

Ignition 8.3 User Manual — OPC UA and Device Drivers

Generated 2026-10-12 from
<https://www.docs.inductiveautomation.com/docs/8.3/>

OPC UA

OPC Specification

Open Platform Communications (OPC) is a standard for exchanging information between hardware and software through specialized drivers. It's used to connect devices like PLCs, sensors, gauges, and databases. OPC is maintained by the OPC Foundation, a global organization focused on ensuring interoperability in industrial systems.

The original OPC specifications were based on Microsoft DCOM and included separate specs for Data Access (OPC-DA), Alarms and Events (A&E), and Historical Data Access (HDA). These were later consolidated into a platform-independent specification, OPC Unified Architecture (OPC UA), which uses modern networking standards like TCP/IP.

OPC UA in Ignition

OPC UA is the primary data access standard used in Ignition. With the OPC UA module installed, Ignition can function as both an OPC UA server and client. This lets you connect to PLCs and other devices, and share data with other Ignition modules or external OPC UA clients.

Ignition also includes a built-in OPC UA client by default, even without the module. This allows the Gateway to connect to third party OPC UA servers that follow the specification. As the latest revision of the OPC standard, OPC UA provides a platform and vendor neutral way to transfer industrial data. It plays a central role in the Gateway and supports communication across a wide range of systems.

Ignition supports any number of connections to OPC UA servers from compliant manufacturers. Once connected, this data can be used throughout the system to drive visualization, alarms, logic, and more.

Device and server connections are configured on the Gateway under **Connections**.

Distributed Systems with OPC UA

Because OPC UA is built on TCP/IP and supports encryption, it works across networks and through firewalls. This makes it ideal for building distributed systems where remote sites collect data and report back to a central location. For instance, a remote Gateway can gather PLC data and securely forward it to a central Gateway for storage and visualization.

In a typical scenario, a central office might be connected to multiple remote facilities through a VPN. Each remote site could run a lightweight Ignition installation with just the OPC UA module, collecting local device data and sending it upstream. The central Gateway would then store the incoming data in a database, making it accessible for visualization through the Perspective or Vision modules or any external application that can query the database. This architecture keeps system costs low while enabling centralized data management across the organization.

OPC Servers and Clients

In OPC terminology, there are two primary roles:

- An **OPC server** provides access to data, such as realtime values from PLCs or sensors.
- An **OPC client** connects to a server to read or write data, monitor changes, or interact with exposed nodes.

Ignition as an OPC Client

Ignition inherently includes OPC UA client functionality, even without the OPC UA module installed. It can connect to external OPC UA servers, such as:

- Third party device servers (e.g., Kepware, Unified Automation Demo Server)
- Another Ignition Gateway's OPC UA server
- Software gateways translating protocols to OPC UA

Once connected, you can browse the server's address space and create OPC tags in the Designer. These tags subscribe to live data and can be used throughout Ignition for visualization, scripting, history, alarms, and more.

Ignition as an OPC Server

When the OPC UA module is installed, Ignition exposes its own built-in OPC UA server. This server provides access to data from:

- Connected devices via installed drivers (e.g., Allen-Bradley, Modbus)
- Memory tags
- Expression tags
- Historical tags

External OPC UA clients can subscribe to tag data in Ignition. For example:

- SCADA systems can pull live data from Ignition
- Other Gateways can consume tag values over the network
- Tools like UaExpert can monitor and test Ignition's tag structure

You can view and configure connections to external clients under **Connections > OPC > Connections** on the Gateway.

Bridging with OPC-DA (COM)

To work with devices or software that use OPC-DA (the DCOM-based standard), install the **OPC COM module**. This allows Ignition to act as a bridge between OPC-DA and OPC UA.

A common use case is connecting to Windows-based OPC servers and exposing their data in Ignition.

CAUTION

DCOM-based communication is difficult to configure and insecure over networks. It is recommended to use OPC UA whenever possible.

OPC UA 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

To read data from a PLC or industrial device, you will first need to [set up a device connection](#) using one of the supported OPC UA drivers. Each driver is designed for a specific protocol or device family, and once configured, the data becomes available to tags, bindings, and scripts throughout your project. See the device types below for more information on your specific device:

- **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**

Connecting to a Server

If you want Ignition to act as an OPC UA client and pull data from another server, you can configure a connection to that server directly on the Gateway. This is useful for integrating third party systems or connecting to another Ignition Gateway running as an OPC UA server.

See the [OPC UA Connections](#) page for more information.

Connecting to an OPC COM Server

To connect to an older OPC DA server that uses DCOM, you will need the OPC COM module. This module supports these legacy systems by bridging the gap between OPC DA and the OPC UA stack.

See the [OPC COM](#) page for details.

OPC Quick Client

Once you have a device connected, you can use the **OPC Quick Client** to browse the OPC UA server and verify live values directly from the Gateway at **Connections > OPC > Quick Client**. This provides a simple way to validate device connections and inspect data without creating tags or opening the Designer.

The page displays a tree view of the Ignition OPC UA Server, including connected devices and their exposed nodes. Expand the tree to browse available objects and tags. Nodes display action indicators that reflect their capabilities. **[R]** indicates read access, **[W]** indicates write access, and **[S]** indicates that the node can be subscribed to. Clicking **[S]** creates a subscription for the selected node and adds it to the Monitored Items table at the bottom of the page, where live values, quality, and timestamps are displayed.

You can select one or more monitored items and click **Set Rate** to adjust the polling rate in milliseconds. Use **Clear Monitored Items** to remove all active subscriptions, or the three dots menu on a row to remove individual items.

NOTE

The OPC Quick Client is intended for quick testing and diagnostics and does not create or persist tags.

Quick Client

NAME	TYPE	
▼ Ignition OPC UA Server	Server	Refresh
▼ Devices	Object	
▼ [Sample_Device]	Object	
▼ [Controls]	Object	
▶ Base Rate	Tag	[S] [R] [W]
▶ Pause	Tag	[S] [R] [W]
▶ Program Counter	Tag	[S] [R] [W]
▶ Repeat	Tag	[S] [R] [W]
▶ Reset	Tag	[S] [R] [W]
▶ Ramp	Object	
▶ Random	Object	
▶ ReadOnly	Object	
▶ Realistic	Object	
▶ Sine	Object	
▶ Writable	Object	

Monitored Items

View

20

Set Rate

Clear Monitored Items

☐	SERVER	ADDRESS	VALUE	QUALITY	TIMESTAMP	
☐	Ignition OPC UA Server	ns=1;s=[Sample_Device] [Controls]/baseRateNode	1000	GOOD	24Dec2025 09:25:44	

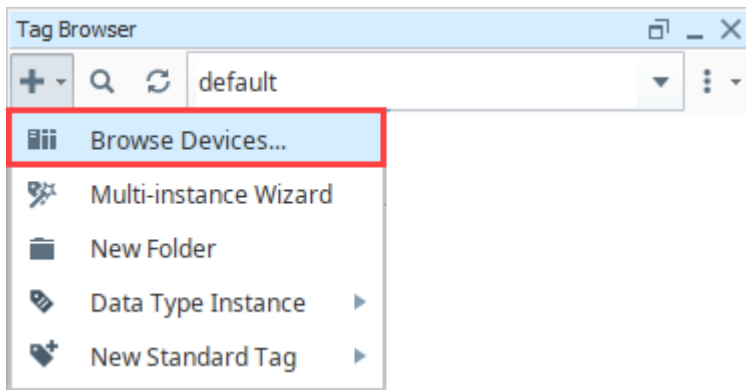
Viewing OPC UA Server Details

Expanding the three dots menu for listed connections and selecting **View Details** will open the Connections Details panel. From here, you can view the **diagnostic information** for the connected OPC UA server, split between a Server and Client tab. The Server Diagnostics tab shows server diagnostics, while the Client Diagnostics tab shows client connection subscription diagnostics. Note that the View Details option will not be available when a connection is Faulted.

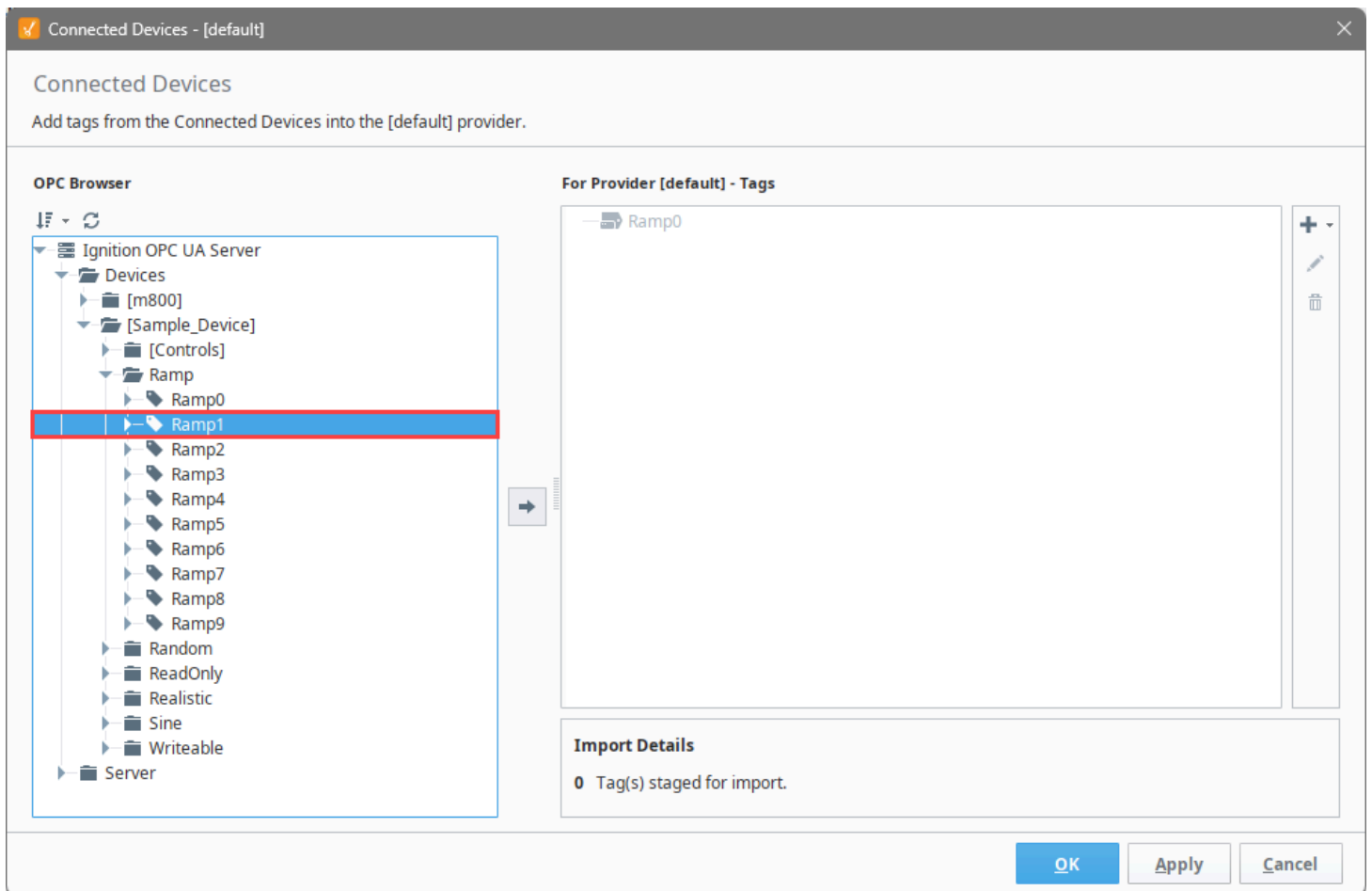
Get PLC Data

Before you can bring PLC data into Ignition, you must have a **connection created** with a device that supports subscriptions. Some drivers, like Modbus and Siemens, may require you to manually configure tags instead of browsing. This means you will not see live values in the OPC Browser and will need to manually create each tag with the correct OPC Item Path based on the device's address syntax. Once you have your device connection configured, you can bring PLC data into Ignition using the Designer.

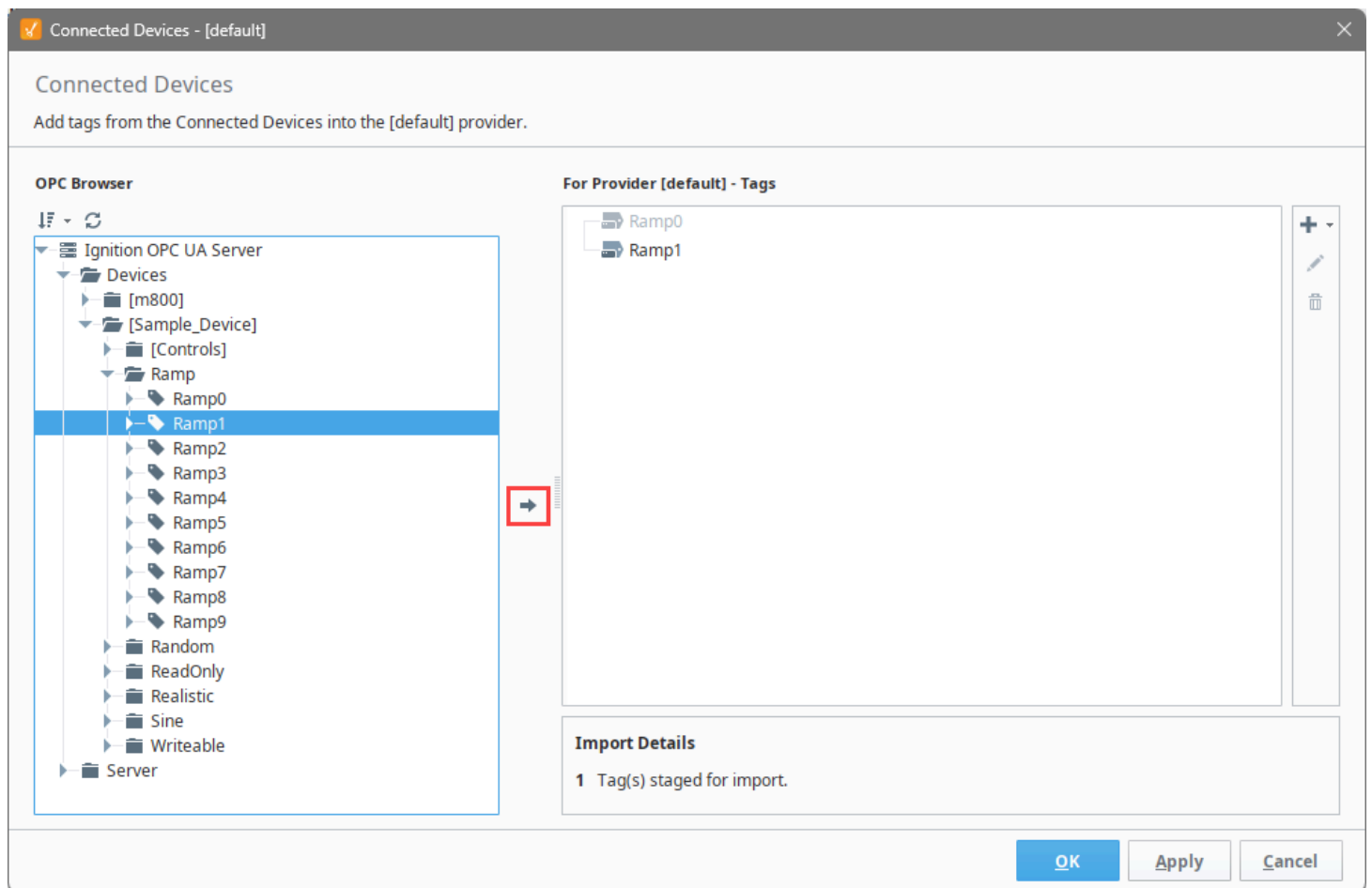
1. Launch the Designer and open your project.
2. From the Tag Browser, click the plus icon and select Browse Devices.



3. On the Connected Devices popup, navigate the OPC Browser on the left to find a live data point from your device.



4. Select any number of tags and click the arrow to stage them for import.



5. Click OK to update the Tag Browser with the imported tags.

The tag appears in your Tag Browser and begins updating in real time. This means Ignition has subscribed to the data point and is now receiving updates from the device on a regular interval, based on the tag group configuration.

Tag	Value
Ramp0	2.52
Ramp1	74.37

If a device doesn't support browsing (like [Modbus](#) or [Siemens](#)), create a tag manually and configure the **OPC Item Path**. See our driver documentation for syntax.

OPC UA Connections

OPC Connection Status

The OPC Connections page displays all currently configured OPC (both UA and DA) connections, as well as their status. If any are faulted, you can click the red faulted status to get an error message popup with a full description of the error.

Property	Description
Name	OPC server name.
Type	OPC server type - UA or DA
Description	A short description of the OPC UA connection.
Uptime	Total time OPC server is connected.
Enabled	True or false value whether the device connection is enabled.
Status	Current status of OPC server.

OPC Connections

Status

[Open Metrics Dashboard](#) 

CONNECTED SERVERS

2 / 2

ERRORED CONNECTIONS

0

SERVERS ENABLED

2 / 2

Connections

View [Create OPC Connection](#) 

☐	NAME	TYPE	DESCRIPTION	UPTIME	ENABLED	STATUS	
☐	Ignition OPC UA Server	OPC UA Connection	Default "loopback" connection to Ignition's OPC UA server	1 hour, 31 minutes, 47 seconds	true	 CONNECTED	
☐	IgnitionUM-OPC UA Server	OPC UA Connection		1 hour, 14 minutes, 40 seconds	true	 CONNECTED	

Showing 2 of 2 items

Diagnostic Information

Diagnostic information for each OPC connection is displayed on the Connections Details panel, separated by Server and Client tabs. The Server Diagnostics tab displays server status, build info, and server diagnostics. The Client Diagnostics tab displays subscription information for that particular server. It will list out all of the subscriptions to that server with the publishing rates, as well as the number of items within that subscription.

To access the Connections Details panel:

1. Go to **Connections > OPC > Connections**.
2. Click the **three dots menu** next to a connection and select **View Details**.

Now you can switch between the Server Diagnostics tab and the Client Diagnostics tab to monitor information as desired.

Server

By default, server diagnostics are disabled since they can generate a considerable amount of network overhead and impact performance. To begin tracking server diagnostics, click **Enable Server Diagnostics** on the Server Diagnostics tab. Note that not all OPC UA servers support server diagnostics.

Server Status

PARAMETER	VALUE
currentTime	2023-08-09T18:26:10.574Z
startTime	2023-08-09T18:26:10.574Z
status	✓ CONNECTED

Build Info

PARAMETER	VALUE
buildDate	2023-08-09T18:26:10.574Z
buildNumber	dev
manufactureName	Inductive Automation
productUri	urn:inductiveautomation:ignition:opcua:server
softwareVersion	dev

Server Diagnostics **DISABLED**[Enable Server Diagnostics](#)

PARAMETER	VALUE
This server does not have diagnostics enabled.	

Changed in 8.3.9 Due to a Milo 1.1.6 upgrade, the OPC UA server library now enforces users to hold a `ConfigureAdmin` or `SecurityAdmin` role when enabling server diagnostics. New installations for versions 8.3.9+ will grant a `ConfigureAdmin` role to the default opcuauser automatically. If you have upgraded from 8.3.8 and earlier to 8.3.9 or later, this role will need to be manually added to the opcua-module User Source. See the [Granting Roles](#) section for more information on how to add `ConfigureAdmin` and `SecurityAdmin` roles.

X

Ignition OPC UA Server Details

Server Diagnostics

Client Diagnostics

Server Status

PARAMETER	VALUE
startTime	2025-06-11T06:46:15.939-07:00
currentTime	2025-06-12T10:35:01.882-07:00
state	CONNECTED

Build Info

PARAMETER	VALUE
productUri	urn:inductiveautomation:ignition:opcua:server
manufacturerName	Inductive Automation
softwareVersion	dev
buildNumber	dev
buildDate	2025-06-11T06:46:15.204-07:00

Server Diagnostics

DISABLED

Enable Server Diagnostics

PARAMETER	VALUE
serverViewCount	null
currentSessionCount	null
cumulatedSessionCount	null
securityRejectedSessionCount	null
rejectedSessionCount	null
sessionTimeoutCount	null
sessionAbortCount	null
currentSubscriptionCount	null
cumulatedSubscriptionCount	null
publishingIntervalCount	null
securityRejectedRequestsCount	null
rejectedRequestsCount	null

A **View Session Diagnostics** option will appear when server diagnostics are enabled. Selecting this will allow you to inspect session and subscription activity.

X

IgnitionUM-OPC UA Server Details

Server Diagnostics

Client Diagnostics

Server Status

PARAMETER	VALUE
startTime	2025-06-12T11:33:20.703-07:00
currentTime	2025-06-12T13:54:35.813-07:00
state	CONNECTED

Build Info

PARAMETER	VALUE
productUri	urn:UnifiedAutomation:UaServerCpp
manufacturerName	Unified Automation GmbH
softwareVersion	1.8.7
buildNumber	644
buildDate	2025-03-20T07:16:37.000-07:00

Server Diagnostics

ENABLED

View Session Diagnostics

Disable Server Diagnostics

PARAMETER	VALUE
serverViewCount	0
currentSessionCount	1
cumulatedSessionCount	5
securityRejectedSessionCount	0
rejectedSessionCount	0
sessionTimeoutCount	0

Granting `ConfigureAdmin` or `SecurityAdmin` Roles

When updating from 8.3.8 and earlier, you will need to manually add either a `ConfigureAdmin` or `SecurityAdmin` role to be able to use the Enable Server Diagnostics option for OPC connections. Only one of these roles is required, but both can be assigned. **This process is not needed with new installs of versions 8.3.9 and later.** Once the role has been assigned, the OPC connection will need to be reconnected to apply the permission changes.

The following process describes how to add a `ConfigureAdmin` role to the `opcuauser` within the `opcua-module` User Source, and then start tracking server diagnostics. The same steps can be followed to add the `SecurityAdmin` role.

1. Navigate to **Platform > Security > User Sources**.
2. Expand the three dots menu for the `opcua-module` User Source and select **Manage Users**.
3. Click the **Roles** tab.
4. Click **Create Role +**.
5. Enter **ConfigureAdmin** in the Role Name field.
6. Click **Create Role**.
7. Return to the **User** tab.
8. Expand the three dots menu for `opcuauser` and select **Assign Roles**.

- 9. Check the box for **ConfigureAdmin**.
- 10. Click **Save Changes**.
- 11. Navigate to **Connections > OPC > Connections**.
- 12. Expand the three dots menus to **Disable** and then **Enable** OPC connections so that the new role will take effect.
- 13. Select **View Details** from the connection's three dots menu once the status returns to Connected.
- 14. Click **Enable Server Diagnostics**.

Client

The Client Diagnostic tab subscription list can be filtered by search criteria, viewed based on a specified number of subscriptions per page, and refreshed as needed to see the latest subscription data. The table below describes the information displayed for each subscription.

Property	Description
Name	Displays the subscription name.
Rate	The rate defined in the Tag Group, used as the requested sampling interval for monitored Items belonging to the corresponding subscription.
Request Publishing Interval	The rate a subscription will report accumulated change notifications at. The interval is derived from Rate, unless explicitly defined.
Revised Publishing Interval	The server's revised rate for accumulated change notifications.
Tag Count	The number of tags currently subscribed to that Tag Group.

Nodes

To monitor subscriptions and diagnostics from the Client Diagnostics tab, expand a group to see the monitored node details. Clicking the dropdown arrow for any of the subscriptions will open the list of subscribed OPC items.

Property	Description
Node ID	The OPC item path.
Requested Sampling Interval	The rate the underlying tag/node will be polled at.
Revised Sampling Interval	The revised rate at which the underlying tag/node will be polled at.
Requested Queue Size	Determines how many data points can be stored and transferred to the client once the sampling rate elapses when the sampling rate is slower than the publishing rate.
Revised Queue Size	The revised amount of data points that can be stored and transferred to the client once the sampling rate elapses when the sampling rate is slower than the publishing rate.
Status Code	Status of the node.

✕

IgnitionUM-OPC UA Server Details

Server Diagnostics

Client Diagnostics

ⓘ

Client Diagnostics and Nodes

✕

A list of the subscriptions to this server with the publishing rates, as well as the number of items within that subscription. Expand each subscription's row below to view a secondary list of subscribed OPC items (or, nodes).

Client Diagnostics

🔍 Filter client diagnostics...

