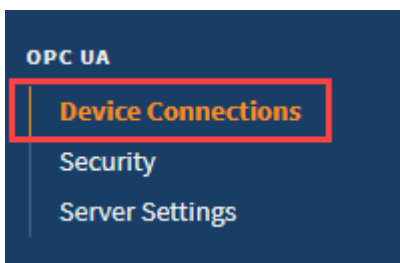
Once a **local device** has been configured a remote BACnet device can be added. You must know the IP address and device number of the remote device to configure a connection.

UDP is a connectionless protocol so a device is considered connected or initialized once Ignition has sent a Who-Is, received an I-Am, and then interrogated the remote device for some basic properties, including its object list and the supported properties of each object.

The object and properties are cached and subsequent connection attempts will only re-read them if the Database_Revision property changes. You can invalidate the browse data under the “More” menu for a given device (Config > OPC UA > Device Connections) on the devices page in the Ignition Gateway.

Connect to a Remote BACnet Device

1. Go to the **Config** section of the **Gateway** Webpage.
2. Scroll down and select **OPC UA > Device Connections**.



3. On the Devices page, click on **Create new Device**.
4. Select **BACnet/IP**, and click **Next**.
5. Select the local device from the dropdown. If no local devices are shown, you'll need to set one up first. See **BACnet Local Device**.
6. Fill in the following fields:
 - Name: **BACnet Remote**
 - Remote Address: Enter the **IP address**, for example 10.10.###.##
 - Remote Device Number: The instance number of the remote device you're setting up.
7. Leave the default values in the remaining fields.

8. Click **Create New Device**.
9. The Devices page is displayed showing the **BACnet/IP** device is successfully created and added to Ignition.

⚙️ Config > Opcua > **Devices**

✔️ Successfully created new Device "BACnet Remote"

Name	Type	Description	Enabled	Status	
BACnet Remote	BACnet/IP		true	Initialized	More ▼edit
CLX	Allen-Bradley Logix Driver		true	Reconnect Wait	deleteedit
DNP3	DNP3 Driver		true	Reconnect Wait	More ▼edit

Device Settings

General

Property	Description
Name	Device name.
Description	Device description.
Enabled	Default is set to true.

Other

Property	Description
Local Device	The name of the local BACnet device instance that will be used when communicating with the remote device. If no local devices are shown, you'll need to set one up first. See BACnet Local Device .
Remote Address	The hostname or IP address of the remote device.
Remote Port	The port that the remote device is listening on. Default is 47,808.
Remote Device Number	The instance number of the remote device you're setting up. Default is 1.
Write Priority	The priority to use when writing to commandable properties. Default is 8.
COV Enabled	If true a Change of Value (COV) subscription is used for properties that support it. If false all properties are polled. Default is true. Note: COV subscriptions are only applied to Present_Value and Status_Flags.
COV Heartbeat Interval	Interval, in seconds, between reading the System_Status property of the Device object as a heartbeat. If three consecutive attempts fail, all COV items will be marked with uncertain quality. Default is 5.
COV Subscription Lifetime	Lifetime, in seconds, of COV subscriptions. Use 0 for indefinite. When non-zero COV subscriptions are renewed at a rate of 75% of the lifetime. Default is 900.
COV Subscription Retry Interval	New in 8.1.33 Interval, in seconds, to attempt recreating COV subscriptions that previously failed to be created or renewed. Use 0 to have failed subscription items remain polled. Default is 120.
Confirmed Notifications Enabled	If true COV subscriptions use confirmed notifications. If false COV subscriptions use unconfirmed notifications. Default is false.
Discovery Timeout	New in 8.1.4 Duration, in seconds, to wait for the remote device discovery process to complete. Default is 5.

IEC 61850 Driver

New in 8.1.25

IEC 61850 is a protocol used for power systems that need to send control signals to different types of intelligent electronic devices (IEDs) defined in a substation. It is similar to the structure of the DNP3 protocol where a single station host can become a bridge to a web of other devices without taxing the network too much. For a more in-depth overview of IEC61850 standards and considerations, refer to the [Understanding IEC61850](#) Knowledge Base article.

The Ignition driver acts as a 61850 Manufacturing Message Standard (MMS) Client. The IEC 61850 driver does NOT support Generic Object Oriented System Event (GOOSE) or Sampled Values (SV) protocols. Check your documentation to see whether your device supports IEC 61850.

NOTE

The IEC 61850 driver only works on Windows and Linux (x64 distributions). ARM chips and macOS platforms are not currently supported.

In addition, the driver requires a modern version of the Visual C++ Redistributable. Read our technical advisory for more details: <https://support.inductiveautomation.com/hc/en-us/articles/13897307938829>

Connecting to a Device



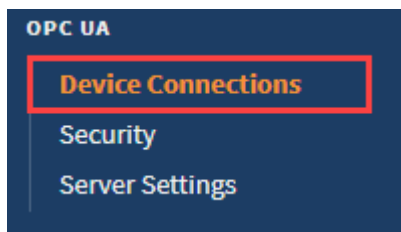
INDUCTIVE
UNIVERSITY

Connecting To IEC 61850
Devices

[Watch the video](#)

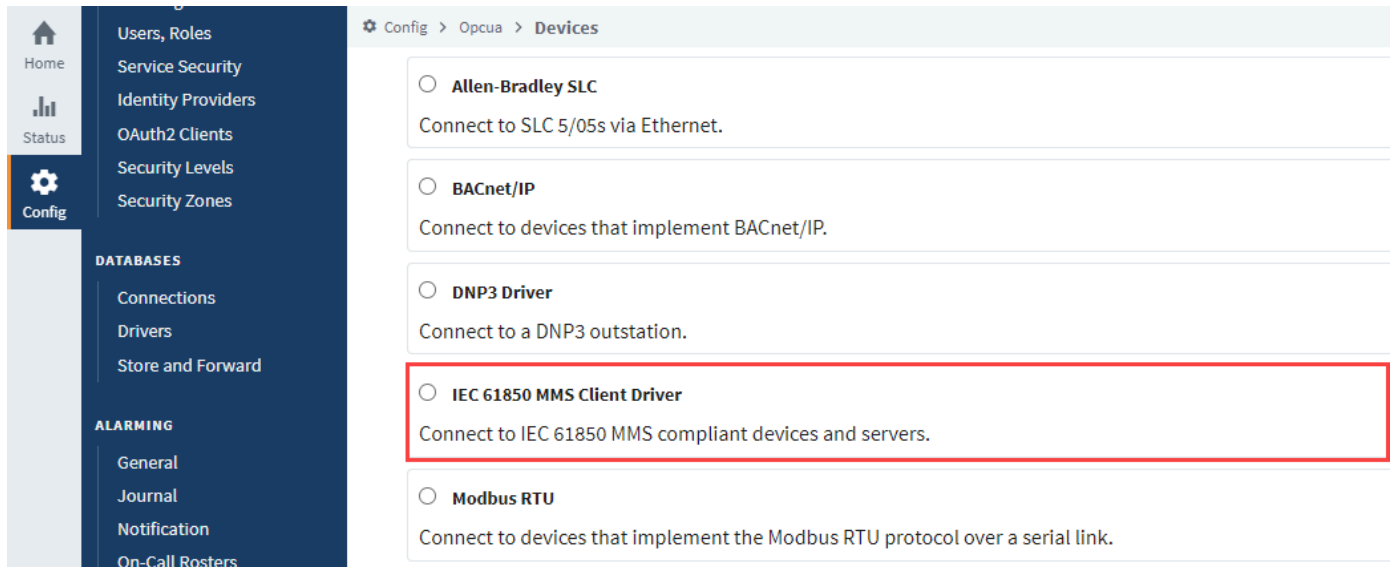
Ignition's IEC 61850 driver can connect directly to any devices that support Ethernet communication through a substation host. It is important to make a new device connection for each of the substation device's IEDs, since only one can be selected per connection.

1. Go to the **Config** section of the Gateway Webpage.
2. Scroll down and select **OPC UA > Device Connections**.



3. Click on **Create new Device**.

4. Scroll down to select the **IEC 61850 MMS Client Driver** and click **Next** at the bottom of the page.



5. Enter a desired name into the **Name** field. In this example, we will be entering **IEC61850**.

6. Enter the IP address or hostname of the device you are connecting to in the **Hostname** field. All other default settings can be left as is.

7. Click **Create New Device**. The **OPC UA > Devices Connections** page now lists the IEC 61850 MMS Client Driver. The **Status** column will show Connected or Idle, depending on the status of the device.

8. On the connected driver, IEC61850 in this example, and click **More > scd**.

i NOTE

Substation Configuration Description (SCD) files grant user control of transfer device descriptions and communication parameters. This file format is the only accepted System Configuration Language (SCL) file type. It is generally recommended to use an SCD file if you intend to limit the request size and visibility of access points during OPC browsing. This will effectively reduce the load against the substation device. Without an SCD file, any device connection restart will request all defined Logical Nodes from your defined IED for OPC browsing, which can become network-traffic heavy.

However, because of how Ignition treats device attributes we recommend **always** using an SCD file, as it may not be possible to resolve associated quality or timestamp values without it.

admin | Log Out →

Help ? Get Designer

Config > Opcua > Devices

Name	Type	Description	Enabled	Status	
IEC61850	IEC 61850 MMS Client Driver		true	Connected	More ▼ edit
Sample_Device	Programmable Device Simulator		true	Running	scd delete

→ Create new Device...

9. Click **Choose File**.

10. Locate your SCD file and select it for upload.

11. Click **Upload**.

Status

Config

Backup/Restore

Ignition Exchange

Licensing

Modules

Projects

Redundancy

Gateway Settings

Config Settings for "IEC61850"

Choose File ABB640_2.scd

Upload

12. Select the IED you want to request for reports.

NOTE

Returning to step 9, the selected IED will now display in the IED Name field and the accessPoint information will now display in the Access Point Name field for the SCL File Settings section of the advanced properties.

Config Settings for "IEC61850"

Select an IED and click 'Save' to load config and restart the driver.



SSSA_52AFA_FPR

accessPoint: AP1

subNetworkName: WA1

Save

13. Click **Save**.

! INFO

The advanced properties will now show the saved SCD file settings. Checking the **Show advanced properties** box on the **Device > edit** page will display the settings. Use SCD File will be checked, and IED Name and Access Point Name will match the selected upload config settings. You can use these settings to verify correct configuration or further tweak your SCD file.

Home

Status

Config

Security Levels

Security Zones

DATABASES

Connections

Drivers

Store and Forward

ALARMING

General

Journal

Notification

On-Call Rosters

Schedules

TAGS

History

Realtime

OPC CLIENT

OPC Connections

OPC Quick Client

OPC UA

Device Connections

Security

Server Settings

BACNET

Config > OpCua > Devices

☒ Show advanced properties

Authentication

Authentication Enabled☐ (default: false)

Authentication Password

SCD File Setting

Use SCD File☒ (default: false)

Use Configured Hostname☐ Override any hostname that may have been set by the SCD file with the configured hostname. (default: false)

Use Configured OSI Params☐ Override any OSI params that may have been set by the SCD file with the configured OSI params. (default: false)

IED NameSSSA_52AFA_FPR

Access Point NameAP1

Advanced Client OSI Parameters

Client AE Qualifier12 Sets the client Application Entity Qualifier. (default: 12)

Use the OPC Quick Client to Confirm Report Statuses

Once you've established a device connection, you'll likely be looking to enable a report so it may be helpful to check the device data before opening your Designer. The OPC Quick Client can be a useful tool for checking report statuses or for troubleshooting purposes. Navigate to **OPC Client > OPC Quick Client** to access the Ignition OPC UA Server folder and locate your IED under the Devices folder. The configured device data is stored in three main folders:

- **Model:** The Model folder contains logical nodes that can be directly polled for information.
 - **Reports:** The Reports folder includes all the available reports and attributes available for subscription.
- NOTE**

Subscribing to any reporting control block attribute will trigger Ignition to attempt to enable the associated report via the **RptEna** attribute. If enabling of a report cannot be achieved you will see quality like `Bad_OutOfService: The source of the data is not operational`.
- **Diagnostics:** The Diagnostics folder contains subfolders that can help identify the status of certain aspects of the configured devices.