View 20 ▾

Refresh ↻

NAME	RATE	REQUESTED PUBLISHING INT	REVISED PUBLISHING INT	TAG COUNT	⚙
tag-group-ignitionum-tag s_default	1000	500	500	1	
NODE ID	REQUEST SAMPLING INT	REVISED SAMPLING INT	REQUESTED QUEUE SIZE	REVISED QUEUE SIZE	STATUS CODE
ns=3;s=Demo.Static.Scal...	1000	1000	1	1	Good

Showing 1 of 1 items

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 Connections section of the Gateway Webpage, go to **OPC > Connections**. The OPC Connections page is displayed.
- 2. Click on **Create OPC Connection +**. A popup wizard will appear.

Connections

View 20

Create OPC Connection +

☐	NAME ↑	TYPE	DESCRIPTION	UPTIME	ENABLED	STATUS	
☐	Ignition OPC UA Server	OPC UA Connection	Default "loopback" connection to Ignition's OPC UA server	3 hours, 14 minutes, 33 seconds	true	CONNECTED	

Showing 1 of 1 items

- 3. Select **OPC UA Connection** 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

Using the **Skip to Advanced Configuration** link on the bottom right of the popup will allow you to manually configure connection settings. This is useful in cases where a server does not allow anonymous endpoint access, but provides separate discovery endpoints.

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

Create OPC UA Connection

Create OPC UA Connection

1. Connection Type

2. Server Discovery

3. Server Selection

4. Endpoint Selection

5. Confirm Settings

6. Configure Connection

Ignition OPC UA Server

☒ `opc.tcp://localhost:62541/discovery`

Back

Skip to Advanced Configuration

Next

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

Create OPC UA Connection

Create OPC UA Connection

1. Connection Type

2. Server Discovery

3. Server Selection

4. Endpoint Selection

5. Confirm Settings

6. Configure Connection

☒ `opc.tcp://localhost:62541`

Security Policy: Basic256Sha256

Security Mode: SignAndEncrypt

Back

Skip to Advanced Configuration

Next

8. A summary of your selections thus far will be displayed. Once you confirm your settings, click **Next**.

Create OPC UA Connection

Create OPC UA Connection

1. Connection Type

2. Server Discovery

3. Server Selection

4. Endpoint Selection

5. Confirm Settings

6. Configure Connection

Discovery URL: opc.tcp://localhost:62541/discovery

Endpoint URL: opc.tcp://localhost:62541

Endpoint Host Override:

Endpoint Security Mode: SignAndEncrypt

Endpoint Security Policy: Basic256Sha256

Back

Skip to Advanced Configuration

Next

9. A Manage Certificate window will open if you haven't previously trusted a certificate. Trust the certificate and click **Next**. If the certificate that the Endpoint sends you to is already in Ignition's trust store, this step is skipped.
10. If you entered a discovery URL in step 4, you also have an option to enter another URL if the host is unreachable.
11. 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.
12. On the **Configure Connection** page, give the connection a name. Some OPC UA servers may also require specific Authentication credentials. 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.

Create OPC UA Connection

Create OPC UA Connection

1. Connection Type

2. Server Discovery

3. Server Selection

4. Endpoint Selection

5. Confirm Settings

6. Configure Connection

GENERAL

Name *

My OPC UA Connection

Description

Enabled

☒ Set whether this connection is enabled or disabled within the Gateway.

Read Only

☐ If selected, the connection to this OPC server will be read-only; all calls to write will fail.

AUTHENTICATION

Authentication Type

Username

Create OPC Connection

13. Once all desired settings have been configured, click **Create OPC Connection**.

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
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
Authentication Type	Changed in 8.3.3 The Authentication Type field has replaced the Username/Password fields to additionally provide Anonymous and Certificate selections. Once an option has been chosen, relevant sub properties based on the Authentication Type will display. • Username: Requires a user and password to authenticate the connection. This is the default type. Username: A username the connection will use when authenticating with the UA server. Password: The password to use when authenticating with the UA server. You can select between None, Embedded, or Referenced. • Anonymous: Allows authentication for this OPC UA connection anonymously. • Certificate: Enables X.509 certificate-based authentication for this OPC UA connection. Certificate Pem: The PEM encoded X.509 certificate that this OPC UA client will use for Session authentication. Private Key Pem: The PEM encoded private key in PKCS1/PKCS8 format. You can select between Embedded or Referenced.

Keep Alive

Property	Description
Keep-Alive Failures Allowed	The allowed number of consecutive failures before disconnecting. A zero or negative value will not cause a disconnect. Default is 1.
Keep-Alive Interval	The interval between keep-alive requests, in milliseconds. Default is 15,000.
Keep-Alive Timeout	The maximum duration to wait for a response to a keep-alive request, in milliseconds. Default is 10,000

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.
Deprecated Data Type Dictionary Support	When enabled, the deprecated DataType Dictionary mechanism will be used to decide custom DataTypes, instead of the DataTypeDefinition attribute.
Browse 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.
Timestamp Source	New in 8.3.1 The source to use for timestamps in values from this connection. Options include: • IGNITION • OPC_PREFER_SERVER • OPC_PREFER_SOURCE

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. 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	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	The password to use when authenticating with the UA server. You can select between None, Embedded, or Referenced.

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 Discovery URL**, **Failover Endpoint URL**, and **Failover Host Override**. 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 **Connections** section of the Gateway Webpage. On the sidebar, locate **OPC > 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

The tables below detail the General Settings tab on the Ignition's OPC UA Server Settings page. 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	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.	false
Max Session Count	The maximum number of client connections to the UA server.	100
GDS Push Enabled	New in 8.3.5 When enabled, the server's certificate and trust list can be managed by an external GDS via the GDS push model.	false

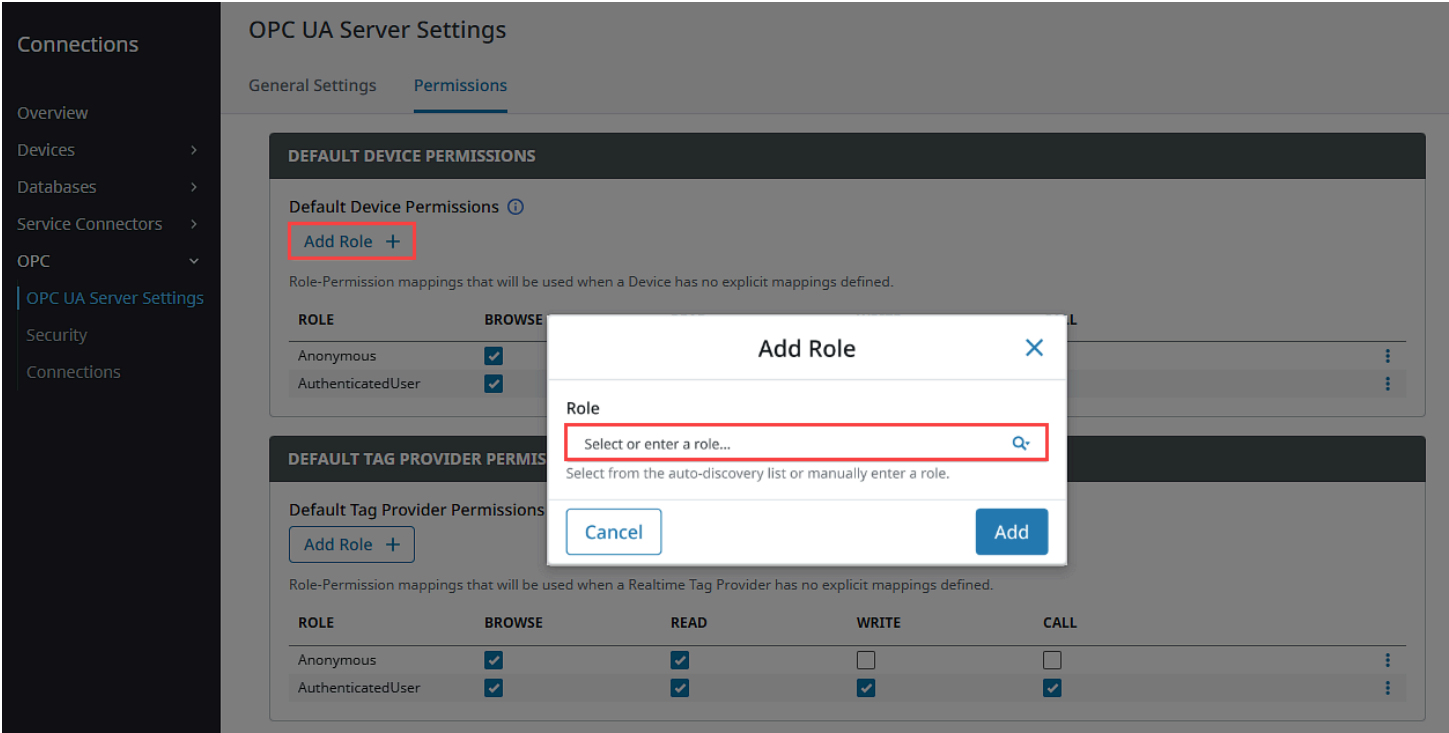
Redundancy

Setting	Description	Default Value
Read-only When Inactive Node	When enabled, this server switches to a read-only state while its Gateway is the inactive node in a redundant pair .	false

OPC UA Server Permissions

The Permissions tab allows users to set role permissions for access to devices and exposed tags. Each role will have the options for Browse, Read, Write, and Call.

Click the **Add Role** option to select or enter a role. Roles already created for the `opcua-module` user source will be available to select from the dropdown. You can also enter a new role here to assign permissions, but then you will need to navigate to the **Platform > Security > User Sources** Gateway page to create a role of same name for the permissions to be applied to the desired `opcua-module` users.



After the role is added to the Permissions list, check the access levels you want to apply. Make sure to click **Save Changes** at the top of the page after any permissions update. You can remove roles by expanding the three dots menu and selecting **Remove**.

Default Device Permissions

Setting	Description
Default Device Permissions	Role-Permission mappings that will be used when a Device has no explicit mappings defined. Role Names must be unique.

Default Tag Provider Permissions

Setting	Description
Default Tag Provider Permissions	Role-Permission mappings that will be used when a Realtime Tag Provider has no explicit mappings defined.

Tag Provider Permissions

Setting	Description
Individual Tag Provider Permissions	Role-Permission mappings, specific to a selected Tag Provider, that will override any Default Tag Provider Permissions defined in the Default Tag Provider Permissions section above. After a Tag Provider is added, you can add permissions for it.

Verifying Redundancy State with the OPC Quick Client

The OPC Quick Client can be used to verify which Gateway is currently active in a **redundant Gateway pair** by reading the OPC UA server's runtime status.

1. Open the Gateway and navigate to **Connections > OPC > Quick Client**
2. Expand **Ignition OPC UA Server**, then expand **Server**.
3. Locate **ServiceLevel**, then click the **[R]** (Read) icon to display the results at the top of the page.

OPC Quick Client

✓ Read Complete

Server: [Ignition OPC UA Server]i=2267

Value: 255

Quality: Good

Timestamp: 29Dec2025 11:32:20

Quick Client

NAME	TYPE	
Ignition OPC UA Server	Server	Refresh
Devices	Object	
Server	Object	
ServiceLevel	Tag	[S] [R] [W]
UrisVersion	Tag	[S] [R] [W]

The **ServiceLevel** value indicates which Gateway is currently active. This value is read-only and reflects the current redundancy state.

Service Level	Description
255	The OPC UA server is running on the master Gateway and is currently active.
254	The OPC UA server is running on the backup Gateway and is currently active.
1	The OPC UA server is inactive.

 NOTE

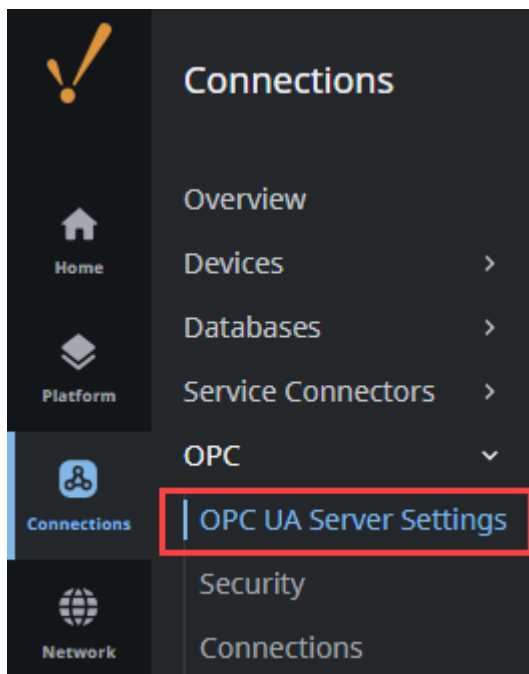
In some network conditions, such as a temporary Gateway Network communication loss, both nodes may briefly report active values. This typically resolves once communication between the nodes is restored.

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 **Connections > 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 can also be appended.

ENDPOINT CONFIGURATION *Required

Bind Port *

62541

The port the UA server will bind to.

Bind Addresses *

[Add Address +](#)

A list of the addresses the server will bind to.

ADDRESS
localhost

Endpoint Addresses *

[Add Address +](#)

A comma separated list of endpoint addresses that the UA server can be reached at.

ADDRESS
<code><hostname></code>
<code><localhost></code>

3. Click **Save Changes** at the top of the page. After making changes to your settings, saving may help flush out any residual information.
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. Click **Save Changes**.

ENDPOINT CONFIGURATION

*Required

Bind Port *

62541

The port the UA server will bind to.

Bind Addresses *

Add Address +

A list of the addresses the server will bind to.

ADDRESS
localhost

5. Check your Security Policies. Possible values for this setting are listed in the table under [OPC UA Server Settings](#). Click **Save Changes**.

Security Policies *

☐ None

☒ Basic256Sha256

☐ Aes128_Sha256_RsaOaep

☐ Aes256_Sha256_RsaPss

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.

OPC Connections

Status

CONNECTED SERVERS

1 / 1

ERRORED CONNECTIONS

0

SERVERS ENABLED

1 / 1

[Open Metrics Dashboard](#)

Connections

View 20 [Create OPC Connection +](#)

☐	NAME	TYPE	UPTIME	ENABLED	STATUS	
☐	Ignition OPC UA Server	OPC UA Connection	7 days, 3 hours, 18 minutes	true	CONNECTED	

Showing 1 of 1 items

OPC UA

OPC UA is the Unified Architecture for connecting to OPC devices that can be used on any OS. It is not tied to Microsoft like its predecessor. 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. This means that 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 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 has lost popularity 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. Since there are many still in use today, Ignition has an **OPC COM module** to connect to them.

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 **Connections** section of the Gateway, go to **OPC > Connections**. The OPC Connections page is displayed showing the OPC UA servers your Ignition is connected to.
2. Click **Create OPC Connection**.
3. Choose **OPC UA Connection** 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 the **Next** to continue.

Create OPC UA Connection

Create OPC UA Connection

1. Connection Type

2. Server Discovery

3. Server Selection

4. Endpoint Selection

5. Confirm Settings

6. Configure Connection

Endpoint URL

opc.tcp://localhost:49320

Example(s): opc.tcp://localhost:62541/discovery

Back

Skip to Advanced Configuration

Next

5. Select the Server you want and click **Next**.
6. A list of available Endpoints with Security Policies and Security Modes options appears. Select an endpoint configuration and click **Next**.
7. On the Manage Certificate page select **Yes** for **Trust Certificate?** and click **Next**.

USING REDUNDANCY

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 **Connections > OPC > Security** page under the **Quarantined Certificates** section.

8. Confirm your settings and click **Next**.
9. This takes you to the new OPC Connection form. 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.

Create OPC UA Server Connection

Create OPC UA Server Connection

1. Connection Type

2. Server Discovery

3. Server Selection

4. Alternate Discovery URL

5. Select Endpoint

6. Override Endpoint Hostname

7. Manage Certificate

8. Confirm Settings

6. Configure Connection

GENERAL

*Required

Name *

KEPServerEX/UA

The name of this connection.

Description

Enabled

☒ Set whether this connection is enabled or disabled within the Gateway.

Read Only

☐ If selected, the connection to this OPC server will be read-only; all calls to write will fail.

AUTHENTICATION

Username

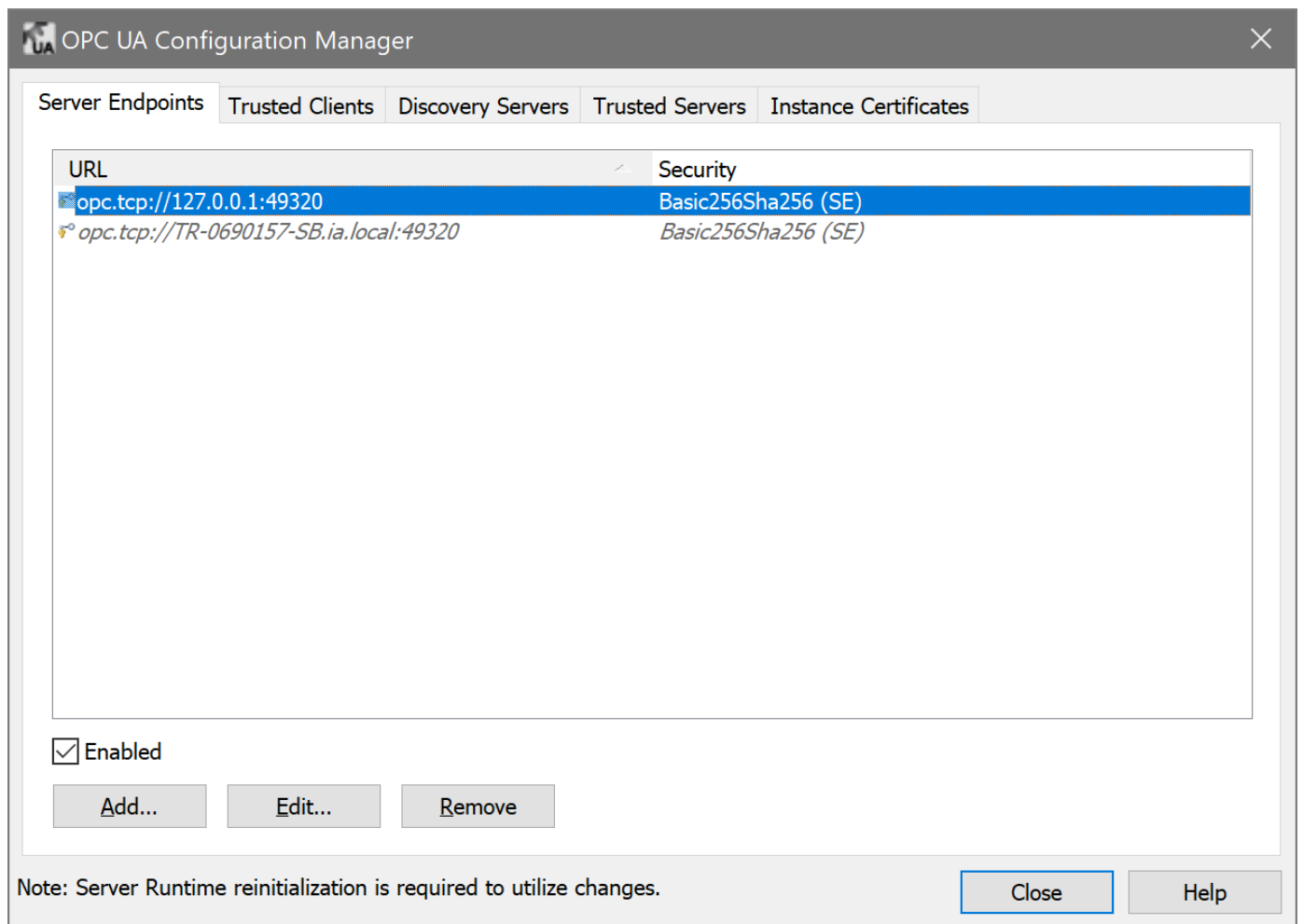
Back

Create OPC Connection

10. Click **Create OPC Connection**.

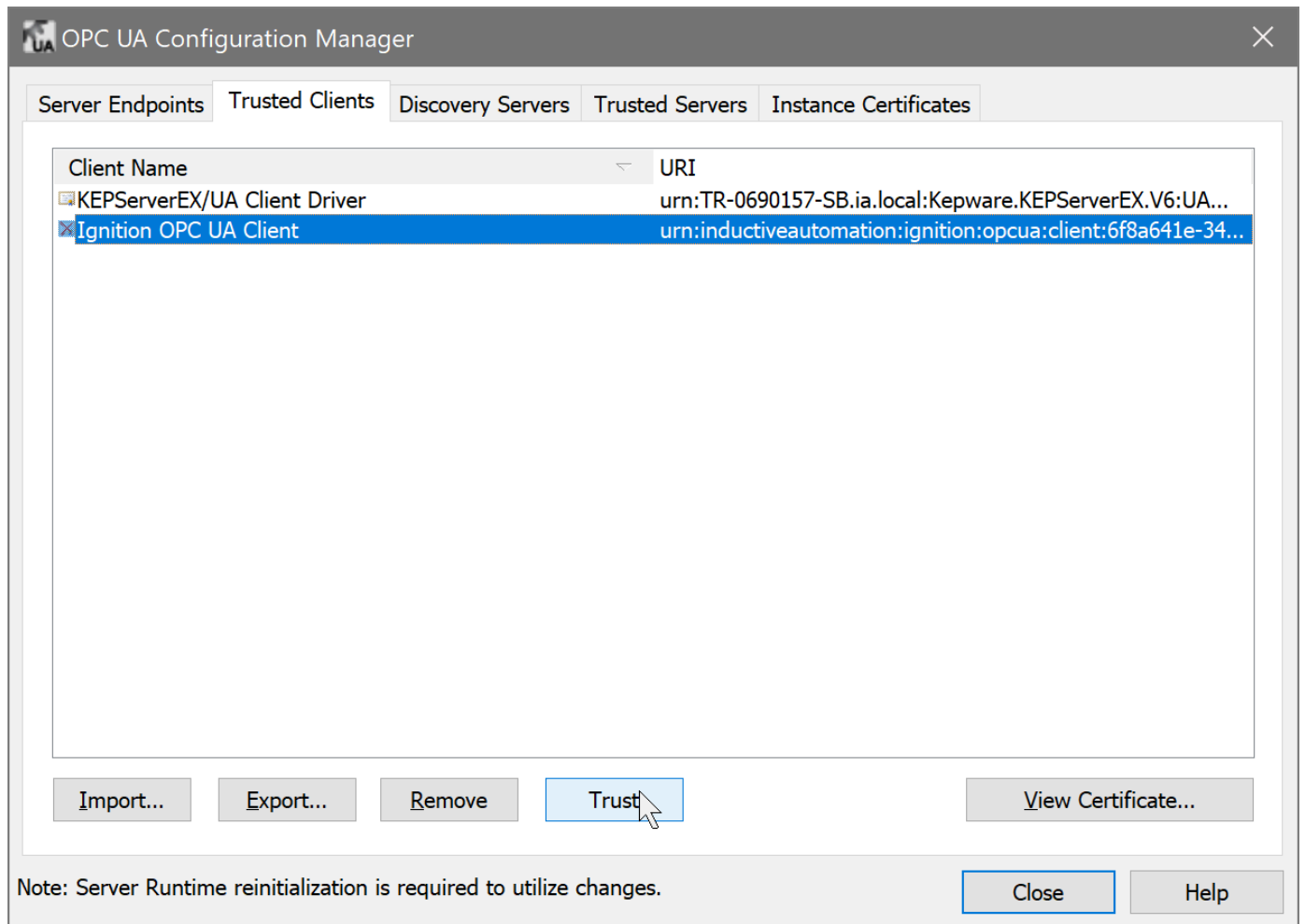
11. The connection will appear as **Faulted**. This is expected because KEPServerEX is denying access to the Ignition OPC UA Client. The next step is to have KEPServerEX trust the Ignition OPC UA Client.

12. On the computer that KEPServer is installed on, right-click 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**.



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 **Connections** section, go to the **OPC > Security** page.
17. Under the Client tab, you will find your new connection listed as Quarantined. Click on the three dot menu and select **Trust**.

Quarantined Certificates

Filter quarantinedcerts... View 20

COMMON NAME	SHA-1 FINGERPRINT	EXPIRATION	
> KEPServerEX/UA Server	7c9691534ffda86c3e60f1697bcb6	09/29/2023	⋮

Download

✓ Trust

Delete

18. Go back to the **Connections** section of the Gateway, to **OPC > Connections**. The status of your KEPServer connection should be **Connected**.
19. To test your tag connections, go to the **OPC > 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 Faulted status for the server connection to see the error message that will 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, expand the three dots menu and select **Edit** to open the server properties. Scroll down to the Failover section to select the **Failover Enabled** property and specify the **Failover Endpoint**. Click **Save Changes** when finished.

Note that 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 **Connections** section in the Gateway Webpage.
2. Select **OPC > Security** from the left side of the page.
3. On the **Client** tab, expand the three dots menu for the desired certificate to select Download, 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

OPC Classic 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. Other operating systems must use OPC UA.

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 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 a new OPC DA server connection.

Install OPC DA Server Connection

1. Go to the Ignition Gateway **Connections** section.
2. Go to **OPC > Connections**, and then select **Create OPC Connection**.
3. Choose the **OPC-DA COM Connection** and click **Next**.

Create OPC UA Server Connection

Create OPC UA Server Connection

1. Connection Type

☐ **OPC UA**
Connect to a device or server that supports OPC UA.

☒ **OPC-DA COM Connection**
Provides access to legacy COM-based OPC-DA servers. Supports OPC-DA versions 2 and 3.

Cancel Next

4. Enter the name for your connection.

5. Under Server, select whether you want to make a Local connection, or a Remote connection if the OPC server resides on 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.

- **Local:** Selecting a local connection displays a dropdown selection with a list of the available and running OPC servers located on the local machine.
- **Remote:** Selecting a remote connection additionally displays a Host Machine setting where you will specify the host name or IP address of the machine the OPC server resides on, which will update the server list under ProgId. Optionally, you can also enter the CLSID of the server. This id can be found in the registry of the machine hosting the server under:

HKEY_CLASSES_ROOT\OPCServerName\CLSID.

Create OPC UA Server Connection

Create OPC UA Server Connection

1. Connection Type

2. Configuration

SERVER

Connection Type

Local

Select either a local or remote connection type.

ProgId

Each OPC server is identified by a "progid". This is the name that will be used to connect to the server.

OPC SETTINGS

*Required

Use DA2 When Possible

☐ Causes the system to use DA version 2 calls even if the server supports version 3.

Use DA2 Flat Browse

☐ When using DA2 browsing, use "Flat" instead of "Hierarchial" mode.

Use Async Operations

☒ Causes the system to use OPC's async read/write functions. Recommended for better performance.

Async Operation Timeout *

30000

Back

Create OPC Connection

6. Select the OPC server you wish to connect to from the list. In the case where your server is not listed, you can manually enter the ProgId.
7. When you are finished fine tuning the settings, click **Create OPC Connection**. You will be redirected to the OPC UA 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 can help with determining why your connection is faulted and how to go about fixing the issue. If you are still having issues after exhausting these options, 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 describes possible OPC-COM connection problems with their solutions. 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 where an installed OPC Server 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 enter the ProgId manually.

With all the correct information about the OPC server a valid connection to the OPC Server can be made. This however is rare. Most of the time when the server is not detected, 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 Page

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

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

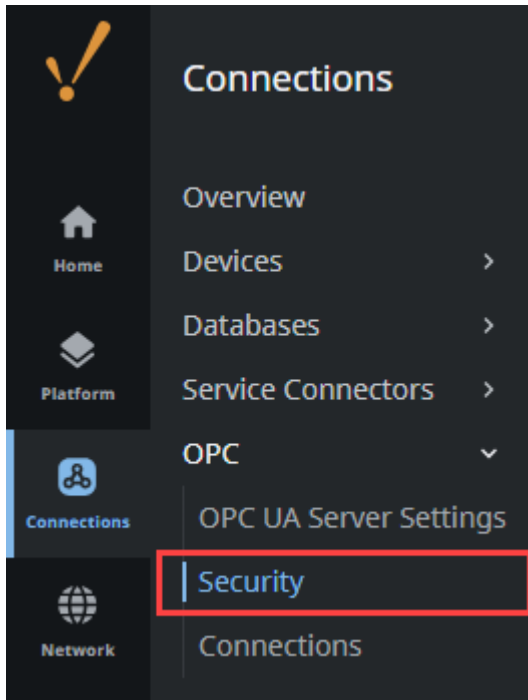
Follow these steps to open up the DCOM security settings on the machine that is running the OPC server:

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 Settings

On the OPC UA Security Settings 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 Security Settings page is located under the Gateway's **Connections** section, under **OPC > Security Settings**:



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 a 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 **Connections** tab, select **OPC > Security Settings**.
2. Click the **Client** tab or **Server** tab, depending on the what certificate you're uploading.
3. Click **Upload Trusted Certificate**.

4. Navigate to the location of the certificate on your system and click **Open**. Alternatively, you can drag the certificate file onto the page.
5. If the upload was successful, you'll see the name of the certificate and a success message. 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 three dot menu and select **Download**.
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 three dot menu and select **Delete**.
2. The certificate is deleted.

To view more information about a trusted certificate, use the **Expand** > icon.

Upload Trusted Certificate 

EXPIRATION	
05/12/2023	
03/20/2023	
03/22/2023	

 Download

 Delete

OPC Security Settings Page Details

Trusted Certificates

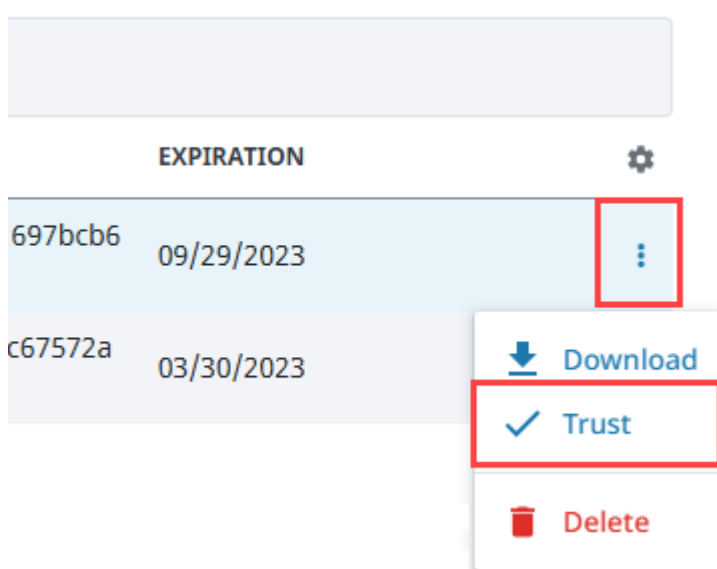
Column Name	Description
Common Name	Name of the certificate.
SHA-1 Fingerprint	The SHA-1 (Secure Hash Algorithm 1) fingerprint is the unique identifier of the certificate.
Expiration	Date the certificate will expire.

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.

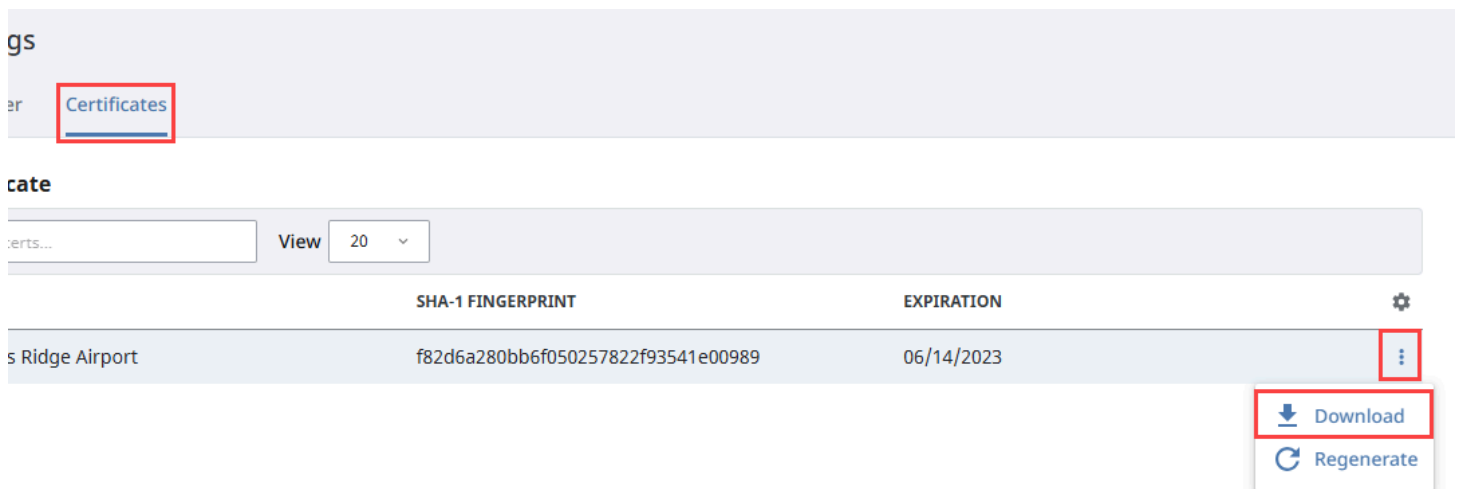
Quarantined Certificates

If you import a certificate that is not trusted, it will appear under the Quarantined Certificates list. From here you can expand the certificate details, **Trust** the certificate, or **Delete** it.



Certificates Tab

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 expanding the dropdown. The certificates can be downloaded by clicking the three dot menu and selecting **Download**. This will perform the same action as downloading a certificate from the **Client** tab as described above.



The Certificates tab menu options also include a **Regenerate** option. Selecting this 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.

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, the DNS names, and IP addresses in the Subject Alternate Name (SAN) fields.

Regenerate Server Certificate

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

Validity Length (Days)

1095

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 address entries to be included in the Subject Alternate Name field.

Cancel

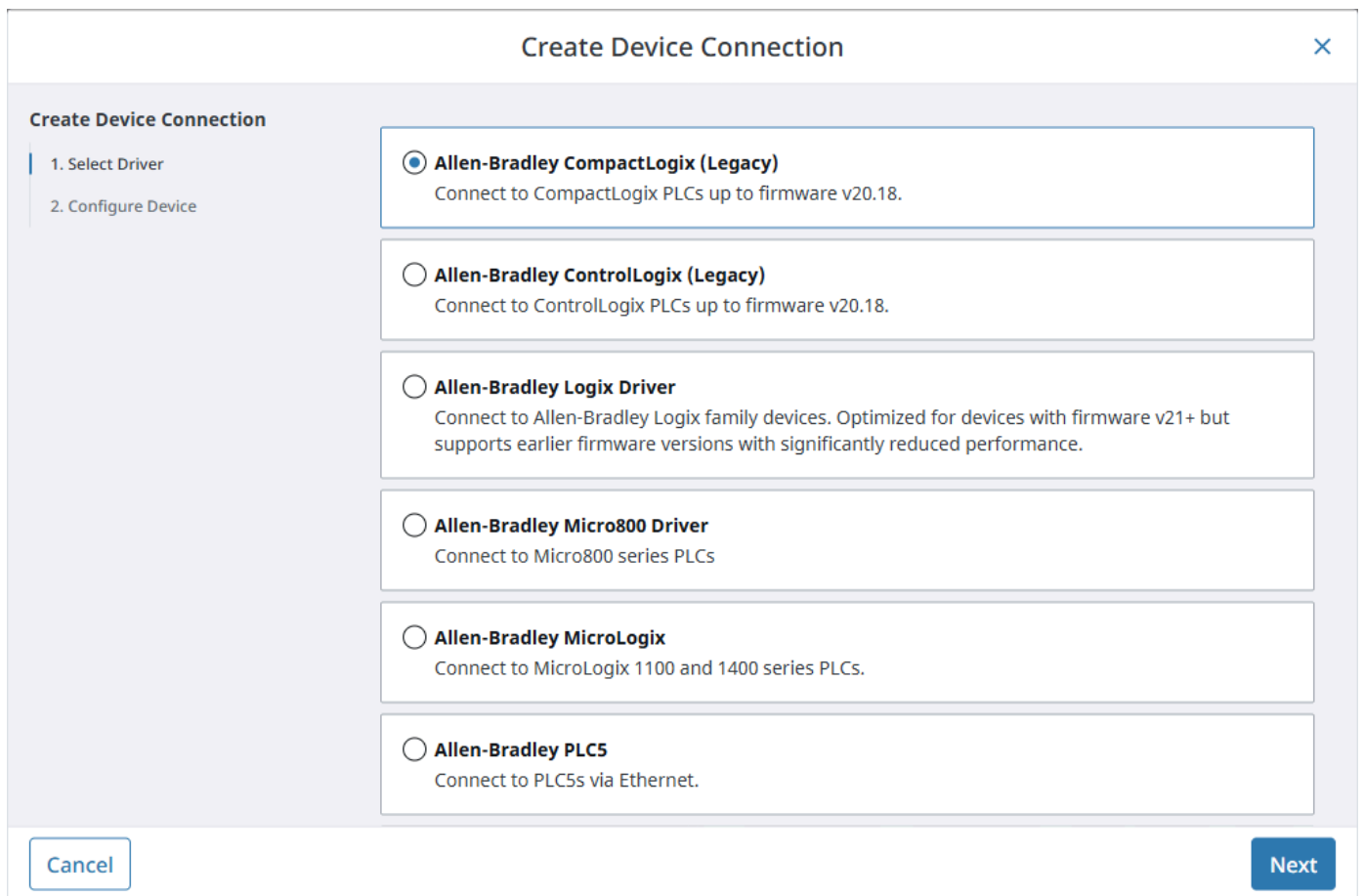
Regenerate

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.

To connect to a supported device:

1. Go to **Connections > Devices > Connections** on the Gateway.
2. Click **Create Device Connection**.
3. Choose a driver from the list of available drivers.



The screenshot shows a dialog box titled "Create Device Connection" with a close button (X) in the top right corner. On the left side, there is a sidebar with the title "Create Device Connection" and two steps: "1. Select Driver" (which is active) and "2. Configure Device". The main area of the dialog displays a list of six Allen-Bradley drivers, each with a radio button and a description:

- ☒ **Allen-Bradley CompactLogix (Legacy)**
Connect to CompactLogix PLCs up to firmware v20.18.
- ☐ **Allen-Bradley ControlLogix (Legacy)**
Connect to ControlLogix PLCs up to firmware v20.18.
- ☐ **Allen-Bradley Logix Driver**
Connect to Allen-Bradley Logix family devices. Optimized for devices with firmware v21+ but supports earlier firmware versions with significantly reduced performance.
- ☐ **Allen-Bradley Micro800 Driver**
Connect to Micro800 series PLCs
- ☐ **Allen-Bradley MicroLogix**
Connect to MicroLogix 1100 and 1400 series PLCs.
- ☐ **Allen-Bradley PLC5**
Connect to PLC5s via Ethernet.

At the bottom of the dialog, there are two buttons: "Cancel" on the left and "Next" on the right.

4. Fill in the connection details like IP address or slot number.
5. Click **Create Device Connection** to finish configuration.

Once connected, your device appears on the Connections page with its current status.

Device Diagnostics Tags

All drivers include built-in diagnostic tags that can be used to monitor the health of the device connection.

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.
Internal Indicator (IIN)	New in 8.3.4 Indicates additional diagnostic information specific to the DNP3 and DNP3 (Legacy) drivers. See the Internal Indicators table for more information.

Sampling Tags

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.

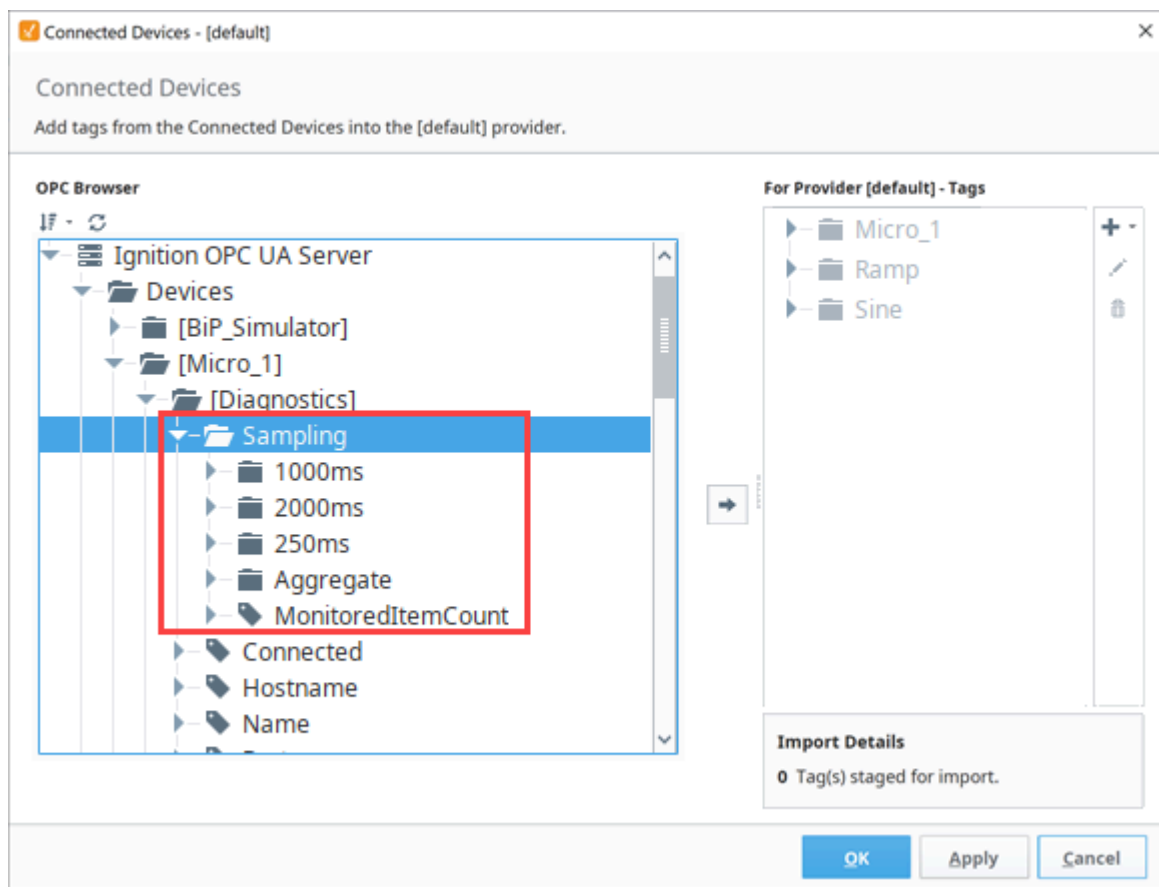
Additional **Sampling Group Folders** are provided based on active subscriptions, and are grouped by sample rate. These folders contain tags that represent throughput of subscription items at set rates. For example, in the image of the [Connected Devices window](#) below we see several folders next to

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](#).



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 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 Available in Aggregate Folder 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

i 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 off 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	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.

Device Details

Some devices have additional metrics you can view on the Gateway **Connections > Devices > Connections** page. To see these metrics, click the desired device connection's three dots menu and select **View Details**.

Aggregate Statistics

The metrics below include various aggregate statistics and monitored item count.

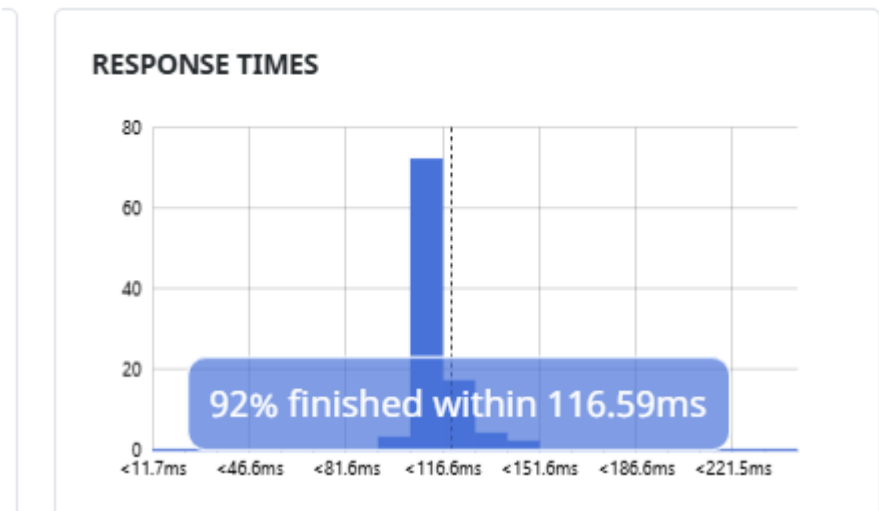
Attribute	Description
Request Count	The number of requests currently scheduled from the driver to the device. A request is a group of tags or items that the driver has grouped together to be read at the same time. Each driver forms these groups based on the protocol being used, and occasionally configuration settings in the driver.
Throughput (Mean)	The average number of requests that come through per second, starting from when the device was last initialized. This value will be reset if the device connection is edited and saved, since this causes the device connection to reinitialize.
Throughput (1 min)	The average number of requests that come through per second within the last minute.
Mean Response Time	The average time it takes for Ignition to get a response from the device. This value is an average based off the Response Times chart .
Monitored Item Count	The total number of items that the device connection is subscribed to, including any Device Diagnostic tags listed above .

Response Times Chart

The Details panel also includes a bar chart that illustrates the distribution of requests by grouping them based on their recorded response times. The x-axis measures the request group's response time in milliseconds. The y-axis measures the percentage of requests that fall within a response time group, calculated from the total number of recorded response times.

When hovering over the bar chart, a tooltip will display the total percentage of requests that were completed within the time frame you are hovering over. This tooltip value can be different than each

individual bar's y-axis value, since the tooltip takes the aggregate y-axis values of the requests that are within the same group or faster.



Log Activity

The Log Activity section displays loggers related to the device connection. Changing the Min Level will filter out any logs that do not meet the set logging level. Options include:

- INFO
- DEBUG
- TRACE

You can also toggle whether the activity updates with live values, or merge your device connection logs with the Gateway [Diagnostic logs](#).

Log Activity

Min Level

INFO

Live Values

Merge to Logs

LOGGER	TIME	MESSAGE	
> WARN NegotiatePDURequest	10Sep2026 13:07:23	NegotiatePDURequest failed. FailureType: DISCONNECTED	Q
> WARN ConnectRequest	10Sep2026 13:07:23	ConnectRequest failed. FailureType: DISCONNECTED	Q
> WARN ConnectRequest	10Sep2026 13:07:23	ConnectRequest failed. FailureType: TIMEOUT	Q

Allen-Bradley Ethernet

Connecting to Allen-Bradley Devices

Ignition provides several drivers for Allen-Bradley devices. These drivers connect through the Gateway to devices that support Ethernet communication or devices with Ethernet modules like ENBT or EN2T. The following device drivers are available in the Allen-Bradley Ethernet module:

- **CompactLogix (Legacy)**: Supports legacy firmware versions v20.18 and prior.
- **ControlLogix (Legacy)**: Supports legacy firmware versions v20.18 and prior.
- **Logix**: Supports ControlLogix and CompactLogix devices with firmware v20.19 and later.
- **Micro800**: Supports Micro800 models including 820, 850, and 870.
- **MicroLogix**: Supports MicroLogix models including 1100 and 1400 series.
- **PLC5**: Supports direct communication with Allen-Bradley PLC5 controllers.
- **SLC**: Supports the 5/05 series using Ethernet.

NOTE

These drivers only support firmware versions 16 and above. Earlier versions may work, but are not officially supported. For unsupported versions, consider using a [third party OPC server](#).

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 result in communication issues.

Verifying Device Connectivity

Device connections can be verified from the **Gateway Webpage** under **Connections > Devices > Connections**. Check the Status column to confirm that the device shows Connected.

Allen-Bradley Connection Paths

All Allen-Bradley drivers include a **Connection Path** setting. This setting is required when you're connecting to a processor through a bridge device or an external Ethernet module, such as an ENBT or EN2T. For example, if you're communicating with a ControlLogix or CompactLogix processor via a ControlLogix Gateway, you'll need to define a connection path.

The connection path tells the driver how to navigate through backplanes, modules, and network links to reach the destination processor. Because hardware configurations vary widely, the path is unique to your setup. However, most connection paths follow a consistent pattern. For detailed examples and templates, refer to the individual driver documentation.

How to Build a Connection Path

Building a connection path involves identifying each segment between Ignition and the processor, then expressing those steps in the expected format. The path is always defined in pairs: an action (such as exiting a port) followed by a destination (such as an IP address or slot).