If you are unsure of your report subscription status, you can use the Read link **[r]** for the **RptEna** tag within the related Reports folder to confirm if the report is available before you try to enable it. If the value returns as True, that report is already enabled elsewhere and unavailable for use.

Config > Opc > OPC Quick Client

✓ Read completed. [Ignition OPC UA Server]ns=1;s=[IEC61850]SSSA_52AFA_FPRLD0/LLN0.rcbDigitals05.RptEna
Value: false
Quality: Good
Timestamp: 1/30/23, 7:50:18 PM UTC

Object					rcbDigitals01
Tag	[s][r][w]				EntryID
Tag	[s][r][w]				RptEna
Tag	[s][r][w]				RptID

CAUTION

Few events in the IEC 61850 standard allow for writes to occur, and no write actions should be attempted on the Quick Client page. See [Device Operations](#) below for recommended practices to write back to an IED.

Device Properties

General

Property	Description
Name	The name of this IEC61850 device connection.
Description	Device connection description (optional). Can be used to provide any useful information / comments about this connection.
Enabled	If True (checked), the connection is enabled; if False (unchecked), the connection is disabled.

Main

Property	Description
Hostname	The IP Address of the device.
Port	The port to use when connecting to an IEC61850 device, default is 102.
Request Timeout	Determines the maximum amount of time a request to the device will wait for a response, default is 2000.

Advanced Properties

Authentication

Property	Description
Authentication Enabled	Determines if a password will be provided for authenticating requests to the device, default is false.
Authentication Password	The password used to authenticate requests to the device, default is an empty string.

SCD File Setting

Property	Description
Use SCD File	Determines if a SCD file upload will be used when requesting from the device. The default value is false.
Use Configured Hostname	Override any hostname that may have been set by the SCD file with the configured hostname, default is false.
Use Configured OSI Params	Override any OSI params that may have been set by the SCD file with the configured OSI params, default is false.
IED Name	If Use SCD File is enabled, this field will be populated automatically with the selected IED name during the SCD File Upload.
Access Point Name	If Use SCD File is enabled, this field will be populated automatically with the selected IED access point information during the SCD File Upload.

Advanced Client OSI Parameters

Property	Description
Client AE Qualifier	Sets the client Application Entity Qualifier, default is 12.
Client AP Title	Sets the client Application Process Title, default is 1,1,1,999,1.
Client Presentation-Selector	Sets the client Presentation-Selector (P-SEL), default is 00000001.
Client Session-Selector	Sets the client Session-Selector (S-SEL), default is 0001.
Client Transport-Selector	Sets the client Transport-Selector (T-SEL). It is optionally transmitted in the OSI Transport Layer connection request (CR), default is 0001.

Advanced Server OSI Parameters

Property	Description
Server AE Qualifier	Sets the server Application Entity Qualifier, default is 12.
Server AP Title	Sets the server Application Process Title, default is 1,1,1,999,1.
Server Presentation-Selector	Sets the server Presentation-Selector (P-SEL), default is 00000001.
Server Session-Selector	Sets the server Session-Selector (S-SEL), default is 0001.
Server Transport-Selector	Sets the server Transport-Selector (T-SEL). It is optionally transmitted in the OSI Transport Layer connection request (CR), default is 0001.

Misc. Advanced Settings

Property	Description
Use Report Timestamp	Determines if the timestamp from the report will be used instead of the timestamp associated with each reported value.

Reports

Reports can exchange monitoring information, power metering, power quality, and fault event analysis, among other data from IED servers to clients. Device data and reports are sent to the client based on user-defined device configuration through report control blocks (RCBs) and SCL files. The IEC 61850 driver supports Unbuffered/Buffered Report Control Blocks (URCB/BRCB). Remember, an RCB can only be enabled by a single device connection to the IED. This is because an RCB is blocked once a client has registered to receive a report by enabling it. Therefore, if the report is in use elsewhere, an attempted overlap could prevent connections from occurring. You will need to replicate the report on your IED if you want multiple Ignition Gateways to access the same report or leverage a Remote Tag Provider.

- **Buffered Reports:** Reports are buffered by the IED in case a connection to the client is interrupted so that report data is still available to the client when the connection resumes.

- **Unbuffered Reports:** Information is not stored during a connection loss, so any data that may have been sent during the lost connection time is not available. Reporting begins again after the connection has resumed.

USING BUFFERED REPORTS PRIOR TO 8.1.27

Changed in 8.1.27 The autobackup default value, set by the `localdb.autobackup.delay` property, changed from 2 minutes to 60 minutes. This update eliminates the need to modify the autobackup trigger time, and in many cases, the autobackup count as described below.

When using a buffered report, the `config.idb` file will be frequently updated with the last `report_id` received. A short delay, such as 2 minutes, could cause frequent backups and paused threads during the DB backup process. Changes to this property can be made in the [data/gateway.xml](#) file and looks like the following:

```
<entry key="localdb.autobackup.delay">60</entry>
```