1. Begin the path with a 1, which tells the driver to move from the current module (such as an Ethernet card) into the chassis backplane, where it can access other modules by slot.
2. Enter the slot number of the module you want to route through.
3. Specify the exit port or channel on that module.
4. Provide the address of the next module in the path—this could be an IP address, ControlNet address, or DH+ station number, depending on the network type.

Repeat this pattern as needed until you reach the processor.

NOTE

Since each step consists of two entries, all valid connection paths must contain an even number of values. If your path ends in an odd number, a step is likely missing.

Module Entry and Exit Codes

Each movement in the path consists of two entries: one for exiting a module and one for entering the next. This means all valid paths contain an even number of entries. If your path ends with an odd number of entries, a step is missing.

By following the path through your networked modules, you can configure communication from Ignition to any accessible Allen-Bradley processor.

Use the following codes based on the module type:

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 (octal 0-77)

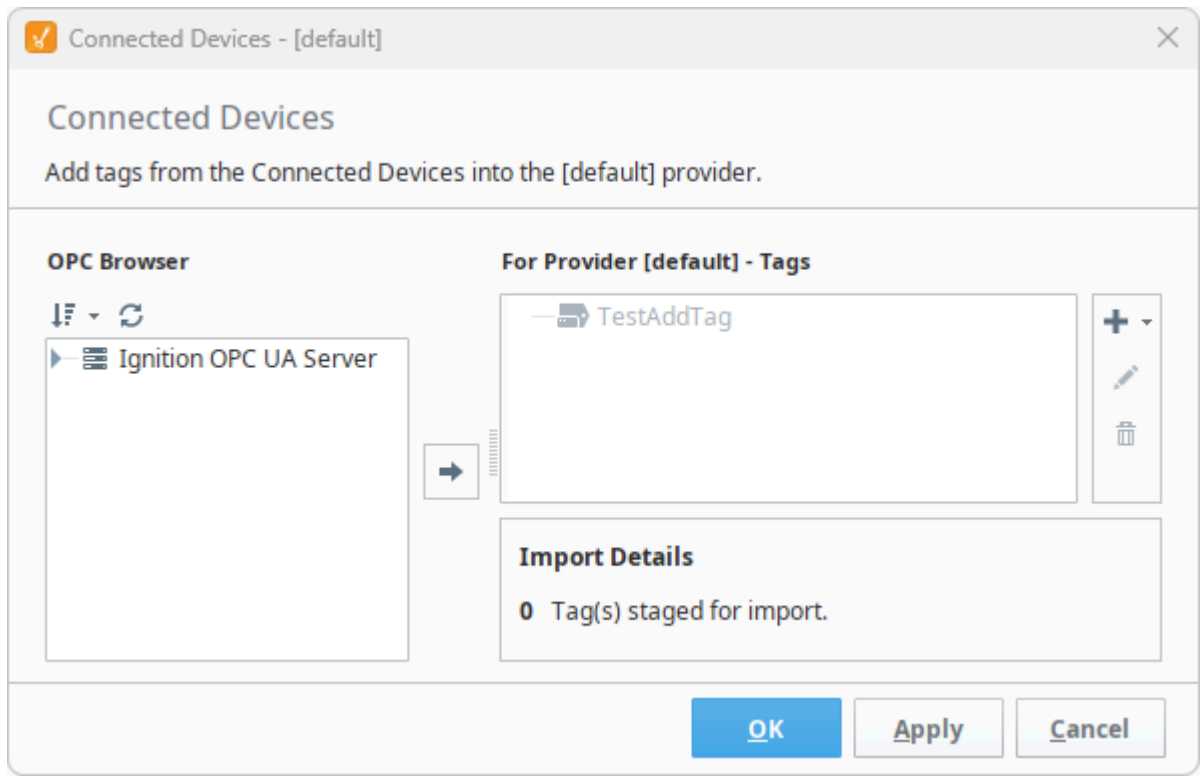
Allen-Bradley Addressing

Ignition's Allen-Bradley drivers all use tag-based (symbolic) addressing, meaning Ignition will automatically browse the connected device for data points that can be added to the Tag Browser. Additionally, the Micro800 driver allows users to either create entries for individual data points, or import configuration files.

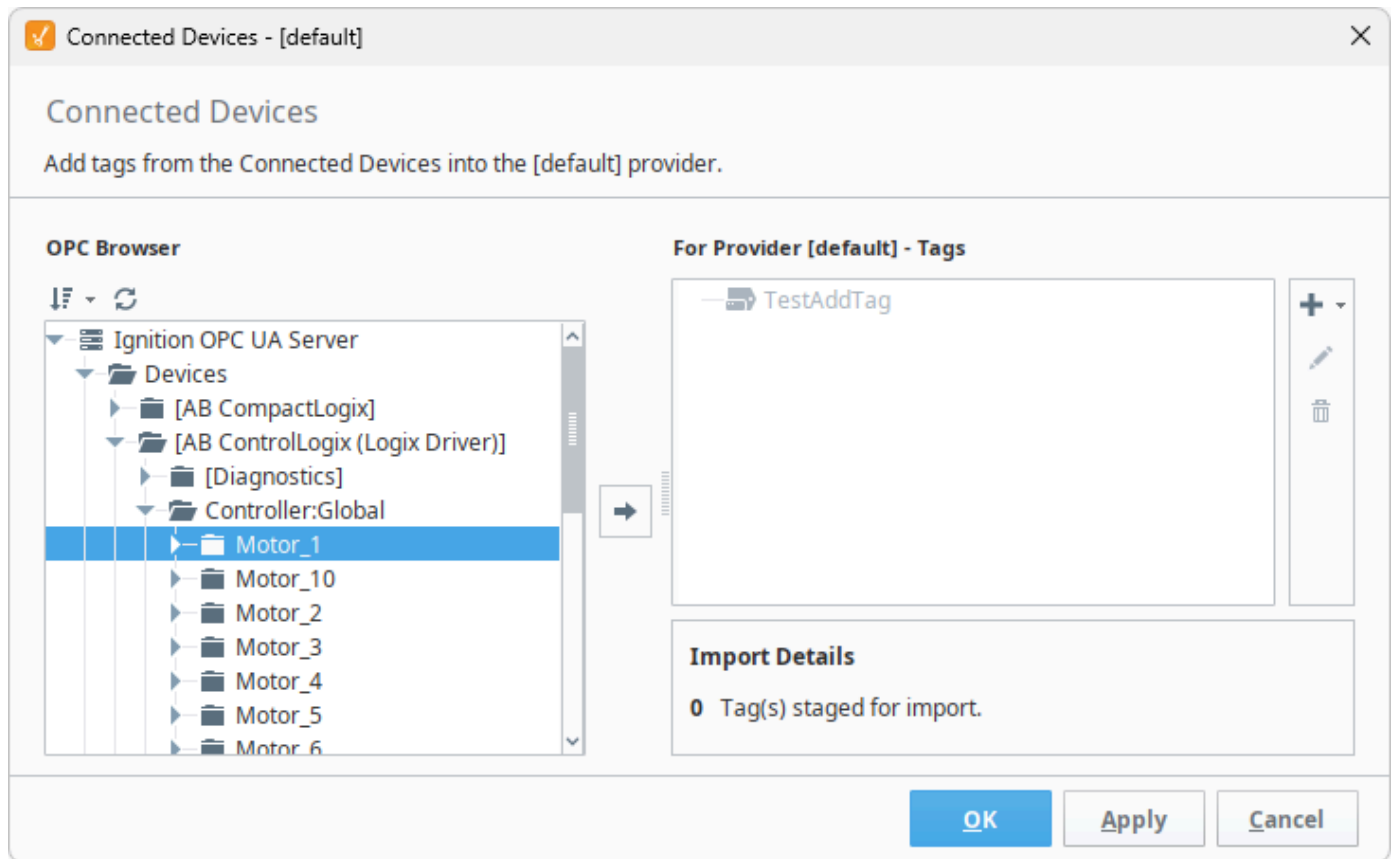
Addressing Example

This example uses the **Logix** driver to browse for data points within the Tag Browser's Connected Devices window and assumes you have already set up a connection to a ControlLogix PLC. For instructions on connecting to a device using the Logix driver, see the **Connect to a Logix Device** section.

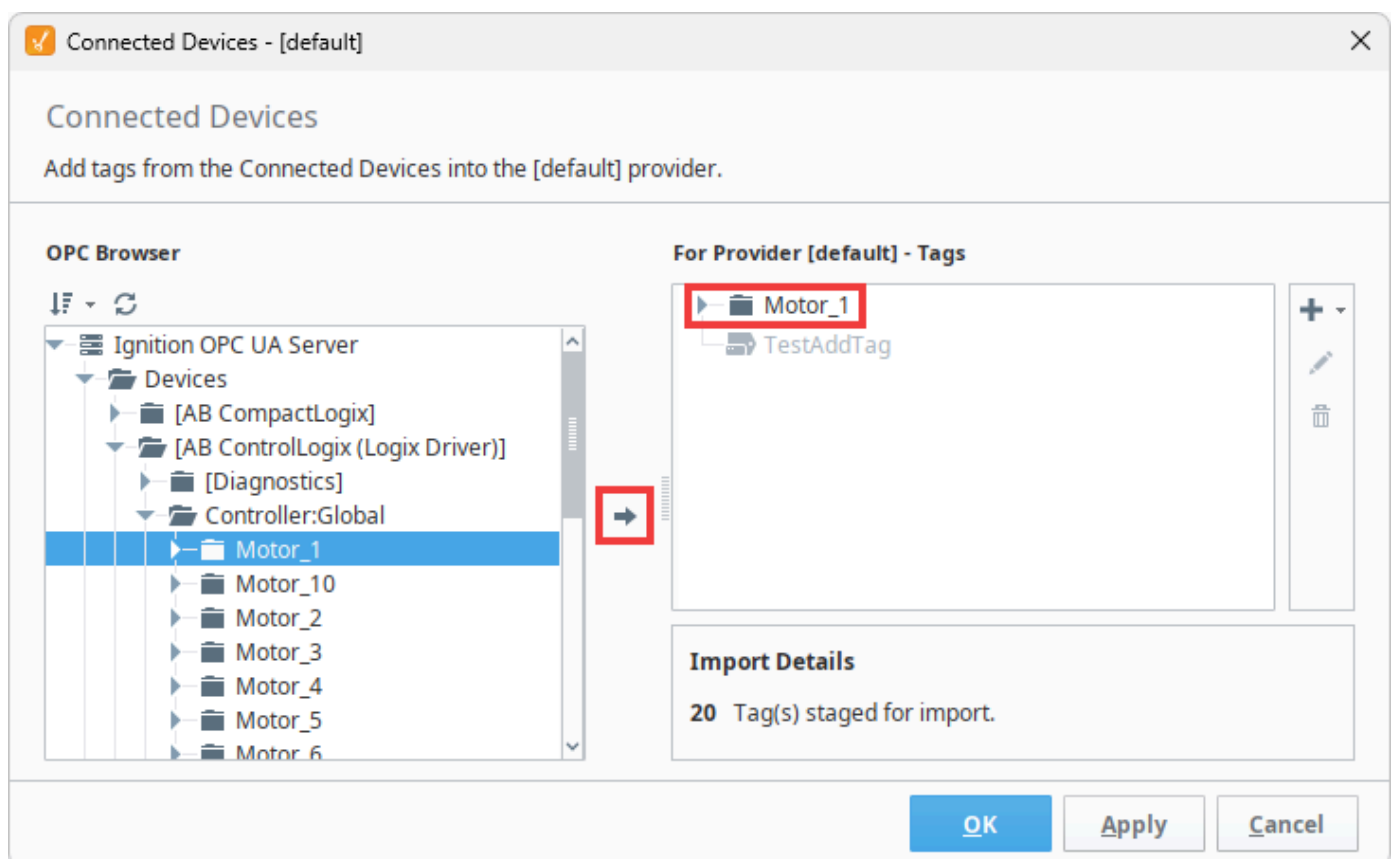
1. In your Tag Browser, click on the **Add +** icon, then click on **Browse Devices...** . The Connected Devices popup window will open.



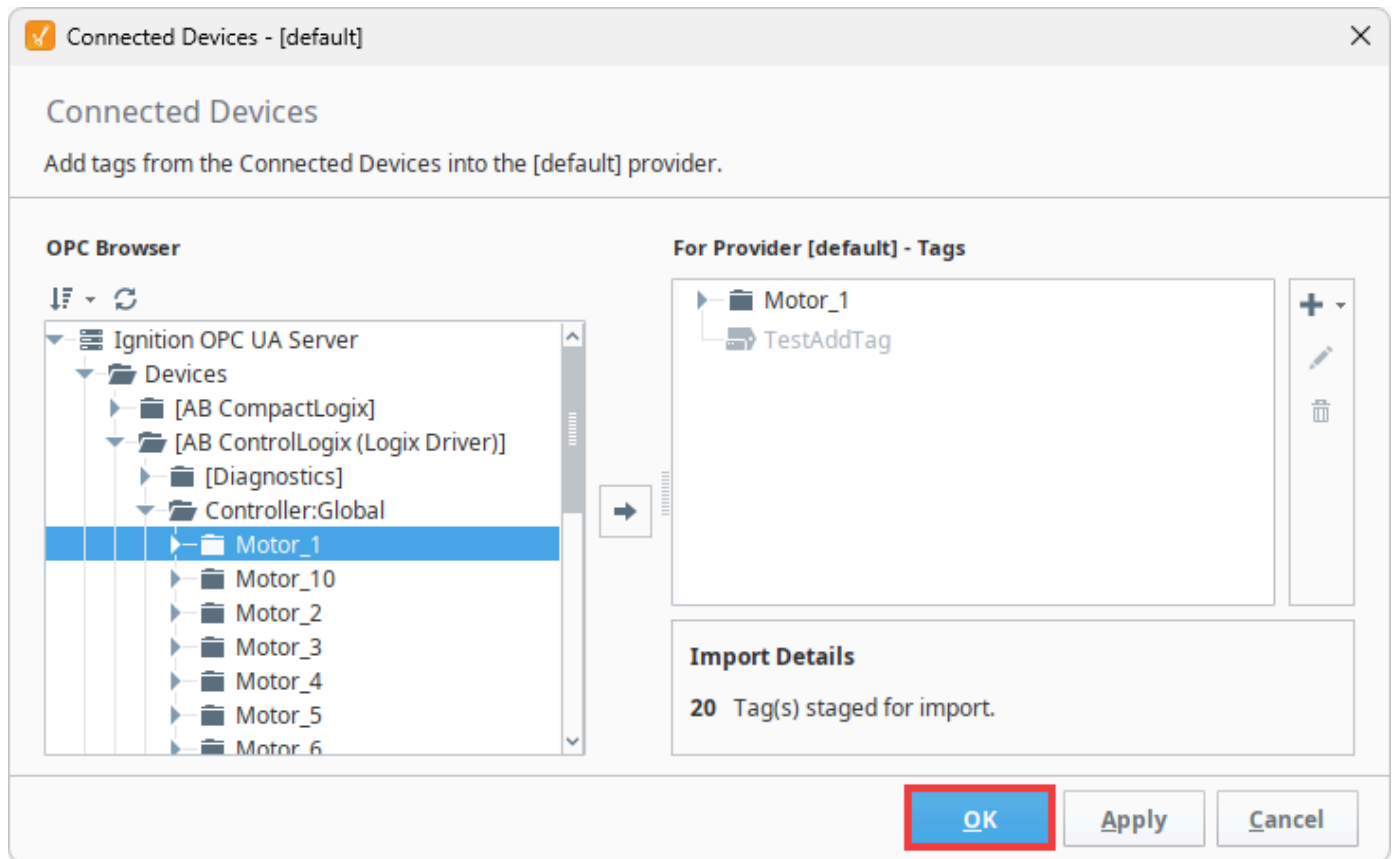
2. Navigate to the tag you want to import. In this example, since we want **Motor_1** within our connected device, we'll go through **Ignition OPC UA Server > Devices > [AB ControlLogix (Logix Driver)] > Controller:Global > Motor_1**.



3. Select the tag or folder to import and add it to your Tag Provider by clicking on the Add ➔ icon. This will stage the tag for when you are ready to import, in case you want to add any other tags or folders.



4. Once you have added the tags and folders you want, click **OK** to add them to your Tag Browser.



Your tags and folders are now available to use in Ignition.

Configuring Data Points

You can expose individual data points to your device connection program so that Ignition knows where to browse for them. Alternatively, if you have a CSV or ZIP file, you can import those instructions to avoid needing to configure data points manually. Currently, only the Allen-Bradley Micro800 driver supports manually adding data points.

Besides importing configuration files, there are various other options you can select, all of which are outlined by the table below:

Button	Description
Import Configuration File	Allows users to import a CSV or ZIP file to add many data points at once.
Reset	Resets the program to its default state before any changes were saved.
Export	Allows users to export the configuration file as a CSV.
Remove All	Removes all entries from the program.
Add +	Allows users to manually add individual data points.

When adding individual data points or importing configuration files, it is helpful to know what fields Ignition is expecting. These fields are detailed by the table below:

Property	Description
Browse Name	The path to the data point on the PLC.
Data Type	The data point's data type. See the Micro800's Data Types section for a list of supported data types.
Dimension	The structure of the PLC's data points, defined by arrays. This lets you use arrays in different dimensions, such as vectors and matrices. For example, <code>[0..9]</code> is a one dimensional array with 10 elements, while <code>[0..4,0..5]</code> is a two dimensional array with 30 elements (five rows by six columns).
String Size	The maximum size of the string. Applicable only for STRING data types.
Read Write	The permission level Ignition has to these data points. Options include • R (Read) • W (Write) • RW (Read and Write)

Configuring Data Points Example

This example demonstrates how to add a single data point to your Micro800 device connection and assumes you have already set up a connection to a Micro800 PLC. For instructions on connecting to a device using the Micro800 driver, see the [Connect to a Micro800 Device](#) section. If you want to add multiple data points at the same time, see the Micro800 driver page for examples on importing and using [CSV](#) and [ZIP](#) files.

- 1. Navigate to **Gateway Webpage > Connections > Devices > Connections**.
- 2. Click the **three dots menu** of the Micro800 device you want to add a data point for, then click **Addresses**.

☐	AB Micro820	Allen-Bradley Micro800 Driver	true	CONNECTED	Edit Rename Duplicate Addresses View Details
☐	AB MicroLogix	Allen-Bradley MicroLogix	true	DISCONNECTED	
☐	AB PLC5	Allen-Bradley PLC5	true	CONNECTED	
☐	AB SLC	Allen-Bradley SLC	true	DISCONNECTED	
☐	DNP3	DNP3 Driver	true	CONNECTED	

- 3. The Program panel for your Micro800 device will open. Click on the **Add +** button on the top right of the panel. This will add a new entry to the table where you can specify a data point.

×

AB Micro820 Program

Import Configuration File

Reset

Export

Remove All

Add +

BROWSE NAME

DATA TYPE

DIMENSION

STRING SIZE

READ WRITE

- 4. Enter the information for the data point you want to expose, then click **Save Changes** on the bottom right.



AB Micro820 Program

Import Configuration File 

Reset 

Export 

Remove All

Add +

BROWSE NAME	DATA TYPE	DIMENSION	STRING SIZE	READ WRITE	
MyRTC.Year	UINT 			RW 	Delete
MyRTC.Month	UINT 			RW 	Delete
MyRTC.Day	UINT 			RW 	Delete

Showing 3 of 3 items

Cancel

Save Changes

Your data point is exposed to Ignition. You can now browse for this data point in the Tag Browser's Connected Devices window and add it to Ignition by following the [addressing example](#) above.

Connecting to CompactLogix

NOTE

This driver supports CompactLogix firmware up to version 20.18.



INDUCTIVE
UNIVERSITY

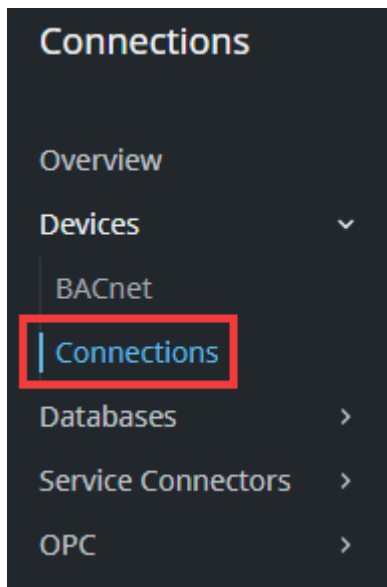
Connecting to Legacy
CompactLogix

[Watch the video](#)

Connect to a CompactLogix Device

This driver requires the OPC UA module to be installed and enabled. If the module is missing or disabled, the device will fault with a “Missing Dependency” error on the **Platform System > Modules** page.

1. On the Gateway, go to **Connections > Devices > Connections**.



2. Click **Create Device Connection +**.
3. Select **Allen-Bradley CompactLogix (Legacy)** from the list and click **Next**.

Create Device Connection

Create Device Connection

1. Select Driver

2. Configure Device

☒ **Allen-Bradley CompactLogix (Legacy)**
Connect to CompactLogix PLCs up to firmware v20.18.

☐ **Allen-Bradley ControlLogix (Legacy)**
Connect to ControlLogix PLCs up to firmware v20.18.

☐ **Allen-Bradley Logix Driver**
Connect to Allen-Bradley Logix family devices. Optimized for devices with firmware v21+ but supports earlier firmware versions with significantly reduced performance.

☐ **Allen-Bradley Micro800 Driver**
Connect to Micro800 series PLCs

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

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

Cancel

Next

4. On the **Configure Device** screen, enter the required fields:
- **Name:** The desired name for the device (e.g., CompactLogix)
 - **Hostname:** The IP address of the PLC (e.g., 10.20.4.55)

Create Device Connection

Create Device Connection

1. Select Driver

2. Configure Device

Configure Device

MAIN

*Required

Name *

CompactLogix

The name of this device connection.

Description

Enabled

☒ Set whether this device connection is enabled or disabled within the Gateway.

CONNECTIVITY

*Required

Hostname *

10.20.4.55

Local Address

Address of network adapter to connect from.

Timeout *

2000

Connection Path

Show Advanced +

Back

Create Device Connection

5. Click **Create Device Connection**.

The device will be listed now on the Connections page. The status will initially show Disconnected, but will move to a Connected state.

Device Connections

Connections

Filter device connections...	View 20	Create Device Connection +		
☐ NAME	DRIVER	ENABLED	STATUS	
☐ CompactLogix	Allen-Bradley CompactLogix (Legacy)	true	CONNECTED: PROTOCOL: EIP - RUN MODE	

Showing 1 of 1 items

Device Connection Settings

General

Name	Description
Name	A user-defined name for the device. This name appears in OPC item paths and the Devices list. Name must begin with an underscore or a letter, and cannot include special characters.
Description	An optional description for the device.
Enabled	Enables or disables the device connection. Only enabled devices are accessible in the system.

Connectivity

Name	Description
Hostname	IP address of the target device or Ethernet module (such as ENBT or EN2T).
Local Address	IP address of the local interface to use when connecting. Leave blank to auto-select.
Timeout	Connection timeout in milliseconds.
Connection Path	Used to define the route to the processor. The Connection Path format contains four comma separated numbers. 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. See Allen-Bradley Connection Paths for more information.

Advanced

Name	Description
Disable Automatic Browse	Disables automatic browsing of the tag structure.
Show String Arrays	Enables visibility of string arrays in the address space.
Status Request Poll Rate	Interval in milliseconds for polling the device status.
Max Concurrent Requests	Number of simultaneous requests allowed. Larger values may improve performance but can overwhelm the device.

Connecting to ControlLogix

NOTE

This driver supports ControlLogix firmware up to version 20.18.



INDUCTIVE
UNIVERSITY

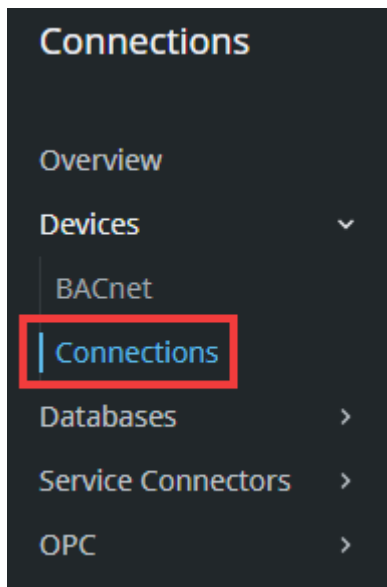
Connecting to Legacy
ControlLogix

[Watch the video](#)

Connect to a ControlLogix Device

This driver requires the OPC UA module to be installed and enabled. If the module is missing or disabled, the device will fault with a “Missing Dependency” error on the **Platform System > Modules** page.

1. On the Gateway, go to **Connections > Devices > Connections**.



2. Click **Create Device Connection +**.
3. Select **Allen-Bradley ControlLogix (Legacy)** and click **Next**.

Create Device Connection

Create Device Connection

1. Select Driver

2. Configure Device

☐ **Allen-Bradley CompactLogix (Legacy)**
Connect to CompactLogix PLCs up to firmware v20.18.

☒ **Allen-Bradley ControlLogix (Legacy)**
Connect to ControlLogix PLCs up to firmware v20.18.

☐ **Allen-Bradley Logix Driver**
Connect to Allen-Bradley Logix family devices. Optimized for devices with firmware v21+ but supports earlier firmware versions with significantly reduced performance.

☐ **Allen-Bradley Micro800 Driver**
Connect to Micro800 series PLCs

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

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

Cancel

Next

4. On the **Configure Device** screen, enter the required fields:
- **Name:** CLX
 - **Hostname:** The IP address of the PLC (e.g., 10.20.4.50)

Create Device Connection

Create Device Connection

1. Select Driver

2. Configure Device

Configure Device

MAIN

*Required

Name *

CLX

The name of this device connection.

Description

Enabled

☒ Set whether this device connection is enabled or disabled within the Gateway.

CONNECTIVITY

*Required

Hostname *

10.20.4.50

Local Address

Address of network adapter to connect from.

Timeout *

2000

Connection Path

Back

Create Device Connection

5. Click **Create Device Connection**.

The device will be listed now on the Connections page. The status will initially show Disconnected, but will move to a Connected state.

Device Connections

Connections

Filter device connections... View 20 Create Device Connection +				
☐	NAME	DRIVER	ENABLED	STATUS
☐	CLX	Allen-Bradley ControlLogix (Legacy)	true	CONNECTED: PROTOCOL: EIP - RUN MODE

Showing 1 of 1 items

Device Connection Settings

General

Name	Description
Name	A user-defined name for the device. This name appears in OPC item paths and the Devices list. Name must begin with an underscore or a letter, and cannot include special characters.
Description	An optional description to help identify the device.
Enabled	When selected, the device is active and available for use in the system.

Connectivity

Name	Description
Hostname	IP address of the Ethernet module (e.g., 1756-ENET) used to connect to the ControlLogix processor.
Local Address	IP address of the local network adapter to bind to. Leave blank to auto-select.
Timeout	Timeout in milliseconds for a request to receive a response from the device.
Connection Path	Used to define a route to the ControlLogix or downstream PLC-5 processor. See Allen-Bradley Connection Paths for more information.

Advanced

Name	Description
Disable Automatic Browse	Disables automatic browsing of the tag tree.
Show String Arrays	Displays string arrays from the device if supported.
Status Request Poll Rate	Frequency in milliseconds for polling the device status.
Slot Number	The zero-based slot number of the ControlLogix processor in the chassis.
Max Concurrent Requests	Number of simultaneous requests the driver sends to the device. Higher values may improve throughput but can overload the PLC.

Supported Connection Methods

This driver supports ControlLogix 5500 devices connected through a 1756-ENET/A or 1756-ENET/B module.

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

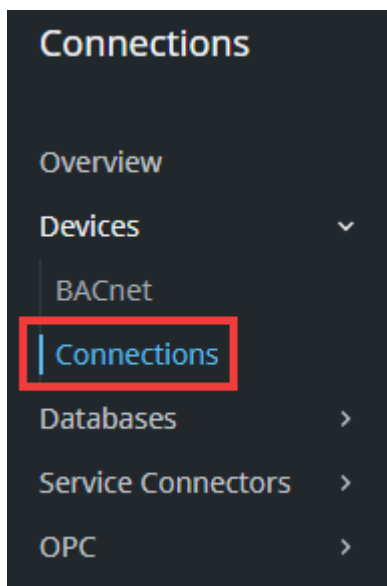
Connecting to ControlLogix
v21

[Watch the video](#)

Connect to a Logix Device

This driver requires the OPC UA module to be installed and enabled. If the module is missing or disabled, the device will fault with a “Missing Dependency” error on the **Platform System > Modules** page.

1. On the Gateway, go to **Connections > Devices > Connections**.



2. Click **Create Device Connection +**.
3. Select **Allen-Bradley Logix Driver** and click **Next**.

Create Device Connection

Create Device Connection

1. Select Driver

2. Configure Device

☐ **Allen-Bradley CompactLogix (Legacy)**
Connect to CompactLogix PLCs up to firmware v20.18.

☐ **Allen-Bradley ControlLogix (Legacy)**
Connect to ControlLogix PLCs up to firmware v20.18.

☒ **Allen-Bradley Logix Driver**
Connect to Allen-Bradley Logix family devices. Optimized for devices with firmware v21+ but supports earlier firmware versions with significantly reduced performance.

☐ **Allen-Bradley Micro800 Driver**
Connect to Micro800 series PLCs

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

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

Cancel

Next

4. On the **Configure Device** screen, enter the required fields:
- **Name:** LGX
 - **Hostname:** The IP address of the PLC (e.g., 10.20.1.54)

Create Device Connection

Create Device Connection

1. Select Driver

2. Configure Device

GENERAL

*Required

Name *

LGX

The name of this device connection.

Description

Enabled

☒ Set whether this device connection is enabled or disabled within the Gateway.

CONNECTIVITY

*Required

Hostname *

10.20.1.54

Port *

44818

Local Address

Address of network adapter to connect from.

Timeout *

2000

Max amount of time to wait, in milliseconds, for responses from the processor.

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.

Show Advanced +

Back

Create Device Connection

5. Click **Create Device Connection**.

The device will be listed now on the Connections page. The status will initially show Disconnected, but will move to a Connected state.

Connections

🔍

Filter device connections...

View

20

▼

Create Device Connection +

☐	NAME	DRIVER	ENABLED	STATUS	⚙️
☐	LGX	Allen-Bradley Logix Driver	true	RECONNECTWAIT	⋮

Showing 1 of 1 items

Driver Implementation

This driver maps boolean arrays as members of 32-bit DWORDs. For example, a **BOOL[64]** in the PLC is represented as a **DWORD[2]** in Ignition.

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

The Logix driver also supports the following data types:

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

Device Connection Settings

General

Name	Description
Name	A user-defined name for the device. This name appears in OPC item paths and the Devices list. Name must begin with an underscore or a letter, and cannot include special characters.
Description	Optional description to help identify the device.
Enabled	When selected, the device is active and 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	TCP port used for communication. The default is 44818.
Local Address	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 maximum amount of time to wait, in milliseconds, for responses from the processor.
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	When enabled, detects new tags and UDT changes and rebrowses automatically.
Identity Request Frequency	Frequency, in milliseconds, to request CIP Identity Object attributes.
CIP Connection Size	Size used in CIP Forward Open requests.
CIP Connection Timeout	Timeout, in milliseconds, for CIP connections and RPI.
Slot Number	The slot in which the processor resides. If not in a rack, use 0.
Max Concurrent Requests	Number of concurrent requests sent to the device. Increasing this may improve throughput but could overwhelm the processor.

Connecting to Micro800

The Micro800 driver enables Ethernet/IP communication over TCP/IP with Allen-Bradley Micro800 devices, including the 820, 850, and 870 series.

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



**INDUCTIVE
UNIVERSITY**

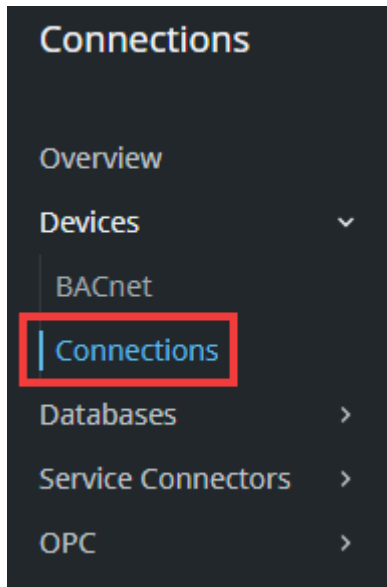
Connecting to Micro800

[Watch the video](#)

Connect to a Micro800 Device

This driver requires the OPC UA module to be installed and enabled. If the module is missing or disabled, the device will fault with a “Missing Dependency” error on the **Platform System > Modules** page.

1. On the Gateway, go to **Connections > Devices > Connections**.



2. Click **Create Device Connection +**.
3. Select **Allen-Bradley Micro800 Driver** and click **Next**.

Create Device Connection

Create Device Connection

1. Select Driver

2. Configure Device

☐ **Allen-Bradley CompactLogix (Legacy)**
Connect to CompactLogix PLCs up to firmware v20.18.

☐ **Allen-Bradley ControlLogix (Legacy)**
Connect to ControlLogix PLCs up to firmware v20.18.

☐ **Allen-Bradley Logix Driver**
Connect to Allen-Bradley Logix family devices. Optimized for devices with firmware v21+ but supports earlier firmware versions with significantly reduced performance.

☒ **Allen-Bradley Micro800 Driver**
Connect to Micro800 series PLCs

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

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

Cancel

Next

4. On the **Configure Device** screen, enter the required fields:

- **Name:** Micro800
- **Hostname:** The IP address of the PLC (e.g., 10.20.1.2)

Create Device Connection

Create Device Connection

1. Select Driver

2. Configure Device

Configure Device

MAIN

*Required

Name *

Micro800

The name of this device connection.

Description

Enabled

☒ Set whether this device connection is enabled or disabled within the Gateway.

CONNECTIVITY

*Required

Hostname *

10.20.1.2

Port *

44818

Local Address

Address of network adapter to connect from.

Timeout *

2000

Max amount of time to wait, in milliseconds, for responses from the processor.

Show Advanced +

Back

Create Device Connection

5. Click **Create Device Connection**.

The device will be listed now on the Connections page. The status will initially show Disconnected, but will move to a Connected state.

Device Connections

Connections

Filter device connections... View 20 Create Device Connection +				
☐	NAME	DRIVER	ENABLED	STATUS
☐	Micro800	Allen-Bradley Micro800 Driver	true	CONNECTED

Showing 1 of 1 items

Device Connection Settings

General

Name	Description
Name	A user-defined name for the device. This name appears in OPC item paths and the Devices list. Name must begin with an underscore or a letter, and cannot include special characters.
Description	An optional description for the device.
Enabled	Enables or disables the device connection. Only enabled devices are accessible in the system.

Connectivity

Name	Description
Hostname	IP address of the Micro800 device.
Port	Port number used to connect to the device (default is 44818).
Local Address	IP address of the local network adapter to use for the connection. Leave blank to auto-select.
Timeout	Timeout in milliseconds to wait for a response from the device.

Advanced

Name	Description
Write Priority Ratio	Number of write requests to execute for every read request when both types are queued.
Automatic Rebrowse	Monitors for tag changes and triggers re-browsing when detected.
CIP Connection Size	The CIP connection size used for Forward Open requests.
CIP Connection Timeout	Timeout in milliseconds for CIP connection attempts and request packet intervals (RPI).
Max Concurrent Requests	Maximum number of simultaneous unconnected and connected requests. Higher values may increase throughput but can overwhelm the device.

Browsing

The Micro800 driver supports browsing global variables. However, some variable types may not be browsable directly:

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

These global variables can still be made browsable in Ignition using the **Import Configuration File** option to load a **CSV** or **.zip** file downloaded from Connected Components Workbench.

 NOTE

This driver is designed for individual global variables exported from CCW. Function Blocks are not supported.

 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

Exporting from Connected Components Workbench

1. Open **Connected Components Workbench** and load the Micro800 project.
2. Right-click on the controller in the Project Organizer.
3. Select **Export > Variables** from the context menu.
4. Choose a save location and select **Comma Separated Values** as the format.
5. Save the file.

Importing the CSV File into Ignition

1. On the Gateway, go to **Connections > Devices > Connections**.
2. Locate your Micro800 device, click the **three dots menu**, and choose **Addresses**.
3. Click **Import Configuration File** and upload your CSV.
4. Review the imported entries, then click **Save Changes**.

Importing Global Variables - Zip File

Exporting from Connected Components Workbench

1. Open **Connected Components Workbench** and load the Micro800 project.
2. Right-click on the controller in the Project Organizer.
3. Select **Export > Export Device** from the context menu.
4. Check **Export Variables Only** and click **Export**.
5. Convert the exported .7z configuration file to .zip format using a file archiver tool.

Importing the Zip File into Ignition

1. On the Gateway, go to **Connections > Devices > Connections**.
2. Locate your Micro800 device, click the **three dots menu**, and choose **Addresses**.
3. Click **Import Configuration File** and upload your .zip file.
4. Review the imported entries, then click **Save Changes**.

Addressing

In most cases, you do not need to manually configure OPC Item Paths, as they are provided through device browsing or file import.

The only exception is when global variables are manually added on the Addresses page (not recommended). In which case, keep in mind the following:

- Paths must match the global variable names defined in Connected Components Workbench.
- Paths must resolve to an atomic data type (or an array of atomic data types). That means you must drill down through any structures until you reach a non-structured value.

To manually expose the UDT as a node in the OPC Browser, the paths need to be specified on the Addresses page by Browse Name, Data Type, and Read/Write access level. Columns for Dimension and String Size are also available to use when needed. See the [Configuring Data Points](#) section for additional details.

Bit-Level Addressing

With bit-level addressing, you can target individual bits by appending a dot and the bit index to the OPC Item Path. This applies for all numerical data types. The bit index must be within range of the addressed variable's data type.

Examples of bit-level addressing include:

- **MyUInt.1**: The first bit of MyUInt.
- **MyByteArray[0].7**: The seventh bit of the 0 array element inside MyByteArray.
- **MyUdt.MyDint.11**: The eleventh bit of the member tag, MyDint, inside the UDT, MyUdtInstance.

Performance Considerations

Due to communication limitations in the Micro800 series, optimization is limited. 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, this could negatively impact the performance of the PLC.

Connecting to MicroLogix



INDUCTIVE
UNIVERSITY

Connecting to MicroLogix

[Watch the video](#)

This driver supports:

- MicroLogix 1100 and 1400 direct over Ethernet/IP
- MicroLogix 1100 and 1400 through a 1761-NET-ENI Ethernet interface
- MicroLogix 1100 and 1400 through a Spectrum Controls WebPort 500

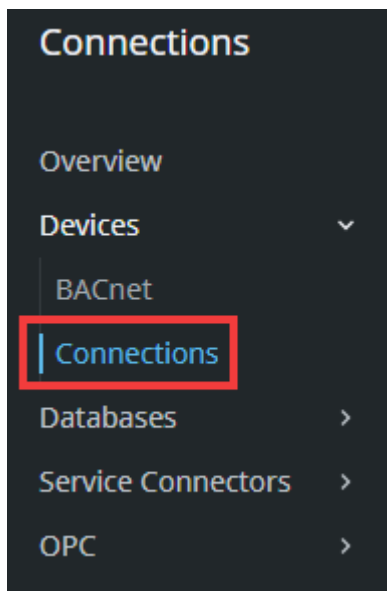
CAUTION

MicroLogix 1200 and 1500 are not fully supported. These devices do not support tag browsing, so **Disable Processor Browse** must be enabled. Additionally, the MicroLogix driver does not support access to Input Parameters.

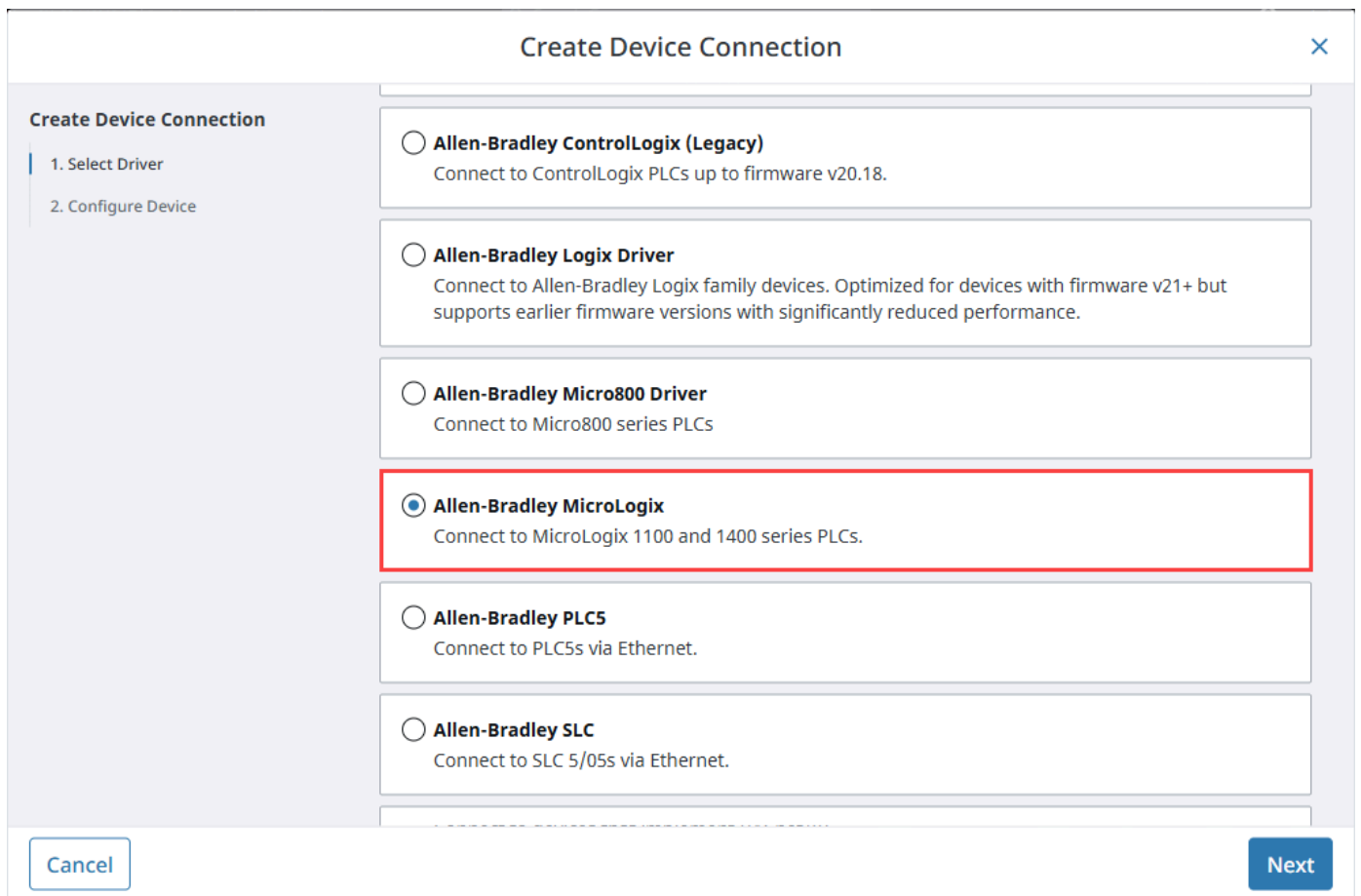
Connect to a MicroLogix Device

This driver requires the OPC UA module to be installed and enabled. If the module is missing or disabled, the device will fault with a “Missing Dependency” error on the **Platform System > Modules** page.

1. On the Gateway, go to **Connections > Devices > Connections**.



2. Click **Create Device Connection +**.
3. Select **Allen-Bradley MicroLogix** and click **Next**.



4. On the **Configure Device** screen, enter the required fields:
 - **Name:** MicroLogix
 - **Hostname:** The IP address of the device (e.g., 10.20.7.77)

Create Device Connection

Create Device Connection

1. Select Driver

2. Configure Device

Configure Device

MAIN

*Required

Name *

MicroLogix

The name of this device connection.

Description

Enabled

☒ Set whether this device connection is enabled or disabled within the Gateway.

CONNECTIVITY

*Required

Hostname *

10.20.7.77

Local Address

Address of network adapter to connect from.

Timeout *

2000

Connection Path

Show Advanced +

Back

Create Device Connection

5. Click **Create Device Connection**.

The device will be listed now on the Connections page. The status will initially show Disconnected, but will move to a Connected state.

Device Connections

Connections

Filter device connections...	View	20	Create Device Connection +		
☐	NAME	DRIVER	ENABLED	STATUS	
☐	MicroLogix	Allen-Bradley MicroLogix	true	CONNECTED	

Showing 1 of 1 items

Device Connection Settings

General

Name	Description
Name	A user-defined name for the device. This name appears in OPC item paths and the Devices list. Name must begin with an underscore or a letter, and cannot include special characters.
Description	Optional description to help identify the device.
Enabled	When selected, the device is active and available for use.

Connectivity

Name	Description
Hostname	The Hostname value is the IP Address of the MicroLogix 1100 processor, MicroLogix 1400 processor or 1761-NET-ENI Ethernet interface. Ethernet/IP protocol on TCP port 44818 (0xAF12) is used to communicate to the listed devices.
Local Address	The local address to connect from when establishing a TCP connection. If left blank, then the driver will select an available address.
Timeout	The maximum amount of time to wait, in milliseconds, for responses from the processor.
Connection Path	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. [Optional] Note: If a MicroLogix device becomes stuck in a Browse Pending state, try setting this property to 1,0 to force routing through backplane slot 0.

Advanced

Name	Description
Disable Processor Browse	Disables processor browse setting. May be required for some MicroLogix 1200 or 1500 devices.
Zero TNS Connection	Disables the Zero TNS connection setting.
Browse Cache Timeout	The amount of time, in milliseconds, to cache tag structure results read from the device.

Connecting to PLC5

NOTE

This driver supports PLC-5 processors using either Ethernet/IP or CSP over TCP/IP.



INDUCTIVE
UNIVERSITY

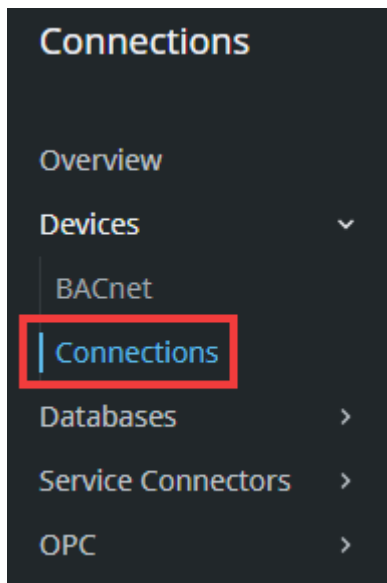
Connecting to PLC5

[Watch the video](#)

Connect to a PLC-5 Device

This driver requires the OPC UA module to be installed and enabled. If the module is missing or disabled, the device will fault with a “Missing Dependency” error on the **Platform System > Modules** page.

1. On the Gateway, go to **Connections > Devices > Connections**.



2. Click **Create Device Connection +**.
3. Select **Allen-Bradley PLC5** and click **Next**.

Create Device Connection

Create Device Connection

1. Select Driver

2. Configure Device

☐ **Allen-Bradley Micro800 Driver**
Connect to Micro800 series PLCs

☐ **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.

☐ **BACnet/IP**
Connect to devices that implement BACnet/IP.

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

☐ **DNP3 Driver (Legacy)**

Cancel

Next

4. On the **Configure Device** screen, enter the required fields:
- **Name:** PLC5
 - **Hostname:** The IP address of the PLC5 (e.g., 10.20.4.56)

Create Device Connection

Create Device Connection

1. Select Driver
2. Configure Device

Configure Device

MAIN

*Required

Name *

PLC5

The name of this device connection.

Description

Enabled
☒ Set whether this device connection is enabled or disabled within the Gateway.

CONNECTIVITY

*Required

Hostname *

10.20.4.56

Local Address

Address of network adapter to connect from.

Timeout *

2000

Connection Path

Show Advanced +

Back

Create Device Connection

5. Click **Create Device Connection**.

The device will be listed now on the Connections page. The status will initially show Disconnected, but will move to a Connected state.

Device Connections

Connections

Filter device connections... View 20 Create Device Connection +				
☐	NAME	DRIVER	ENABLED	STATUS
☐	PLC5	Allen-Bradley MicroLogix	true	DISCONNECTED: DETERMINING PROTOCOL

Showing 1 of 1 items

Device Connection Settings

General

Name	Description
Name	A user-defined name for the device. This name appears in OPC item paths and the Devices list. Name must begin with an underscore or a letter, and cannot include special characters.
Description	Optional description to help identify the device.
Enabled	When selected, the device is active and available for use.