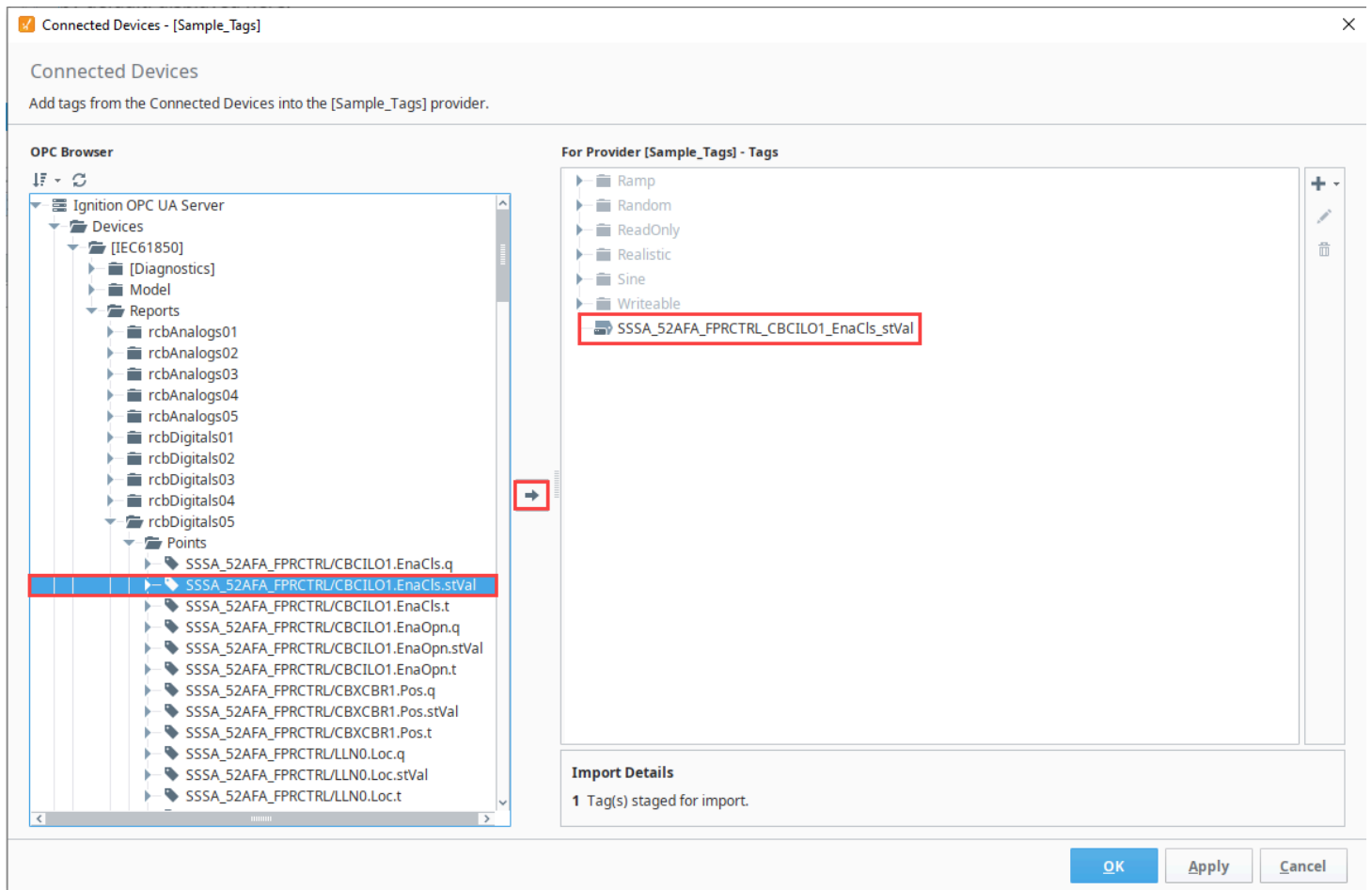
You can also change the `localdb.autobackup.count` property to 0, which will turn off this feature entirely. However, this method is **not** recommended.

```
<entry key="localdb.autobackup.count">0</entry>
```

Use the Tag Browser to Enable a Report

Adding a report folder in your Designer immediately enables a report and allows you to see related attributes in the Tag Browser. This process is done in the same manner as [creating a tag](#).

As you navigate to your desired Logical Node and Points folder, you'll notice a different format than what you might expect from a typical tag in the Tag Browser where all attributes are contained in a single tag. Attributes are listed as separate tags with their own qualified values (e.g. value, quality, and timestamps). This is because the Ignition needs to map the 61850 specification to the OPC UA model that requires an associated value, quality, and timestamp. Ignition will attempt to retrieve the `q` and `t` attributes that correspond with reported value attributes if they are available. Be sure to only add a single folder from the Points folder to your Tag Browser to avoid any issues that can occur from enabling multiple reports.



Report Attributes

Once a specific attribute tag has been added in the Connected Devices window, you'll see your enabled report data in the Tag Browser. You can further curate this data by deleting unwanted data from the browser display.

Tag Browser	
Sample_Tags	
Tags	UDT Definitions
Tag	Value
Random	
ReadOnly	
Realistic	
Sine	
Writeable	
SSSA_52AFA_FPRCTRL_CBCILO1_EnaCls_stVal	☒
AlarmEvalEnabled	☒
CanRead	☒
CanWrite	☒
Deadband	0
Documentation	
Enabled	☒
EngHigh	100
EngLow	0
EngUnit	
FormatString	#,##0.##
HistoryEnabled	☐
Name	SSSA_52AFA_FPRCTRL_C...
OpcItemPath	ns=1;s=[IEC61850]SSSA...
OpcServer	Ignition OPC UA Server
Quality	Good
ReadOnly	☐
TagGroup	Default
Timestamp	2011-04-28 3:30:43 PM
Tooltip	
value	☒
ValueSource	opc ☒

IEC 61850 Hierarchical Data Model

Additionally, the OPC item path that Ignition uses can be isolated to identify individual attribute data because paths are built with an hierarchical data model. The data included in the path identifies the IED, logical device, logical node, data object, and attribute information. The example attribute we've been using in this section is shown below.

```
[SSSA_52AFA_FPR]LD0/LLN0.rcbDigitals05$SSSA_52AFA_FPRCTRL/CBCILO1.EnaCls.stVal
```

- IED: SSSA_52AFA_FPR
- Logical Device: LD0/LLN0
- Logical Node: rcbDigitals05\$SSSA_52AFA_FPRCTRL/CBCILO1
- Data Object: EnaCls
- Attribute: stVal

Disable a Report

Deleting all Points from the Tag Browser in your Designer and/or deleting all report point subscriptions from the OPC Quick Client page will disable a report. You can confirm a report is successfully disabled by using the read from link **[r]** for the corresponding **RptEna** tag on the OPC Quick Client page.

Additionally, if a device is disabled while a subscription is still active, the subscription status code will show `Uncertain_LastUsableValue` instead of the `good` value.

Device Operations



INDUCTIVE
UNIVERSITY

Using IEC 61850 Scripting
Functions

[Watch the video](#)

The IEC 61850 driver supports certain system functions to control operations and perform file services. These functions can be used to discover object data, test or maintain desired statuses, and modify device files. Check out the [system.iec61850](#) page for complete function information.