Connectivity

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	The local address to connect from when establishing a TCP connection. If left blank, then the driver will select an available address.
Timeout	The maximum amount of time to wait, in milliseconds, for responses from the processor.
Connection Path	The Connection Path value is used to define the route of the PLC-5 processor to connect to. [Optional] See Connection Path Format for more details.

Advanced

Name	Description
Disable Processor Browse	Disables processor browse setting.
Zero TNS Connection	Disables the Zero TNS connection setting.
Browse Cache Timeout	The amount of time, in milliseconds, to cache tag structure results read from the device.

Connection Path Format

The **Connection Path** is used when routing to a PLC5 via ControlLogix hardware, such as a 1756-DHRIO module. Use the following format:

1,<DHRIO slot number>,<DHRIO channel>,<DH+ node number>

- **1**: Instructs the 1756-ENET module to move to the backplane
- **DHRIO slot number**: Slot of the 1756-DHRIO module (0–16 depending on the chassis size)
- **DHRIO channel**: Use 2 for Channel A, or 3 for Channel B
- **DH+ node number**: The node number configured in the PLC-5 (octal, range 00–77)

Supported PLC5 Connection Methods

- PLC-5 L/20E, L/40E, and L/80E via Ethernet/IP
- Any PLC-5 processor connected over DH+ through a 1756-DHRIO module

 NOTE

ASCII data types from PLC-5 devices are not supported by the PLC5 driver.

Connecting to SLC

NOTE

This driver supports SLC 5/05 and compatible SLC processors using CSP or Ethernet/IP protocols.



INDUCTIVE
UNIVERSITY

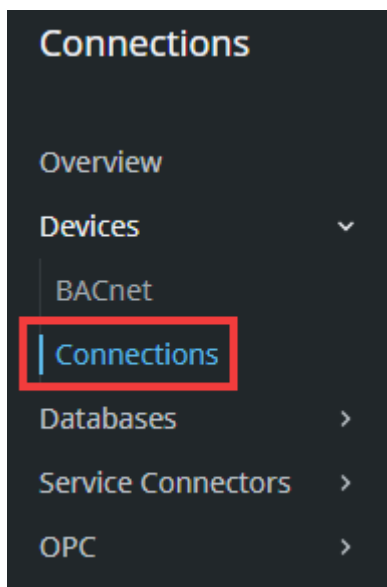
Connecting to SLC

[Watch the video](#)

Connect to an SLC Device

This driver requires the OPC UA module to be installed and enabled. If the module is missing or disabled, the device will fault with a “Missing Dependency” error on the **Platform System > Modules** page.

1. On the Gateway, go to **Connections > Devices > Connections**.



2. Click **Create Device Connection +**.
3. Select **Allen-Bradley SLC** and click **Next**.

NOTE

SLC PLCs do not have built-in Ethernet. Use a 1761-NET-ENI, ENBT, or similar device to enable communication.

Create Device Connection

Create Device Connection

1. Select Driver

2. Configure Device

☐ **Allen-Bradley Micro800 Driver**
Connect to Micro800 series PLCs

☐ **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.

☐ **BACnet/IP**
Connect to devices that implement BACnet/IP.

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

☐ **DNP3 Driver (Legacy)**

Cancel

Next

4. On the **Configure Device** screen, enter the required fields:

- **Name:** SLC
- **Hostname:** IP address of the device (e.g., 10.20.4.56)

Create Device Connection

Create Device Connection

1. Select Driver

2. Configure Device

Configure Device

MAIN

*Required

Name *

SLC

The name of this device connection.

Description

Enabled

☒ Set whether this device connection is enabled or disabled within the Gateway.

CONNECTIVITY

*Required

Hostname *

10.20.4.56

Local Address

Address of network adapter to connect from.

Timeout *

2000

Connection Path

Show Advanced +

Back

Create Device Connection

5. Click **Create Device Connection**.

The device will be listed now on the Connections page. The status will initially show Disconnected, but will move to a Connected state.

Device Connections

Connections

Filter device connections...	View	20	Create Device Connection +		
☐	NAME	DRIVER	ENABLED	STATUS	
☐	SLC	Allen-Bradley PLC5	true	CONNECTED: PROTOCOL: DIRECT	

Showing 1 of 1 items

Device Connection Settings

Main Settings

Name	Description
Name	A user-defined name for the device. This name appears in OPC item paths and the Devices list. Name must begin with an underscore or a letter, and cannot include special characters.
Description	Optional description to help identify the device.
Enabled	When selected, the device is active and 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	The local address to connect from when establishing a TCP connection. If left blank, then the driver will select an available address.
Timeout	The maximum amount of time to wait, in milliseconds, for responses from the processor.
Connection Path	The Connection Path value is used to define the route of the PLC-5 processor to connect to. [Optional] See Connection Path Format for more details.

Advanced Settings

Name	Description
Disable Processor Browse	Disables processor browse setting.
Zero TNS Connection	Disables the Zero TNS connection setting.
Browse Cache Timeout	The amount of time, in milliseconds, to cache tag structure results read from the device.

Connection Path Format

If the SLC is connected over DH+ through a 1756-DHRIO module, use the following format for **Connection Path**:

1,<DHRIO slot number>,<DHRIO channel>,<DH+ node number>

- **1**: Indicates a transition to the ControlLogix backplane
- **DHRIO slot number**: Slot number of the 1756-DHRIO module (0–16 depending on chassis)
- **DHRIO channel**: Use 2 for Channel A or 3 for Channel B
- **DH+ node number**: Octal value of the SLC node, matching its configuration

Supported SLC Connection Methods

- SLC 5/05 direct over Ethernet
- SLC 5/03, 5/04, 5/05 via 1761-NET-ENI
- SLC 5/04 via 1756-DHRIO
- SLC 5/03, 5/04, 5/05 via 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 backplane.
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 backplane (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



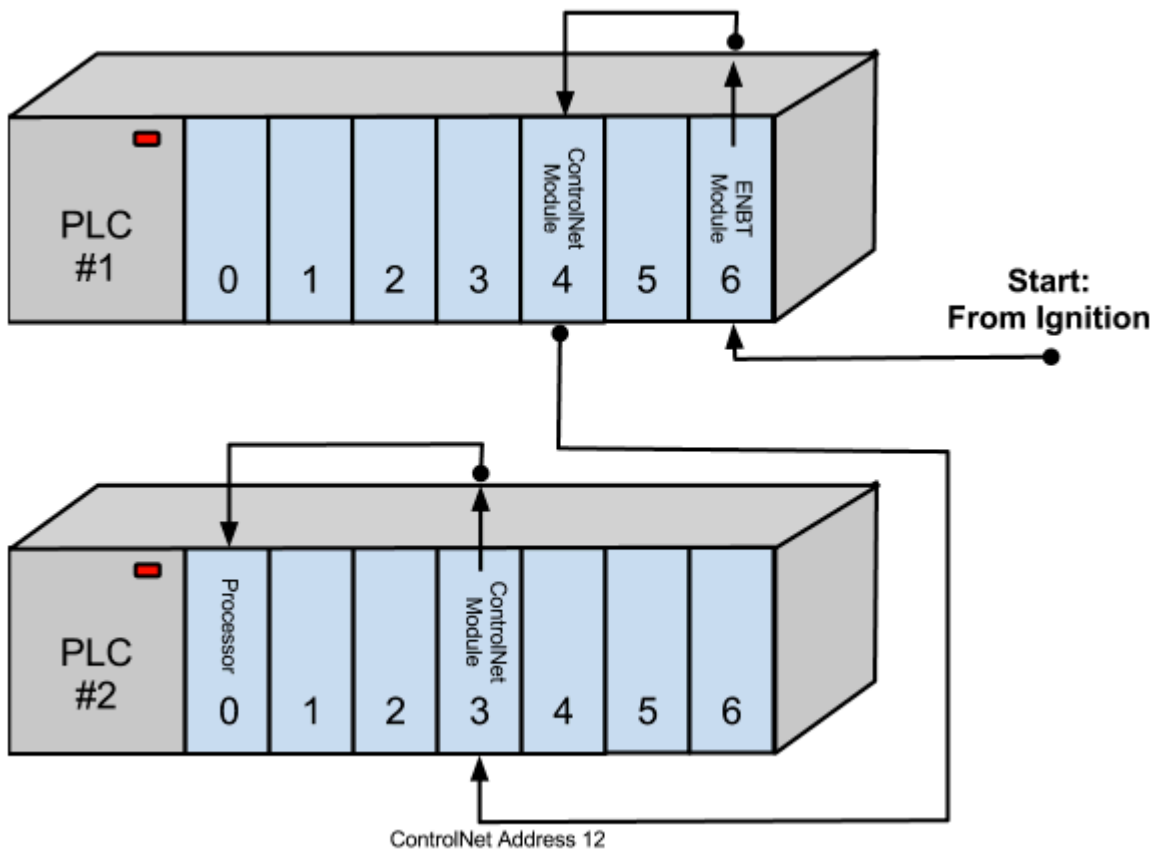
INDUCTIVE
UNIVERSITY

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.



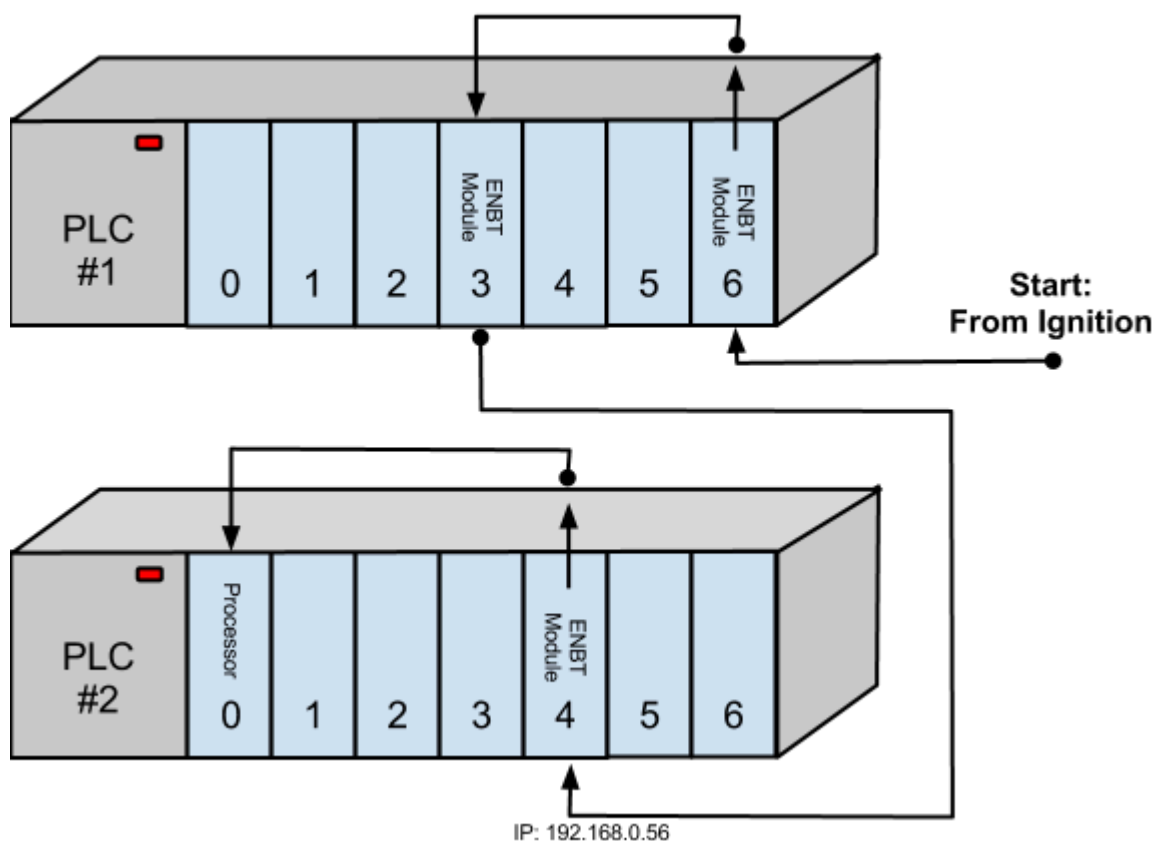
Connecting to a PLC Using ENBT

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).

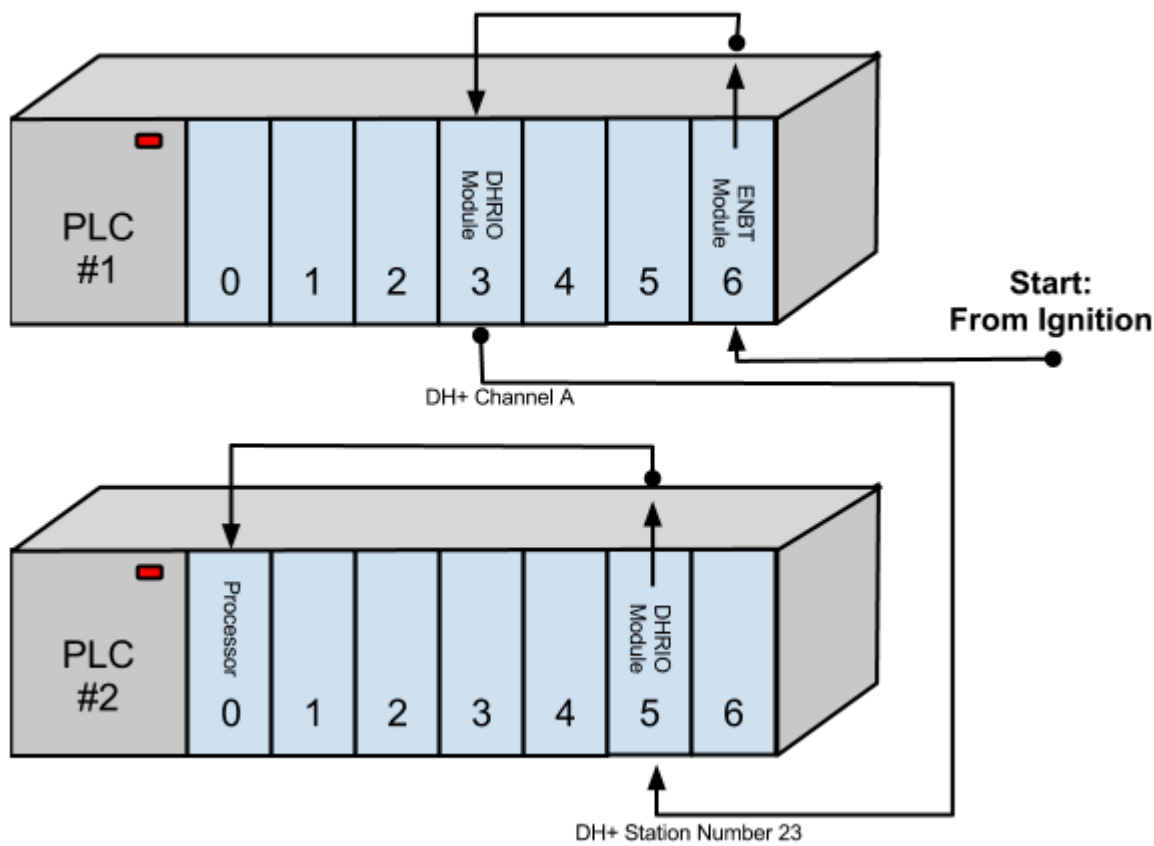
Number/Entry	Description
1	Move out of ENBT Module to the backplane.
3	Move to ENBT Module in Slot 3.
2	Move out the Ethernet Port.
192.168.0.56	Move in EBNT Module at IP Address 192.168.0.56.
1	Move out of EBNT to the backplane.
0	Move to the Processor in Slot 0.



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+)



INDUCTIVE
UNIVERSITY

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

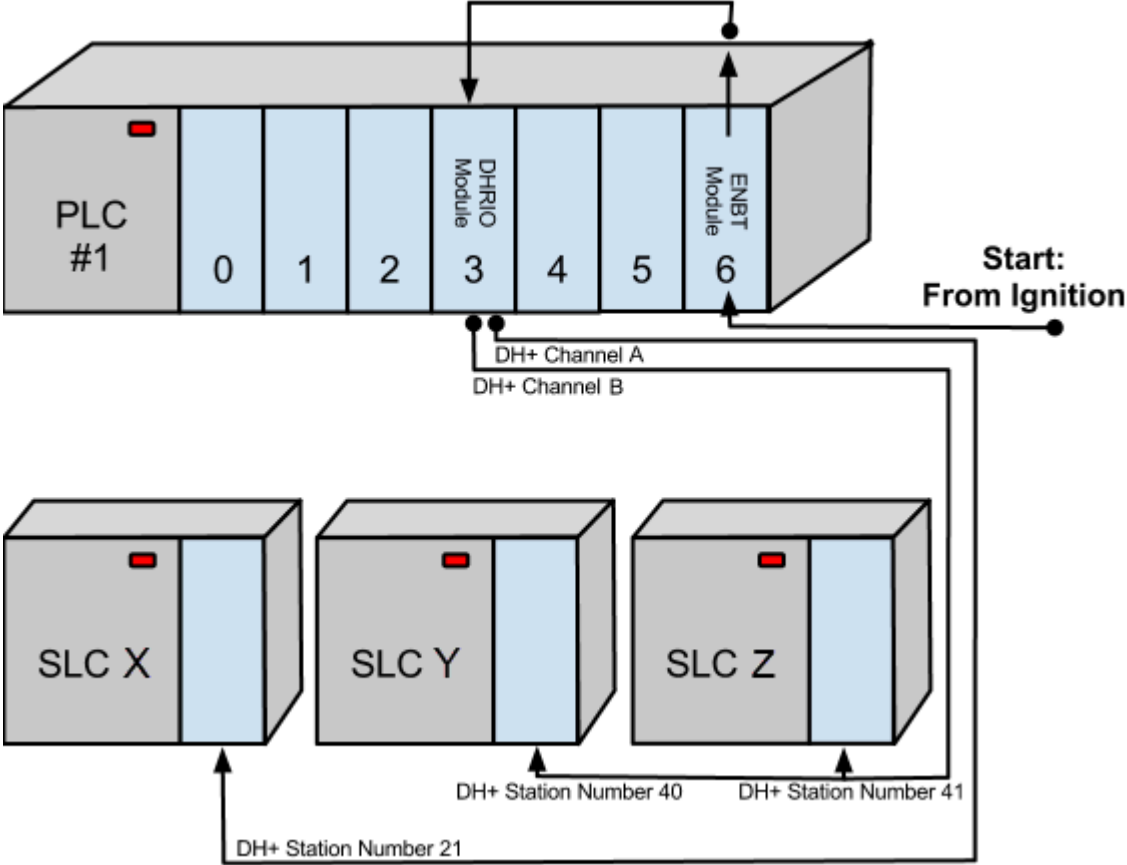
Number/Entry	Description
1	Move out of ENBT Module to the backplane.
3	Move to DHRIO Module in Slot 3.
2	Move out DH+ Channel A.
21	Move in DH+ Station Number 23.

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 Modbus Device

CAUTION

This driver requires the OPC UA module to be installed and enabled. If the module is missing or disabled, the device will fault with a “Missing Dependency” error on the **Platform > System > Modules** page.

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

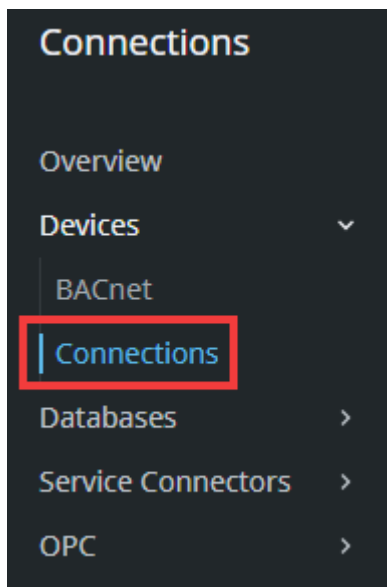
Connecting to Modbus Device

[Watch the video](#)

Connect to a Modbus Device

This driver requires the OPC UA module to be installed and enabled. If the module is missing or disabled, the device will fault with a “Missing Dependency” error on the **Platform System > Modules** page.

1. On the Gateway, go to **Connections > Devices > Connections**.



2. Click **Create Device Connection +**.
3. Choose a Modbus driver from the list and click **Next**:
 - **Modbus TCP** for devices that support native Modbus TCP/IP.

Create Device Connection

Create Device Connection

1. Select Driver

2. Configure Device

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

☒ **Modbus TCP**

Connect to devices that implement the Modbus TCP protocol.

☐ **OPC-COM Tunnel Driver**

Adds a new type of device that exposes configured OPC-COM connections through the OPC-UA server.

☐ **Omron FINS/TCP**

Connect to devices that implement FINS over TCP.

☐ **Omron FINS/UDP**

Connect to devices that implement FINS over UDP.

☐ **Omron NJ Driver**

Connect to Omron NJ series PLCs.

☐ **Programmable Device Simulator**

A simulator device that can be configured with a user-defined hierarchy of static or function-driven values.

Cancel

Next

- **Modbus RTU over TCP** for devices that encapsulate RTU messages inside TCP packets.

Create Device Connection

Create Device Connection

1. Select Driver

2. Configure Device

☐ **Modbus RTU**

Connect to devices that implement the Modbus RTU protocol over a serial link.

☒ **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.

☐ **OPC-COM Tunnel Driver**

Adds a new type of device that exposes configured OPC-COM connections through the OPC-UA server.

☐ **Omron FINS/TCP**

Connect to devices that implement FINS over TCP.

☐ **Omron FINS/UDP**

Connect to devices that implement FINS over UDP.

☐ **Omron NJ Driver**

Connect to Omron NJ series PLCs.

Cancel

Next

- **Modbus RTU** for serial RTU communication.

Create Device Connection

1. Select Driver
2. Configure Device

- ☐ **Mitsubishi TCP**
Connect to Mitsubishi devices supporting MELSEC over TCP.
- ☒ **Modbus RTU**
Connect to devices that implement the Modbus RTU protocol over a serial link.
- ☐ **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.
- ☐ **OPC-COM Tunnel Driver**
Adds a new type of device that exposes configured OPC-COM connections through the OPC-UA server.
- ☐ **Omron FINS/TCP**
Connect to devices that implement FINS over TCP.
- ☐ **Omron FINS/UDP**

Cancel Next

4. On the **Configure Device** screen, enter the required fields:

- **Name:** Modbus
- **Hostname** (TCP and RTU over TCP): IP address of the device (e.g., 10.20.8.117)
- **Serial Port** (RTU): Serial port path (e.g., COM1)

Create Device Connection

Create Device Connection

1. Select Driver

2. Configure Device

GENERAL

*Required

Name *

Modbus

The name of this device connection.

Description

Enabled

☒ Set whether this device connection is enabled or disabled within the Gateway.

CONNECTIVITY

*Required

Hostname *

10.20.8.117

Hostname/IP address of the Modbus device.

Port *

502

Port to connect to.

Local Address

Address of network adapter to connect from.

Communication Timeout *

2000

Maximum amount of time to wait for a response.

Back

Create Device Connection

5. Click **Create Device Connection**.

The device will be listed now on the Connections page. The status will initially show Disconnected, but will move to a Connected state.

Device Connections

Connections

Filter device connections...

View

20

Create Device Connection +

☐	NAME	DRIVER	ENABLED	STATUS	
☐	Modbus	Modbus TCP	true	CONNECTED	

Showing 1 of 1 items

NOTE

Modbus devices do not support tag browsing. To access tags, you must manually define addresses in the Tag Browser or configure mappings using the [Modbus Addressing](#) and [Modbus Address Mapping](#) pages.

Modbus Protocol Support

The Modbus driver can connect to any device that supports the Modbus protocol via Ethernet or serial. Only one Modbus device should be added per IP address in the Ignition Device List. For Modbus Gateways that route to multiple devices, use separate unit IDs per device, either in the tag path or via the address mapping page.

Supported Function Codes

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

Device Connection Settings

General

Name	Description
Name	A user-defined name for the device. This name appears in OPC item paths and the Devices list. Name must begin with an underscore or a letter, and cannot include special characters.
Description	Optional description to help identify the device.
Enabled	When selected, the device is active and available for use.

Connectivity – Modbus TCP / RTU over TCP

Name	Description
Hostname	Specifies the IP address of the Modbus device or TCP Gateway.
Port	Defines the TCP port used by the device. The default is 502.
Local Address	Specifies the IP address of the local adapter to bind from. Leave this blank to allow automatic selection.
Communication Timeout	Sets the time, in milliseconds, to wait for a response before the request is considered failed.

Connectivity – Modbus RTU

Name	Description
Serial Port	Specifies the name of the serial port (for example, COM1).
Bit Rate	Sets the bit rate used for the serial connection.
Data Bits	Defines the number of data bits in the message frame.
Parity	Sets the parity configuration for serial communication.
Stop Bits	Defines the number of stop bits used in the frame.
Handshake	Specifies the flow control method for the serial connection.
Communication Timeout	Sets the time, in milliseconds, to wait for a response before the request is considered failed.
RS-485 Mode	Enables RS-485 mode instead of the default RS-232.

Request Optimization

Name	Description
Max Holding Registers per Read Request	Changed in 8.3.1 Specifies the maximum number of holding registers allowed per read request. The default is 125.
Max Holding Registers Per Write Request	New in 8.3.1 Specifies the maximum number of holding registers allowed per write request. The default is 123.
Max Input Registers per Request	Specifies the maximum number of input registers to read per request. The default is 125.
Max Coils per Request	Sets the maximum number of coils to read in a single request. The default is 2000.
Max Discrete Inputs per Request	Sets the maximum number of discrete inputs to read per request. The default is 2000.
Concurrent Requests	Defines the number of simultaneous requests the driver can issue. Not available for Modbus RTU.
Span Gaps	Specifies whether to span gaps in the address space to reduce the total number of requests.

Write Request

Name	Description
Allow Write Multiple Registers Request	Enables the use of function code 0x10 for writing multiple registers. Disable this option only if the device does not support it.
Force Multiple Register Writes	Forces the driver to use function code 0x10 for all multi-word register writes.
Allow Write Multiple Coils Request	Enables the use of function code 0x0F for writing multiple coils. Disable this option only if the device does not support it.

Read Request

Name	Description
Allow Read Multiple Registers Request	Enables reading registers in blocks instead of individually. Disable this option to read each register separately.
Allow Read Multiple Coils	Enables reading coils in blocks instead of individually. Disable this option to read each coil separately.
Allow Read Multiple Discrete Inputs	Enables reading discrete inputs in blocks instead of individually. Disable this option to read each input separately.

Advanced

Name	Description
Reconnect After Consecutive Timeouts	Forces the driver to reconnect after three consecutive timeouts. The default setting is true.
Reverse Word Order	Swaps the low and high words in 32-bit values. Enable this option if the data appears reversed.
Zero-based Addressing	Enables zero-based addressing. This is required by some Modbus implementations.
Max Retry Count	Sets the number of times the driver will retry a request after receiving a device error. The default is 1.

String Handling

Name	Description
Reverse String Byte Order	Reverses the byte order when reading string values.
Right Justify Strings	Pads string values to the right instead of the default left padding.
Read Raw Strings	Reads the full configured string length, including any null characters.

Modbus Addressing



INDUCTIVE
UNIVERSITY

About Modbus Addressing

[Watch the video](#)

Manually Addressing a Modbus Device

Modbus does not support tag browsing, meaning you cannot browse Modbus addresses in the **Connected Devices window** in the Designer or through the Gateway using the **OPC Quick Client**. However, once Modbus tags have been created, the OPC Quick Client can be used to read and verify their values.

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.

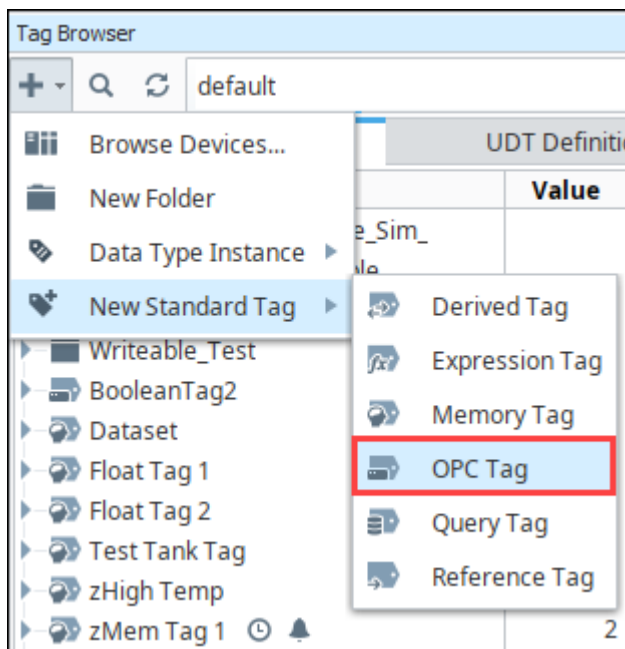
2. By specifying the address mapping

- 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 blocks of common addresses to be entered for the driver to then create the individual addresses. These individual addresses will be displayed in the **Connected Devices window** or through the Gateway using the **OPC Quick Client**

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. The 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.

Radix *

10

The base number of unique digits for modbus addresses. Determines what format addresses in the device are incremented and labeled.

Address Mapping

Import

Export

Add

PREFIX	START	END	STEP	UNIT ID	MODBUS TYPE	MODBUS ADDRESS	
Temp	1	10	☐	0	Holding ...	10	Delete

Showing 1 of 1 items

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 **Connections** section of the Gateway Webpage.
2. Select **Devices > Connections**.
3. Click on the **three dots** menu and select **Addresses** to the right of your Modbus device.

Device Connections

Connections

View 20 ▾ Create Device Connection +

☐	NAME ↑	DRIVER	ENABLED	STATUS	⚙
☐	FINSUDP	Omron FINS/UDP	true	BOUND	⋮
☐	modbus rtu	Modbus RTU over TCP	true	❗ DISCONNECTED	⋮

Showing 2 of 2 items

Edit

Rename

Duplicate

Addresses

View Details

Roles & Permissions

Disable

4. Click on **Add**.

Radix *

10

The base number of unique digits for modbus addresses. Determines what format addresses in the device are incremented and labeled.

Address Mapping

Import Export **Add +**

PREFIX	START	END	STEP	UNIT ID	MODBUS TYPE	MODBUS ADDRESS
--------	-------	-----	------	---------	-------------	----------------

Showing 0 of 0 items

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)

Radix *

8

The base number of unique digits for modbus addresses. Determines what format addresses in the device are incremented and labeled.

Address Mapping

Import

Export

Add +

PREFIX	START	END	STEP	UNIT ID	MODBUS TYPE	MODBUS ADDRESS	
V	2000	3777	☐	0	Holding ...	1024	Delete

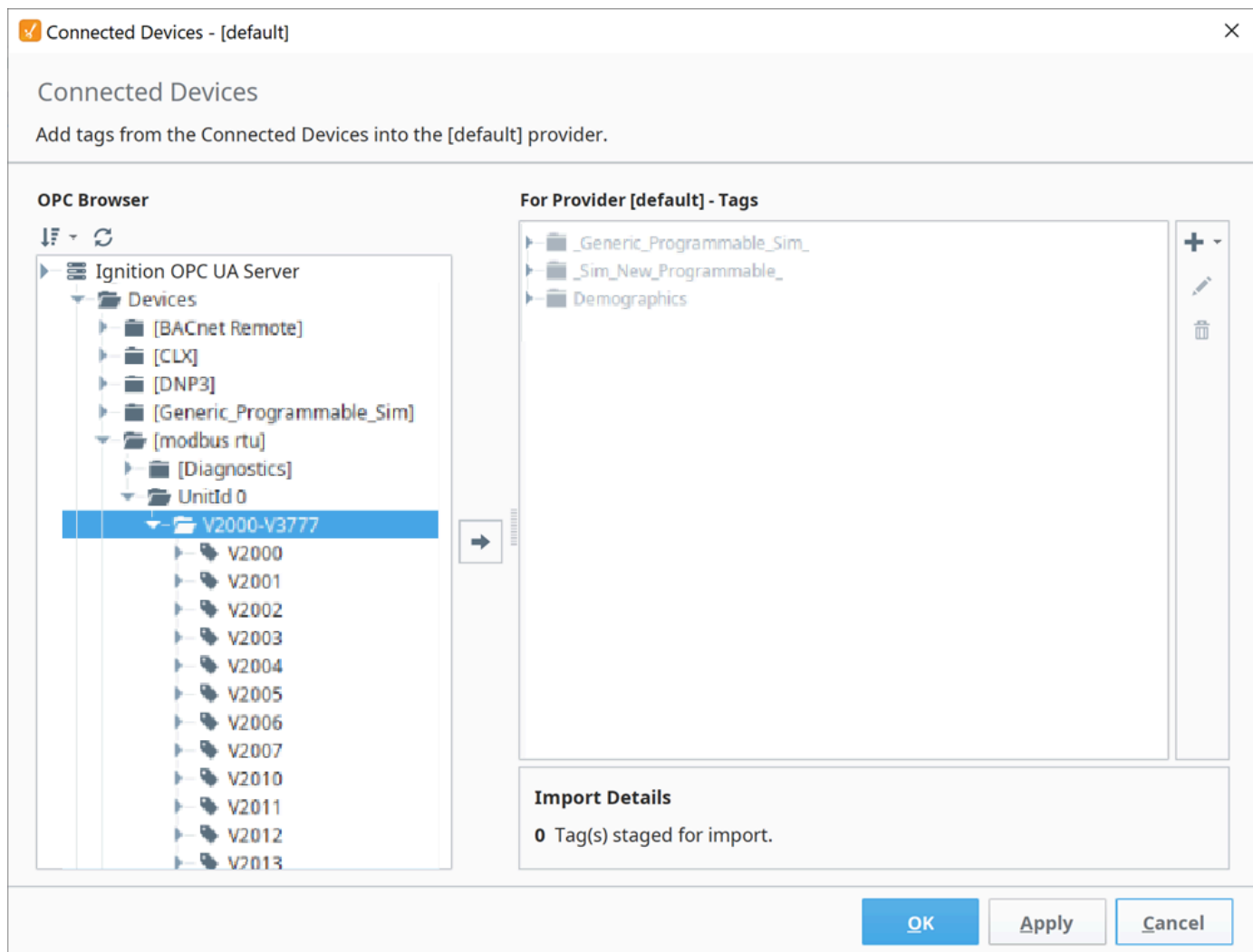
Showing 1 of 1 items

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.

6. Click **Save Changes**.
7. Go to the **Connected Devices** window in the Designer or the **Connections > OPC > Quick Client** page on the 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

Radix *

8

The base number of unique digits for modbus addresses. Determines what format addresses in the device are incremented and labeled.

Address Mapping

Import

Export

Add +

PREFIX	START	END	STEP	UNIT ID	MODBUS TYPE	MODBUS ADDRESS	
V	2000	3777	☐	0	Holding ...	1024	Delete
X	0	477	☐	0	Discrete...	2048	Delete
Y	0	477	☐	0	Coil	2048	Delete
TAO	0	127	☐	0	Input Re...	0	Delete

Showing 4 of 4 items

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

Radix *

8

The base number of unique digits for modbus addresses. Determines what format addresses in the device are incremented and labeled.

Address Mapping

Import

Export

Add

PREFIX	START	END	STEP	UNIT ID	MODBUS TYPE	MODBUS ADDRESS	
V	2000	3777	☐	0	Holding ...	1024	Delete
V	2000	3777	☐	1	Holding ...	2024	Delete

Showing 2 of 2 items

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.

Radix *

10

The base number of unique digits for modbus addresses. Determines what format addresses in the device are incremented and labeled.

Address Mapping

Import

Export

Add

PREFIX	START	END	STEP	UNIT ID	MODBUS TYPE	MODBUS ADDRESS	
R	1024	1030	☒	0	Holding ...	1024	Delete

Showing 1 of 1 items

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

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. The available S7 device drivers are listed below by the module providing them:

Siemens Driver Module

- Siemens S7-300 Driver: Connects to S7-300 PLCs, supports absolute addressing only.
- Siemens S7-400 Driver: Connects to S7-400 PLCs, supports absolute addressing only.
- Siemens S7-1200 Driver: Connects to S7-1200 PLCs, supports absolute addressing only.
- Siemens S7-1500 Driver: Connects to S7-1500 PLCs, supports absolute addressing only.

Siemens Enhanced Driver Module

- Siemens Enhanced Driver: Connects to S7-300, S7-400, S7-1200, and S7-1500 devices. Supports browsing, symbolic access, and access to optimized data blocks when connected to S7-1200 and S7-1500 devices.

Deciding what driver is best for you depends on a few factors, such as your licensing requirements and setup since the Siemens Enhanced Driver module is not included on a Platform license. Additional considerations include your operating system and hardware. The Siemens Driver module is supported on Windows, Linux, Mac, and ARM. However, the Siemens Enhanced Driver module is only supported for Windows, Linux/x64, and Linux/ARM, but not Mac.

Considerations for 1200 and 1500 Devices

The requirements listed below apply when connecting to the S7-1200 and S7-1500 types, through either the Siemens drivers or the Siemens Enhanced driver options. The drivers are split into their own sections as the requirements will differ when using Symbolic addressing with the Siemens Enhanced driver.

Siemens Driver

The following list is applicable to the S7-1200 and S7-1500 driver connections available through the Siemens Driver module.

- Only global DBs can be accessed.

- [illegible]

Additionally, 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, instead of addressing each tag manually.

Siemens Enhanced Driver

The following requirements are applicable to the Siemens Enhanced driver when used to connect to an S7-1200 or S7-1500 device.

- [illegible]

Refer to the [Siemens Enhanced Driver](#) and [Siemens Drivers](#) pages for more information on driver configuration and connection. The Siemens Drivers page covers addressing and connection settings for the S7-300, S7-400, S7-1200, and S7-1500 drivers.

Siemens Drivers

The following Siemens drivers in Ignition connect to PLCs via TCP/IP using the S7 protocol. Similar to Modbus and some Allen Bradley connections, the following 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.

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

Connecting to a Device

The Siemens Drivers 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 **Platform > System > Modules** page.



INDUCTIVE
UNIVERSITY

Connecting to S7 Devices

[Watch the video](#)

1. Go to the **Connections** section of the Gateway Webpage.
2. Select **Devices > Connections**.
3. On the Devices page, click on **Create Device Connection +**.
4. All four of the available driver options are shown on the Select Driver form. For this example we will select **Siemens S7-1200 Driver**.
 - **Siemens S7-1500 Driver:** Connects to Siemens S7-1500 PLCs over Ethernet.
 - **Siemens S7-1200 Driver:** Connects to Siemens S7-1200 PLCs over Ethernet.
 - **Siemens S7-300 Driver:** Connects to Siemens S7-300 PLCs over Ethernet.
 - **Siemens S7-400 Driver:** Connects to Siemens S7-400 PLCs over Ethernet.

5. Click **Next**.

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 +** to see the additional settings, but you can keep all the defaults.

8. Click **Create Device Connection**. 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.

Configuring Siemens Addressing



INDUCTIVE
UNIVERSITY

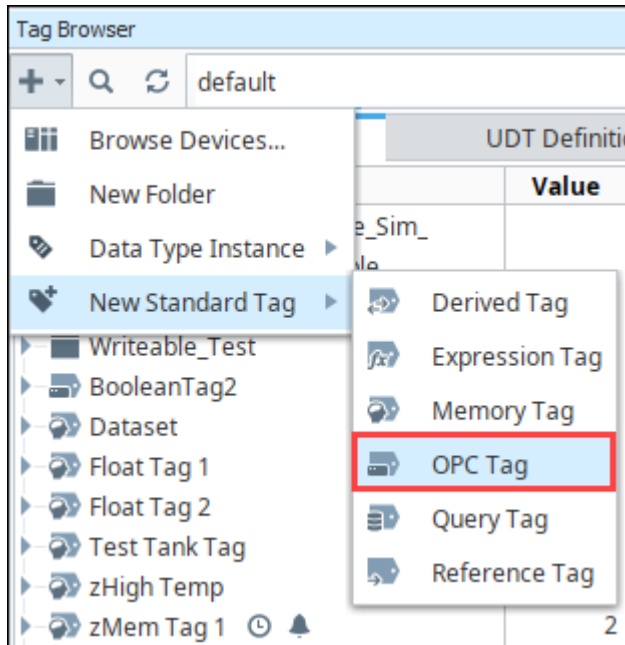
About Siemens Addressing

[Watch the video](#)

To see and browse the tags, you need to create the tags manually or by importing bulk using the tags CSV import functionality. Both options are completed in the Designer.

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. Refer to the following list for more information on data types and offsets to use when forming addresses.

- **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.

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	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	LI Note: Due to limitations on Siemens devices, this data type is only supported on the following device: S7-1500	Signed
LReal	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).

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.
Port	The port to use when connecting to the device. The default is 102.
Local Address	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
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.

Siemens Enhanced Driver

USING THE SIEMENS ENHANCED DRIVER WITH IGNITION 8.3.3+

The AGLink library has been updated to support Siemens S7-1200 G2 PLCs. With this update, the Siemens Enhanced Driver module also now requires a GLIBC version of 2.34 (or newer) on x64 platforms and 2.17 for ARM platforms. This prevents the Siemens Enhanced Driver module from running on the following distros with an x64 architecture: Ubuntu 20.04, Debian 10, and Debian 11.

The Siemens Enhanced driver supports browsing, symbolic addressing, and access to optimized data blocks when S7-1200 and S7-1500 device types are selected in the connection settings. Symbolic addressing is not supported for S7-300 and S7-400 devices.

CAUTION

There is a known issue for Siemens S7-1500 PLCs running versions 4.0.0 or 4.0.4 that can cause the Siemens Enhanced Driver to lose connection to the PLC. This has been resolved in version 4.1.2 of the PLC firmware. If you are running into this issue, try updating your PLC firmware.

Note that the Siemens driver has a device connection limit of 250 devices.

REQUIRED SYMLINKS

If you are installing Ignition on a clean Linux environment and you want to use secure connections when connecting to a PLC, you will need to create symlinks for your specific OpenSSL version. You can create the symlinks by installing the package `libssl-dev`, or you can manually create the symlinks with `ln`. Two symlinks are necessary. One symbolic link named `libssl.so` that links to your specific libssl library and another named `libcrypto.so` that links to your specific `libcrypto` library. Since these libraries are loaded by a native DLL, these symlinks must exist within the `LD_LIBRARY_PATH`. If you are using OpenSSL 3.0 on a 64-bit Ubuntu installation, you might create the symlinks by running:

```
~# cd /usr/lib/x86_64-linux-gnu
/usr/lib/x86_64-linux-gnu# ln -s libssl.so.3 libssl.so
/usr/lib/x86_64-linux-gnu# ln -s libcrypto.so.3 libcrypto.so
```

These symlinks are provided automatically in the Docker images.

Connecting to a Device

The Siemens Enhanced Driver 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 **Platform > System > Modules** page.

1. Go to the **Connections** section of the Gateway Webpage.
2. Select **Devices > Connections**.
3. On the Devices page, click on **Create Device Connection +**.
4. On the Select Driver form, select **Siemens Enhanced Driver** and click **Next**.
5. On the Configure Device form, leave all the default values and type in the following fields:
 - Name: **S7_Symbolic**
 - Hostname: Enter the IP address, for example 10.20.4.71.
6. Under **Device Settings**, select your Device Type.
 - **S7-1500**
 - **S7-1200**
 - **S7-300**
 - **S7-400**
7. Select Symbolic or Absolute for the **Address Type**.

S7-300 AND S7-400 ADDRESSING

If your Device Type is either a S7-300 or S7-400, you must select Absolute as your Address Type to establish a connection.

8. Click **Create Device Connection**.

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.

Addressing Syntax

The Siemens Enhanced driver supports two addressing types: Absolute and Symbolic. Absolute addressing accepts the same syntax as the [Siemens drivers](#), as well as a variation that resembles the syntax used in the TIA portal, with the addition of supporting arrays. This variation is described in the [Absolute Addressing](#) section. If you've connected with the Symbolic address type, refer to the [Symbolic Addressing](#) section.

Absolute Addressing

Data Type	Syntax Address
Numeric - Inputs, Outputs, Flags	<Area><DataType><Offset>[ArrayDimensions]
Numeric - Data Blocks	DB<DBNumber>.[DB]<DataType><Offset> [ArrayDimensions]
Boolean/Bit - Inputs, Outputs, Flags	<Area>X<Offset>[.BitNumber][ArrayDimensions]
Boolean/Bit - Data Blocks	DB<DBNumber>.[DB]X<Offset>[.BitNumber] [ArrayDimensions]
String - Inputs, Outputs, Flags	<Area>STRING<Offset>[Length]
String - Data Blocks	DB<DBNumber>.[DB]STRING<Offset>[Length]
Timer	T<Offset>
Counter	C<Offset>

Manual Addressing Examples

Address	Description
IW0	16 bit unsigned integer (Word) from offset 0.
QI12	16 bit signed integer (Int) from offset 12.
MD5[2]	Array of two 32 bit unsigned integers (DWord) from offset 5.
IW0[3,2,2]	Multidimensional array (with dimensions [3,2,2]) of Words from offset 0.
DB500.DBREAL10	32-bit floating point number from offset 10.
DB400.DBDI52[4,8]	Array with 4 rows of 8 columns of 32-bit signed integer (32 Int32 values in total) from offset 52.
DB1.DBX9.7	Bit 7 from offset 9
DB1.X8	Bit 0 from offset 8
DB1.DBX8[8]	Array of 8 booleans, Bit 0 from offset 8.
DB500.STRING255[7]	String of up to 7 characters from offset 255.
T9	Timer from offset 9.
C4	Counter from offset 4.

Data Types

S7 Data Type	OPC UA Data Type
BOOL	Boolean
BYTE	Byte
WORD	UInt16
DWORD	UInt32
LWORD	UInt64
SINT	SByte
INT	Int16
DINT	Int32
LINT	Int64
USINT	Byte
UINT	UInt16
UDINT	UInt32
ULINT	UInt64
REAL	Float
LREAL	Double
S5TIME	UInt32 Note: Represents some number of milliseconds between 0 and 9990000.
TIME	Int32 Note: Represents some number of milliseconds between Int32.MIN and Int32.MAX.

S7 Data Type	OPC UA Data Type
LTIME	Int64 Note: Represents some number of nanoseconds between Int64.MIN and Int64.MAX.
DATE	UInt16 Note: Represents the number of days since 01-01-1990.
TIME_OF_DAY (TOD)	UInt32 Note: Represents the number of milliseconds since the beginning of the day.
LTOD (LTIME_OF_DAY)	UInt64 Note: Represents the number of nanoseconds since the beginning of the day.
DT (DATE_AND_TIME)	DateTime
LDT	UInt64 Note: Nanoseconds since epoch (01-01-1970 00:00:00.000000000).
DTL	N/A Note: Structured representation of a date and time.
CHAR	String
WCHAR	String
STRING	String
WSTRING	String

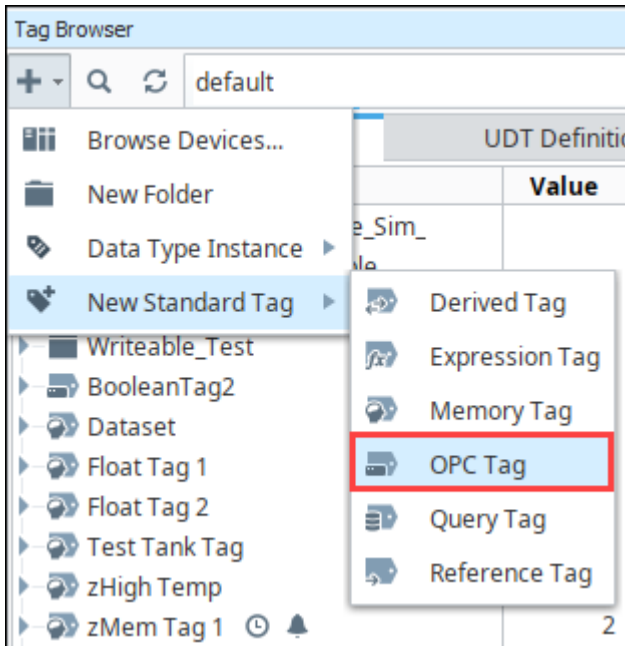
Symbolic Addressing

The Symbolic address type for S7-1200 and S7-1500 makes tag browsing possible in the Designer.