Control Functions

The supported IEC 61850 system control functions are [getControlParams](#), [Select](#), [Operate](#), and [Cancel](#). The `getControlParams` function returns a list of report control names and their attributes contained in the configured IEC 61850 device. Having a function to collect this data in the moment is useful for many instances, including to provide the required params keys for the `Select`, `Operate`, and `Cancel` functions if they are otherwise unknown.

The type of control model (`ctlModel`) defines how to use the `Select` (if applicable) and `Operate` control functions to write to the connected 61850 device when possible. A Status-only type means you cannot perform any operation actions. Direct types allow an immediate change to an access point using the `Operate` function only. Select before Operate (SBO) types require use of the `Select` function to confirm a chosen access point before the `Operate` function will run. Control model types are represented by the following values:

- 0: Status-only
- 1: Direct-with-normal -security
- 2: SBO-with-normal-security

- 3: Direct-with-enhanced-security
- 4: SBO-with-enhanced-security

The Cancel function will clear a selection of an SBO type control if an operate command is not desired to be executed on the selected control.

Safety Considerations

When working with the IEC 61850 system control functions, it's crucial to understand the potential safety implications. Misuse of these functions can lead to unintended operations and potentially catastrophic results for a high voltage substation.

CHECK PARAMETER, CONTROL INTERLOCK, AND SYNCHROCHECK

The **Check** parameter associated with [system.iec61850.getControlParams](#) is of critical importance for safety. Inside the Intelligent Electronic Device (IED) itself, there are logical nodes for interlocking and synchronizing, named **Control Interlock (CILO)** and **Synchrocheck (RSYN)**. If the **Check** parameter is **00**, the IED will ignore **CILO** and **RSYN** and execute the command even if there is an interlock activated or the grid is not synchronized with the generator speed. The results of not knowing this can be catastrophic for a high voltage substation. It is crucial for an integrator to be aware of this.

SELECT FUNCTION

The [system.iec61850.select](#) function is used to confirm a chosen access point before the Operate function will run. This is especially important for Select Before Operate (SBO) type controls. Misuse of this function can lead to unintended operations. Always ensure that the 'Check' parameter is properly set and understood before using this function, as it can influence the behavior of the **Control Interlock (CILO)** and **Synchrocheck (RSYN)** logical nodes in the Intelligent Electronic Device (IED).

OPERATE FUNCTION

The [system.iec61850.operate](#) function is used to write to the connected 61850 device when possible. However, it should be used with extreme caution. For Direct types, it allows an immediate change to an access point. For Select Before Operate (SBO) types, it should only be used after the Select function has confirmed the access point. Always ensure that the **Check** parameter is properly set and understood before using this function, as it can influence the behavior of the **Control Interlock (CILO)** and **Synchrocheck (RSYN)** logical nodes in the Intelligent Electronic Device (IED).

CANCEL FUNCTION

The [system.iec61850.cancel](#) function will clear a selection of a Select Before Operate (SBO) type control if an operate command is not desired to be executed on the selected control. This function can be used as a safety measure to prevent unintended operations. Always ensure that the **Check** parameter is properly set and understood before using this function, as it can influence the behavior of the **Control Interlock (CILO)** and **Synchrocheck (RSYN)** logical nodes in the Intelligent Electronic Device (IED).

File Functions

The supported IEC 61850 system file functions are [listFiles](#), [readFile](#), and [writeFile](#). These functions allow users to obtain a list of files present on configured devices, and transfer IED files between remote IED servers and local paths. Note that local paths for file functions are local to the Gateway. For example, if these functions get used on a Perspective page, the file will be downloaded to the Gateway, not the client where the browser is running.

Mitsubishi TCP Driver

New in 8.1.28

The Mitsubishi TCP driver provides a user interface in the Gateway where you can create, import, or export addresses when connected to devices that support the MELSEC protocol. After setup, configured tags will be browsable in the OPC Browser of the Designer with read/write access. Supported Hardware Series include:

- iQ-R
- iQ-F (FX5U)
- Q
- L
- **New in 8.1.31** F (FX3)

Connecting to a Mitsubishi Device



INDUCTIVE
UNIVERSITY

**Connecting To Mitsubishi
Devices**

[Watch the video](#)

Before you can connect your Mitsubishi device to this driver, your GX Works software settings must meet certain requirements. Configuring GX Works will also provide information needed during the driver connection process, such as Hostname and Port.

To connect a Mitsubishi Device:

1. Navigate to **Config > OPC UA > Device Connections** on the Gateway Webpage.
2. On the Devices page, click on **Create new Device**.
3. Select **Mitsubishi TCP**, and click **Next**.
4. Fill in the Name field. Entering a description is optional.
5. Select the MELSEC series you will be connecting to using the dropdown. Options include iQR, iQF, Q, and L.

CAUTION

Be sure to select the correct device type when connecting to avoid causing an error condition on the device.

6. Enter the device hostname in the **Hostname** field.

7. Enter the device port in the **Port** field.

Config > Opcua > Devices

General	
Name	MitsubishiDriver
Description	
Enabled	☒ (default: true)

Main	
Series	iQR (default: iQR)
Hostname	
Port	1030 (default: 1,025)
Local Address	 Address of network adapter to connect from. (default:)
Timeout	5000 The request timeout, specified in milliseconds. (default: 5,000)

8. Enter the local address of the network adapter to connect from. This is necessary if you have multiple network cards and you need to specify the particular network card that has PLC access.

9. Leave the default values in the remaining fields.

10. Click **Create New Device**.

The Devices page now displays the Mitsubishi TCP device is successfully connected.

Device Settings

General

Property	Description
Name	Device name.
Description	Device description.
Enabled	If true, attempt connection to the device, default is true.

Main

Property	Description
Series	The series of the Mitsubishi device. Default is iQR.
Hostname	The IP Address of the device.
Port	The port the device is listening on.
Local Address	Address of network adapter to connect from.
Timeout	The request timeout, specified in milliseconds with a valid range from 0 to 16383750. Default is 5000.

Request Optimization

Property	Description
Max Gap Size	The maximum address “gap” to span when optimizing requests to contiguous address locations. Increasing the max gap size will reduce the number requests but increase the amount of data requested. Default is 0.
Write Priority Ratio	The number of write requests to execute for every read request when an abundance of both types are queued up for execution. Default is 5.

Access Route Settings

New in 8.1.43

Property	Description
PC Number	The network module station number of an access target. Default is 255. Note: The station number that should be used as the PC number will vary depending on how your PLC network is set up. For instance, when accessing a station via network, the PC number is the station number of the access target. However, when accessing a multi-drop connection station via network, the PC number is the station number of the relay station.
Network Number	The network number of an access target. Default is 0.
Request Destination Module I/O Number	The start input/output number of a multi-drop connection source module, or the CPU module of a multiple CPU system. Default is 1023.
Request Destination Module Station Number	The station number of an access target module when accessing a multi-drop connection station. So, when the PC Number is set to the station number of the relay station, this property should be set to the target station number. Default is 0.

Adding Addresses

You can add addresses through a few different options on the Gateway or in the Designer.

Addressing on the Gateway

The Mitsubishi addresses page on the Gateway allows entering Browse Path and Address information individually, importing user-created CSV files containing a list of addresses, and importing a GX Works CSV file. Even if you are planning to import multiple addresses using a CSV file, it's recommended to first enter a single tag address to test if your connection is working as expected. The addresses page has three areas of information:

- Browse Path: This represents the browse path for the OPC UA client.
- Address: This is the device, offset, and additional specifications to find desired values.

- Description: This is an optional field to help in identifying displayed values.

To access the addresses page:

1. Navigate to the **Config > OPC UA > Device Connections**.
2. Click the **More** dropdown and choose **addresses** on your Mitsubishi device.

⚙️ Config > Opcua > Devices

Name	Type	Description	Enabled	Status	
MitsubishiDriver	Mitsubishi TCP		true	Connected	More ▾ addressesdelete edit

→ Create new Device...

Adding Individual Address Rows

To manually enter addresses:

1. Click **Add Row** to populate rows for address information.
2. Enter Browse Path field and Address field information. Enter Description information as desired.

NOTE

All components are case-insensitive.

3. Continue adding rows and address information until you've added all the addresses you want to be available as tags.
4. Click **Save Configuration** when you are finished.

Load Configuration File 

 Remove All 

Browse Path	Address	Description	
			Remove

Add Row

Save Configuration

<< < 1 > >>

Importing and Exporting

To import a CSV file:

1. Select **Load Configuration File** on the device address page.
2. Select **Choose File**.
3. Locate the CSV file you want to upload and select it to open.

⚠ CAUTION

Make sure a comma separates column information, and if your Description column includes commas, use quotes to prevent misinterpretations when uploading.

4. Select the **Append to current configuration** box if you already have addresses listed and you want to add on to them. Leave unchecked if you want to replace existing rows.

5. Click **Load**.

⚠ CAUTION

If an existing address shares a tag name with an address imported by the CSV file, it will be replaced with the imported address.

Device Connections / MitsubishiDriver

The screenshot shows the 'Device Connections / MitsubishiDriver' interface. At the top, there is a blue button labeled 'Load Configuration File' with a small icon. Below this, a modal dialog is open, titled 'Load configuration from file (*.csv)'. Inside the dialog, there is a 'Choose File' button and the text 'No file chosen'. Below that, there is a checkbox labeled 'Append to current configuration?'. At the bottom of the dialog is a blue button labeled 'Load'. In the background, there is a text input field labeled 'Address' and a blue button labeled 'Save Configuration'.

6. Click **Save Configuration**.

To export the address list from the Gateway page, simply use the Export Configuration  icon to download.

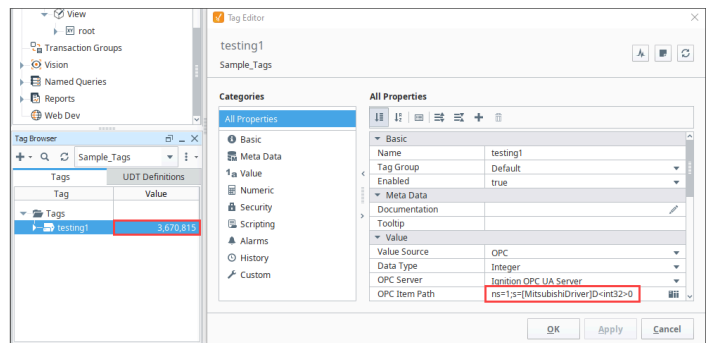
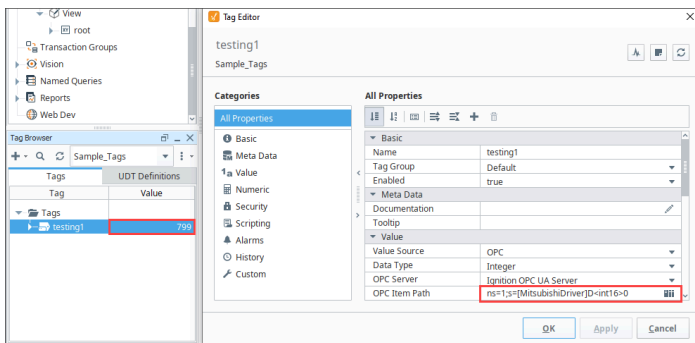
Addressing in the Designer

You also have the option to address tags via an Ignition OPC tag in the Ignition Designer. See [Ignition OPC UA Server Node Configuration](#) for more information. As opposed to adding addresses on the Gateway, the device needs to be specified here.

Example syntax, where optional components are included in curly brackets

```
ns=1;s=[DeviceName]Area{<DataType{[array]}>}Offset{.Bit}
```

Regardless of if the address path was entered on the Gateway and added as a tag, or entered directly in the Designer, you can continue to edit the item path in the Tag Editor if needed. The example below shows how changing the OPC Item Path data type from int16 to int32 instantly changes the tag value from 799 to 3,670,815.



Determining Addresses



INDUCTIVE
UNIVERSITY

Mitsubishi Addressing

[Watch the video](#)



INDUCTIVE
UNIVERSITY

Mitsubishi Address Mapping

[Watch the video](#)

The Mitsubishi driver requires Area and Offset address components, but also allows many optional specifications. It's important to know where the optional components must be placed in your address to properly connect. The example syntax below shows how optional data type (including modifiers)

and array specifiers are listed before the Offset specification, and the optional array element index and bit index must be listed after the Offset specification.

Example syntax, where optional components are included in curly brackets

```
Area{<DataType{[array]}>}Offset{.Bit}
```

The image below demonstrates how different address specifications correspond with a Mitsubishi data register. The Description column functions the same way as it does on the Mitsubishi Driver addresses page to detail where the address is looking for data. The following syntax sections on this page provide an in-depth breakdown on how to find and enter addresses similar to the ones shown in these examples.

		BIT POSITION																ADDRESSING	DESCRIPTION
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
OFFSET	0	0																D0	basic addressing by offsets
	1	10																D1	
	2	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	D2.0	bit addressing within an element
	3	30																D<int16>3	use of explicit datatypes
	4	40																D<int16>4	
	5	65,536																D<int32>5	double word size
	6																		
	7	131,072																D<int32>7	
	8																		
	9	100																D<int16[4]>9	1-D array allocation, size 4x1
	10	200																	
	11	300																D<int16[4]>9[2] or D<int16>11	single array element accessing
	12	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0	0	D<int16[4]>9[3].0 or D<int16>12.0	bit addressing within an array element
	13	1000																D<int16[3,3]>13	2-D array allocation, size 3x3
	14	2000																	
	15	3000																	
	16	4000																	
	17	5000																	
	18	6000																D<int16[3,3]>13[1][2] or D<int16>18	single array element accessing
	19	7000																	
	20	8000																	
	21	9000																	
	22	H																D<string6>22	string of length 5, null-terminated
	23	e																	
	24	l																	
	25	l																	
	26	o																	
	27	\0																	

Areas

 NOTE

On this page, and for the Mitsubishi driver, the term Area refers to the accessible Device Items. See the Area table below for potential device names and their related keywords and types.

MELSEC-Q/L, iQ-F, and iQ-R

The following table shows the MELSEC-Q/L, iQ-F and iQ-R series devices, keywords, and native types recognized by the driver. Extended Data Registers and Extended Link Registers are currently not supported.

Device Name	Keyword	Type
Special Relay	SM	Bit
Special Register	SD	Word
Input	X	Bit
Output	Y	Bit
Internal relay	M	Bit
Latch relay	L	Bit
Annunciator	F	Bit
Edge relay	V	Bit
Link relay	B	Bit
Step relay	S	Bit
Data register	D	Word
Link register	W	Word
Timer (Contact)	TS	Bit
Timer (Coil)	TC	Bit
Timer (Current value)	TN	Word
Long timer* (Contact)	LTS	Bit
Long timer* (Coil)	LTC	Bit
Long timer* (Current value)	LTN	Double word
Retentive timer (Contact)	STS	Bit

Device Name	Keyword	Type
Retentive timer (Coil)	STC	Bit
Retentive timer (Current value)	STN	Word
Long retentive timer* (Contact)	LSTS	Bit
Long retentive timer* (Coil)	LSTC	Bit
Long retentive timer* (Current value)	LSTN	Double word
Counter (Contact)	CS	Bit
Counter (Coil)	CC	Bit
Counter (Current value)	CN	Word
Long counter* (Contact)	LCS	Bit
Long counter* (Coil)	LCC	Bit
Long counter* (Current value)	LCN	Double word
Link special relay	SB	Bit
Link special register	SW	Word
Direct access input	DX	Bit
Direct access output	DY	Bit
Index register (Index register)	Z	Word
Index register* (Long index register)	LZ	Double word
File register (Block switching method)	R	Word

Device Name	Keyword	Type
File register (Serial number access method) Note: This register is not available for Q/L series with Ignition versions 8.1.43 and earlier.	ZR	Word
Refresh data register*	RD	Word

*Not available for Q/L series

MELSEC-F

New in 8.1.31

The following table shows the MELSEC-F series devices, keywords, and native types recognized by the driver.

Device Name	Keyword	Type
Input	X	Bit
Output	Y	Bit
Internal relay	M	Bit
Step relay	S	Bit
Data register	D	Word
Timer (Contact)	TS	Bit
Timer (Current value)	TN	Word
Counter (Contact)	CS	Bit
Counter (Current value)	CN	Offsets 0-199: Word Offsets 200-255: Double Word
File register (Block switching method)	R	Word

Offsets

A simple example of reading a bit would be D0.0, which reads the bit0 of int16 at offset 0. You can also read bits within array elements that represent scalar values following the same addressing format. To read the bit 0 of element 0 of an array of length 4 at offset 0 would be `D<int16[4]>0[0].0`.

Depending on the PLC series and bit area combination, offsets must be specified in octal, hex, or decimal. The following table lists the combinations that have a specification other than decimal.

	iQ-R	iQ-F	Q	L	F (FX3)
X	Hex	Octal	Hex	Hex	Octal
DX	Hex	N/A	Hex	Hex	N/A
Y	Hex	Octal	Hex	Hex	Octal
DY	Hex	N/A	Hex	Hex	N/A
B	Hex	Hex	Hex	Hex	N/A
SB	Hex	Hex	Hex	Hex	N/A
W	Hex	Hex	Hex	Hex	N/A
SW	Hex	Hex	Hex	Hex	N/A

NOTE

Note that leading zeroes and lowercase hex characters are accepted. For example XF would be read the same as X0f.

Optional Data Types

The Mitsubishi driver supports reading/writing data types larger than the area’s native type by combining bytes from adjacent offsets into a single value. For example, each Data Register (D) point is natively a word (16-bit), but reading D<int32>0 requests bytes for D0 and D1 and combining the bytes into a signed 32-bit value.

The Mitsubishi driver does not support reading/writing data types smaller than the area’s native type. For example, each Long Index Register (LZ) point is a double word (32-bit) which means LZ<int16>0

and LZ<uint16>0 cannot be specified.

Some areas in particular cannot be combined and specific data types corresponding to its native type must be specified:

- Any areas having a native type of Bit
- Any area having to do with Current value, such as:
 - TN
 - LTN
 - STN
 - LSTN
 - CN
 - LCN

NOTE

When specifying a string data type, the length of the string must also be specified on the address. For example, D<string10>0 requests a string of up to 10 characters at offset 0 in the data register device.

If the length of the string is less than the address string length, it will be padded with null characters. For example, writing “hello” to D<string10> is equivalent to writing “hello\0\0\0\0” where “\0” are null characters. If the length of the string is more than the address string length, it will be truncated to the address string length. For example, writing “hello world!” to D<string10> is equivalent to writing “hello worl”.

The string length will be automatically rounded up to the nearest even value. This is because two characters are stored inside a word. For example, D<string9>0 will be treated as D<string10>0.

The following data types are recognized by the driver:

- Bool
 - Int16
 - Int32
 - Int64
 - UInt16
 - UInt32
 - UInt64
 - Float
 - Double
-

- **Changed in 8.1.31** String (to use for ASCII characters)
- **New in 8.1.31** Wstring (to use for Unicode characters)

If the data type is omitted, a default data type will be used depending on the area's native type:

Area Native Type	Default Data Type
Bit	bool
Word	int16
Double word	int32

The following optional data type modifiers are recognized by the driver:

Attribute	Order
@BE	Big Endian Byte Order
@LE	Little Endian Byte Order (default when not specified)
@HL	High-Low Word Order (default when not specified)
@LH	Low-High Word Order

Optional Arrays

Arrays are a concept created within the driver representing contiguous blocks of memory starting from an offset. For example, instead of reading D<int16>0 through D<int16>8, you can create an array of the same values. The number of elements in an array are determined by multiplying the array dimensions. So reading D<int16[3, 3]>0 is a length of 3x3 where the start offset is 0 and the last offset is 8.

Similarly, reading an element within an array is equivalent to reading a single address. For example, you can see in the table below that reading D<int16[3, 3]>0[1][2] is equivalent to reading D<int16>5.

D[0][0] = D<int16>0	D[0][1] = D<int16>1	D[0][2] = D<int16>2
D[1][0] = D<int16>3	D[1][1] = D<int16>4	D[1][2] = D<int16>5
D[2][0] = D<int16>6	D[2][1] = D<int16>7	D[2][2] = D<int16>8

Large arrays should be used with caution. If an array read/write request is too large to fit into a single request, it will be broken up into multiple non-atomic sequential requests. Splitting requests when writing can result in the first send of the write requests creating new values and the remaining write requests failing. Similarly, if read requests are sent separately the resulting array may contain stale points.

Determining if an array is too large to be sent at the same time depends on how the array is optimized. Optimization refers to the process of minimizing the number of requests sent to the PLC by batching together device points with the goal to fit as many points as possible into a single request before splitting it off into multiple requests. Request optimization happens automatically when points are read or written to at the same time.

Optional Bit Index

Bit indexing can be specified on any integer data type to retrieve a boolean value at a bit position. Addresses with a bit index are read-only and any attempts to write to them will result in a Bad_NotSupported error due to the MELSEC protocol.

A bit index cannot be specified for the following:

- Non-integer data types (bool, float, double, and string)
- Non-scalar addresses (arrays)

GX Works Configuration

GX Works is the programming software designed for Mitsubishi PLCs. Before you can use the Mitsubishi driver, some configurations need to be made in GX Works when connecting to MELSEC-Q/L, iQ-F and iQ-R series devices with the Mitsubishi driver. These instructions are written for GX Works3. GX Works2 has slightly different settings but should in general resemble the following setting requirements.

1. Under the navigation tab, locate **Parameter > CPU > Module Parameter > Basic Settings > Own Node Settings**.
2. Set **Enable/Disable Online Change** to Enable All (SLMP).

3. Set **Communication Data Code** to Binary.
4. Navigate to External Device Configuration > Detailed Setting to add an SLMP Connection Module to the Ethernet Configuration with Protocol set to TCP.
5. Make sure to also identify the IP Address and Port No. These settings are what will be used for the Mitsubishi TCP driver Hostname and Port connection fields.
6. Write all changes to the PLC (Online > Write to PLC) and initiate a power cycle.

Configuring Device Points

Some areas or offsets within an area may not be accessible by default unless additional memory is allocated.

1. Navigate to **Parameter > CPU > CPU Parameter > Memory Device Setting > Device/Label Memory Area Setting > Detailed Setting**.
2. Change Points to desired size. The point allocation from all areas must not exceed the Total Device size.
3. Click Apply.
4. Write to the PLC (Online > Write to PLC) and initiate a power cycle.

Using GX Works3 to Troubleshoot

If you have access to GX Works3, you can use the Device/Buffer Memory Batch Monitor feature to test or troubleshoot your connection. This can be found under Online > Monitor > Device/Buffer Memory Batch Monitor. Type in the device keyword with an offset and hit enter to begin monitoring values.

GX Works also has an Ethernet Diagnostics tool which is useful for determining the status of the connection as well as other information like IP Address, Port No, and Latest Error Code. To access this tool, navigate to Diagnostics > Ethernet Diagnostics.

FX Configurator

FX Configurator is the required software for initial PLC setup when connecting to MELSEC-F series devices. Once you have created an FX3U model GXDeveloper project, complete the following configuration to begin making device connections.

1. Click Tools from the menu bar to select FX Special Function Utility > FX Configurator-EN.
2. Enter the minimum required FX3U-ENET configuration information.

3. Open the Operational Settings under Ethernet Module Settings and set the settings to the following:
 - Communication Data Code: Binary code.
 - Initial Timing: Always wait for OPEN.
 - Input Format: DEC.
 - Send Frame Setting: Ethernet(V2.0).
 - TCP Existence Confirmation Setting: Use the KeepAlive.
4. Select End when finished.

You can now add connections by selecting Open Settings back in the FX Configurator-EN window. These connections must set the Protocol to **TCP** and Fixed Buffer Communication Procedure to **Procedure Exist(MC)**. Recommendations for the remaining settings are as follows:

- Open System: Unpassive
- Fixed Buffer: Send
- Pairing Open: Disable
- Existence Confirmation: Confirm