Changed in 8.3.10 Requests for the Siemens Enhanced driver are now limited to 100 items to improve performance. Multiple requests will be created when subscriptions are greater than 100

items, and the driver can handle multiple requests at the same time.

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, you will need to enter a Name, select a Data Type, and select the **Ignition OPC-UA Server** as your OPC Server.
4. For the OPC Item Path field, select the Devices icon to browse the OPC UA Server. Use the dropdowns to navigate to your desired tag path and click **Apply**. The OPC Item Path will now be filled.

5. Click **OK**. Now you can see your tag in the Tag Browser.

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.
Port	The port to use when connecting to the device. The default is 102.
Timeout	The maximum amount of time to wait for a response, specified in milliseconds. The default is 3,000.

Device Settings

Setting	Description
Device Type	The type of S7 model used for this driver connection. Options include S7-300, S7-400, S7-1200, and S7-1500.
Address Type	The type of addressing used for this driver connection. Options include Absolute or Symbolic for S7-1200 and S7-1500 devices. S7-300 and S7-400 devices can only use Absolute addressing. New in 8.3.7 If Symbolic is selected, an Export PLC Symbols option will be available in the device connection's three dots menu.
Rack Number	The number of the rack that the device is positioned in. The default is 0.
Slot Number	The slot number assigned to the CPU. The default is 0.

Advanced Settings

Setting	Description
Optimize Writes	When enabled, tag writes will be batched.
Secure Connection Mode	Changed in 8.3.4 Controls how secure connections (TLS) are handled. Options include: - REQUIRED: The device will only use a secure connection. - PREFERRED: The device will attempt to use a secure connection if available, but will use an unsecure connection if not. - DISABLED: The device will only use unsecure connections. In versions 8.3.3 and earlier, this property was called Force Secure Connection . Secure Connection Mode will be automatically set if upgrading from a previous version that used Force Secure Connection. - If Force Secure Connection was enabled, Secure Connection Mode will use REQUIRED. - If Force Secure Connection was disabled, Secure Connection Mode will use PREFERRED. Default is PREFERRED.
Password	The password selections available for this connection. Options include None, Embedded, and Referenced.

UDP and TCP Driver

Ignition's UDP and TCP drivers allow communication with simple devices such as barcode scanners and scales. These drivers are not general-purpose PLC drivers, but lightweight tools for raw stream data over TCP or UDP.

They connect to one or more ports on a given IP address and receive any incoming data (ASCII characters, for example) as tag values. Both drivers can act as passive listeners, meaning they do not write or make requests unless configured to do so. This is ideal for devices that continually push data without expecting a poll/response.

The TCP driver supports an optional writeback capability, exposing writable tags to send data back to the device.

NOTE

These drivers do not parse industrial protocols, but instead handle stream-oriented or delimited data only.

Structure in the Address Space

When you connect with these drivers, the device appears under the Devices folder of the OPC UA server with the configured name. Inside the device node, you will see:

- One folder per configured port
- Inside each port folder, one tag for the entire message
- Plus one tag for each configured field

For example, a device with four fields would expose five tags in its port folder. See the [Address Browsing](#) section for information on browsing these tags.

Connecting to a Device

Instead of connecting directly to a PLC, these drivers connect to ports. The connected device must be configured to send data to those ports. You can configure multiple ports for a single device connection.

These drivers require the OPC UA module to be installed and enabled. If the module is missing or disabled, the devices will fault with a “Missing Dependency” error on the **Platform > System > Modules** page.

TCP Driver

The TCP driver can act as a passive receiver, or allow writes to be sent back to the device through tags if writeback is enabled.

Connecting to a TCP Device

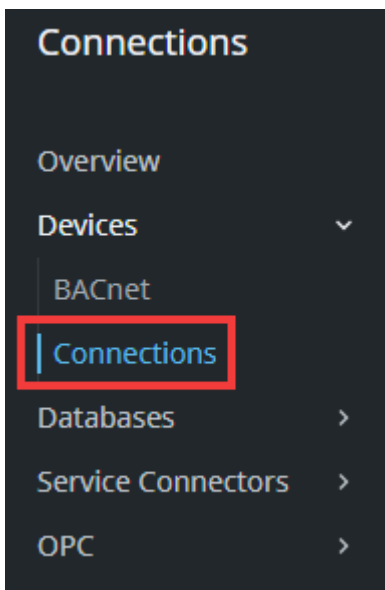


INDUCTIVE
UNIVERSITY

Connecting to TCP Device

[Watch the video](#)

1. On the Gateway, go to **Connections > Devices > Connections**.



2. Click **Create Device Connection +**.
3. Select **TCP Driver** from the list and click **Next**.

4. On the **Configure Device** screen, enter the required fields:

- **Name:** The desired name for the device (e.g., TCP)
- **Port(s):** The TCP port(s) to connect to, comma-separated. (e.g., 12345)
- **Address:** The IP address of the device (e.g., 10.20.3.45)

5. Click **Create Device Connection**.

The device will be listed now on the Connections page. The status will initially show Disconnected, but will move to a Connected state.

TCP Device Settings

General

Name	Description
Name	A user-defined name for the device. This name appears in OPC item paths and the Devices list. Name must begin with an underscore or a letter, and cannot include special characters.
Description	Optional description to help identify the device.
Enabled	When selected, the device is active and available for use.

Connectivity

Name	Description
Port(s)	Specifies the TCP port or ports to connect to. Multiple ports can be entered as a comma-separated list.
Address	Specifies the IP address of the device to connect to.
Local Address	Defines the local IP address to bind from, if needed.
Connect Timeout	Sets the time, in milliseconds, to wait for a socket connection. A value of zero disables the timeout.
Inactivity Timeout	Sets the time, in milliseconds, of allowed inactivity before the connection is reset. A value of zero disables the timeout.

Message

Name	Description
Message Delimiter Type	Defines how message boundaries are determined. Options include Packet Based, Character Based, or Fixed Size.
Message Delimiter	Specifies the characters or length used to identify the end of a message, depending on the delimiter type.
Field Count	Sets the expected number of fields in each message.
Field Delimiter	Specifies the characters used to separate fields within a message.

Writing

Name	Description
Writeback Enabled	Enables writable tags to send data to the device.
Writeback Message Delimiter	Specifies the delimiter that signals the end of a message during writeback operations.
Write Timeout	Sets the time, in milliseconds, to wait for a write to complete. The default value is 5000.

TCP Tags

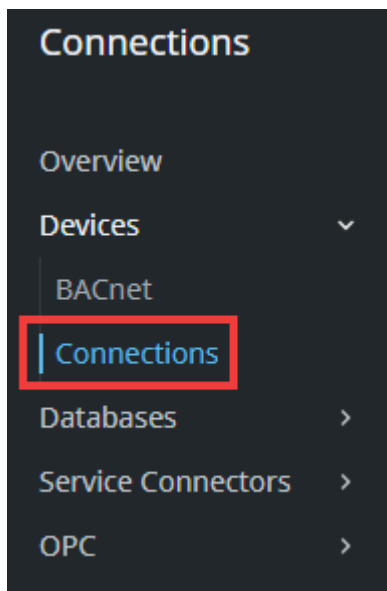
Name	Description
Last Receive Time	Displays the timestamp of the last message received from the device.
Message	Shows the string representation of the last received message.
MessageBytes	Displays the binary (byte-level) representation of the last received message.
Writable	Allows writing of string data to the device if writeback is enabled.
WritableBytes	Allows writing of binary data to the device if writeback is enabled.

UDP Driver

The UDP driver is a passive listener only. It does not send data back to the device and is well-suited for broadcast-type devices.

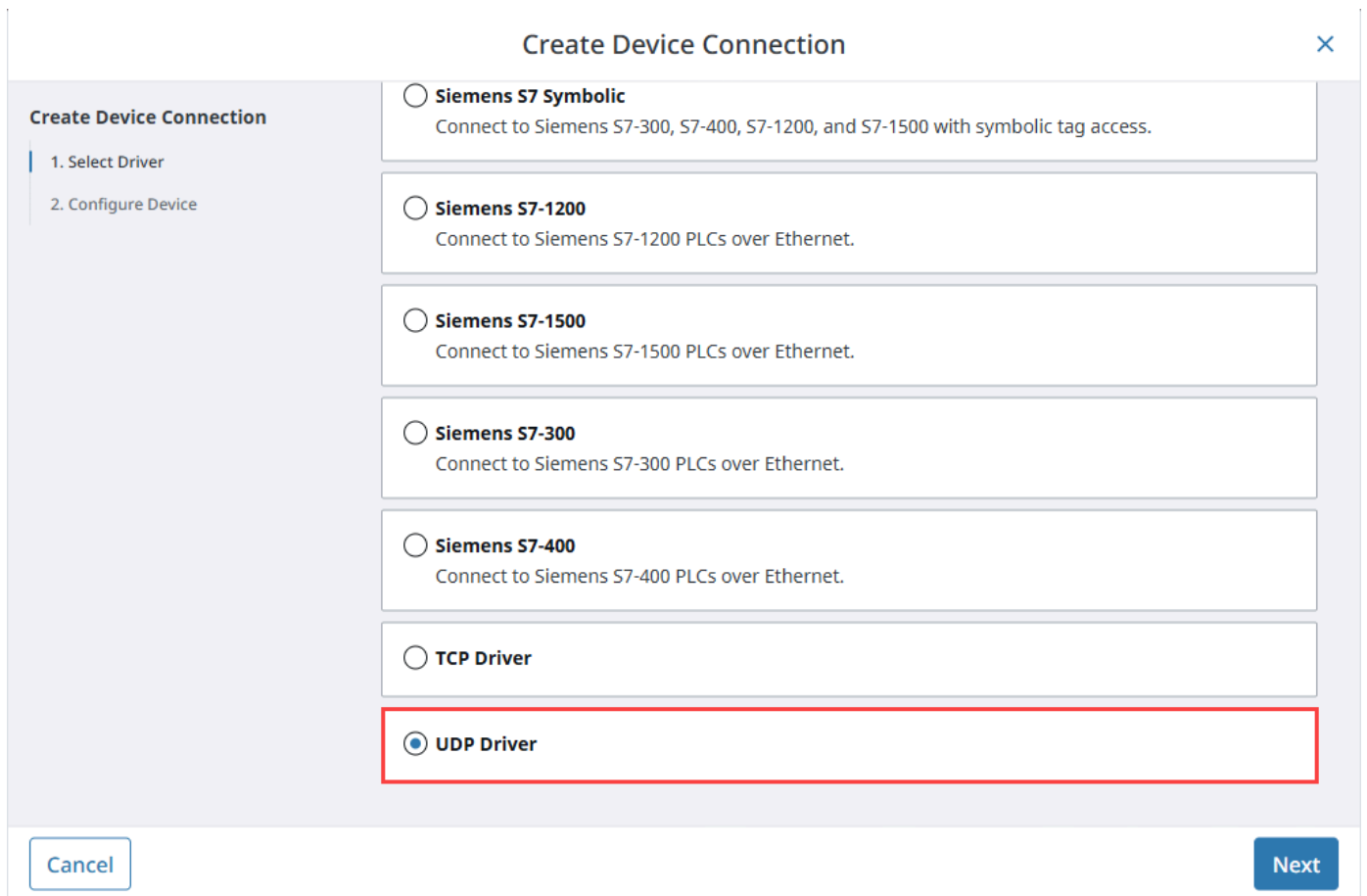
Connecting to a UDP Device

1. On the Gateway, go to **Connections > Devices > Connections**.



2. Click **Create Device Connection +**.

3. Select **UDP Driver** from the list and click **Next**.



4. On the **Configure Device** screen, enter the required fields:

- **Name:** The desired name for the device (e.g., UDP)
- **Port(s):** The UDP port(s) to connect to, comma-separated. (e.g., 12345)
- **Address:** The IP address of the device (e.g., 0.0.0.0 to listen on all interfaces)

GENERAL

*Required

Name *

UDP

The name of this device connection.

Description

Enabled

☒ Set whether this device connection is enabled or disabled within the Gateway.

CONNECTIVITY

*Required

Address *

0.0.0.0

The IP address to connect to.

Port(s) *

12345

The port(s) to connect to.

5. Click **Create Device Connection**.

The device will be listed now on the Connections page. The status will initially show Disconnected, but will move to a Connected state.

Device Connections

Connections

Filter device connections...		View	20	Create Device Connection +	
☐	NAME	DRIVER	ENABLED	STATUS	
☐	UDP	UDP Driver	true	CONNECTED	

Showing 1 of 1 items

UDP Device Settings

General

Name	Description
Name	A user-defined name for the device. This name appears in OPC item paths and the Devices list. Name must begin with an underscore or a letter, and cannot include special characters.
Description	Optional description to help identify the device.
Enabled	When selected, the device is active and available for use.

Connectivity

Name	Description
Address	Specifies the IP address that the device will listen on.
Port(s)	Defines the UDP port or ports to listen on. Multiple ports can be entered as a comma-separated list.

Message

Name	Description
Message Delimiter Type	Defines how message boundaries are determined. Options include Packet Based, Character Based, or Fixed Size.
Message Delimiter	Specifies the characters or length used to identify the end of a message, depending on the delimiter type.
Field Count	Sets the expected number of fields in each message.
Field Delimiter	Specifies the characters used to separate fields within a message.

UDP

Name	Description
Message Buffer Size	Sets the size of the message buffer, in bytes.
Multicast	Enables multicast mode for receiving messages.

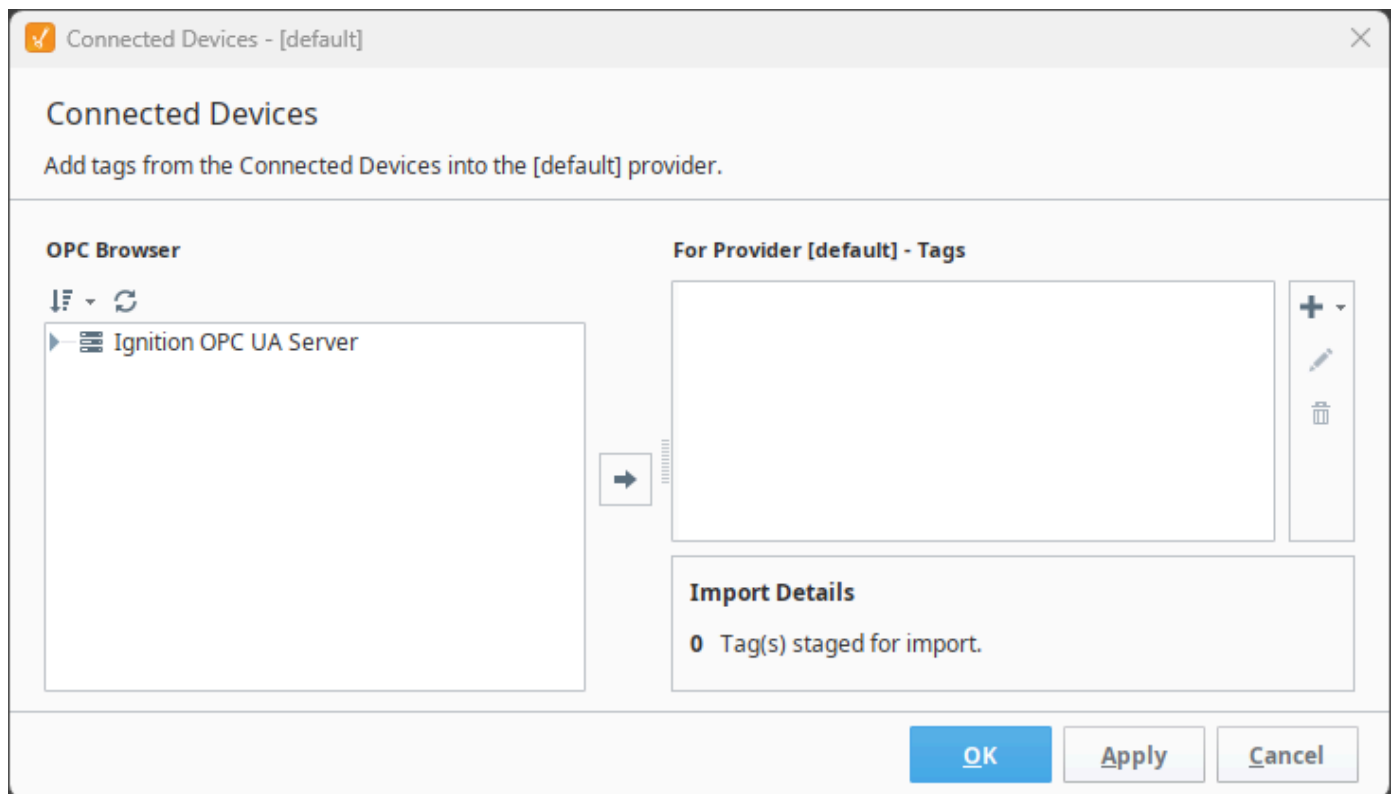
UDP Tags

Name	Description
Last Receive Time	Displays the timestamp of the last message received from the device.
Message	Shows the string representation of the last received message.
MessageBytes	Displays the binary (byte-level) representation of the last received message.

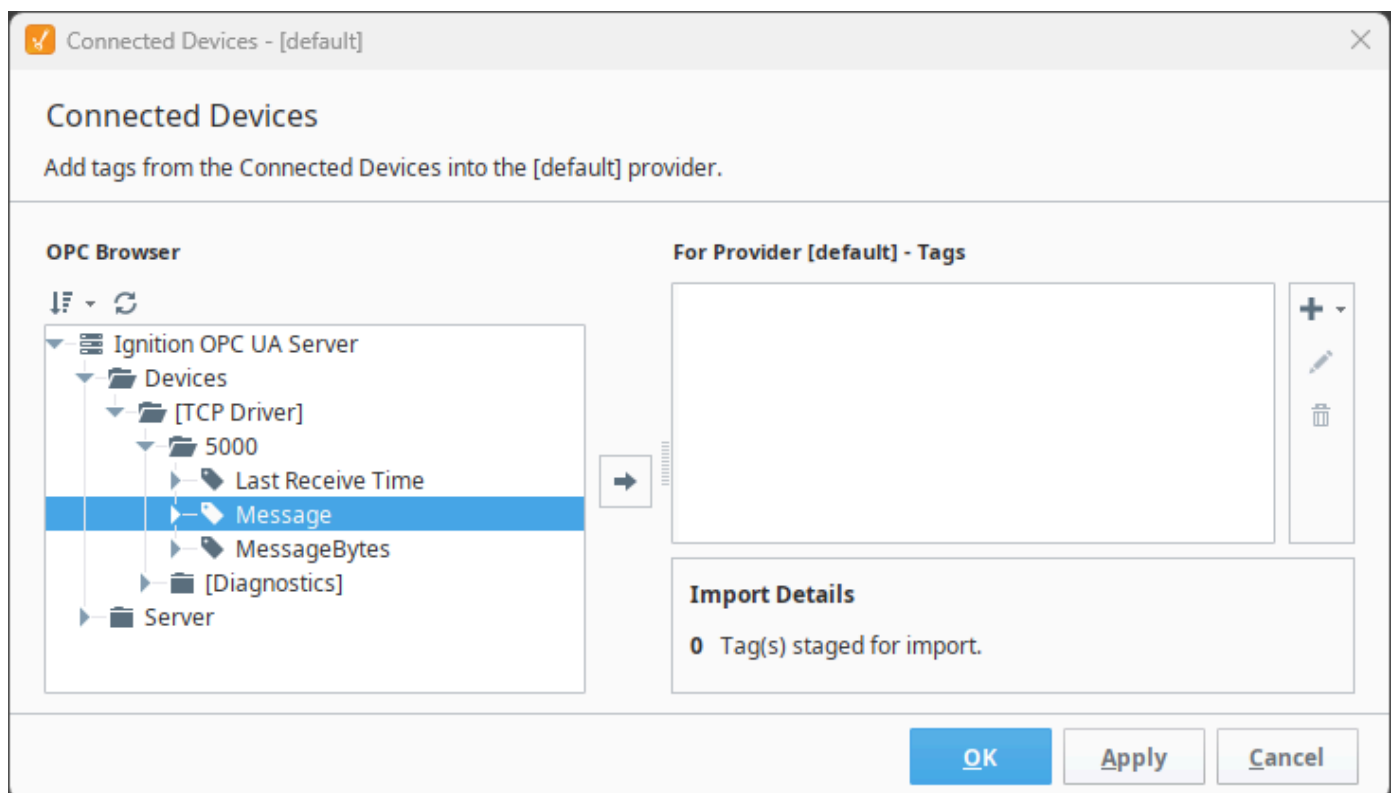
Address Browsing

Once your TCP or UDP connection is set up, you can start browsing the driver for data points. The example below uses the TCP driver and assumes you have already connected to a valid port and IP address. For instructions on setting up a TCP driver connection, see [Connecting to a TCP Device](#).

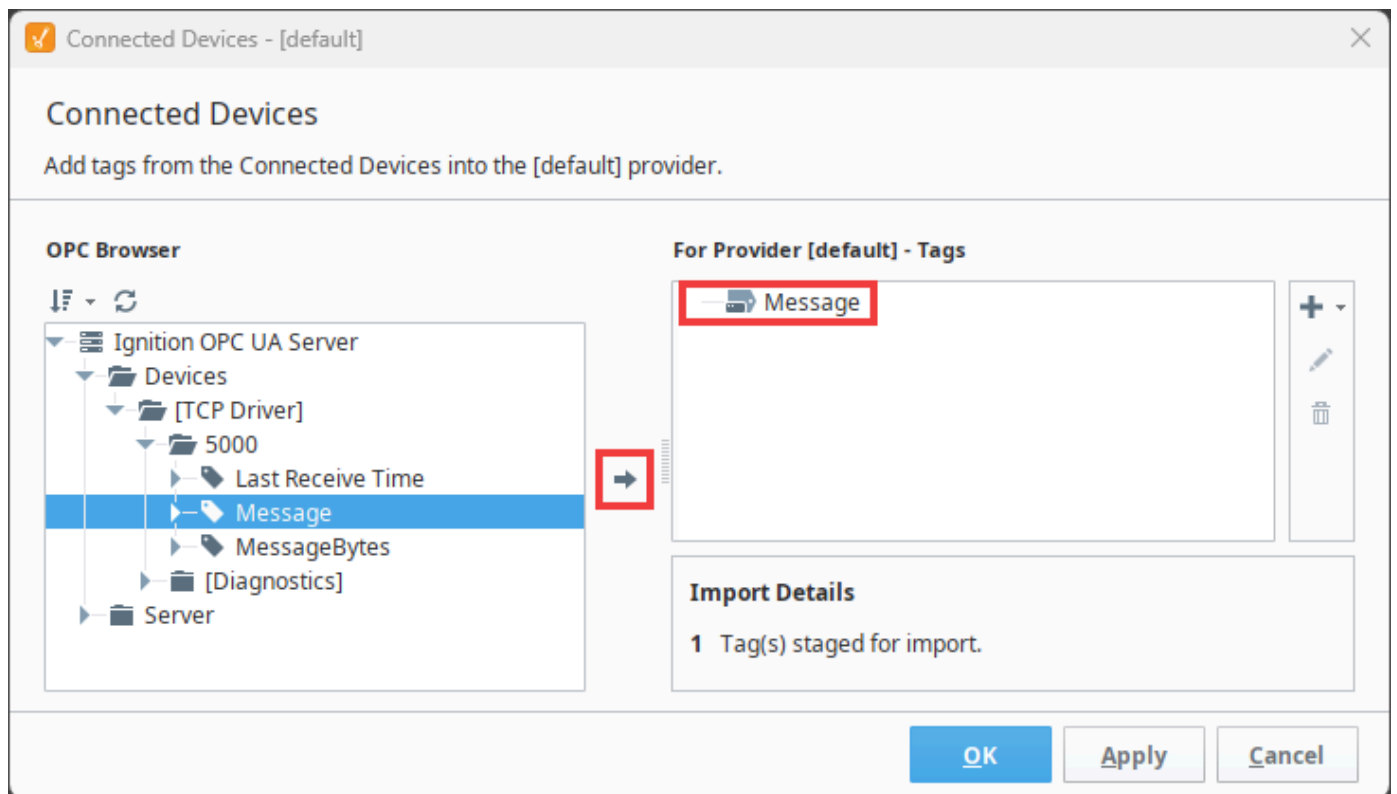
1. In your Tag Browser, click on the **Add +** icon, then click on **Browse Devices...** . The Connected Devices popup window will open.



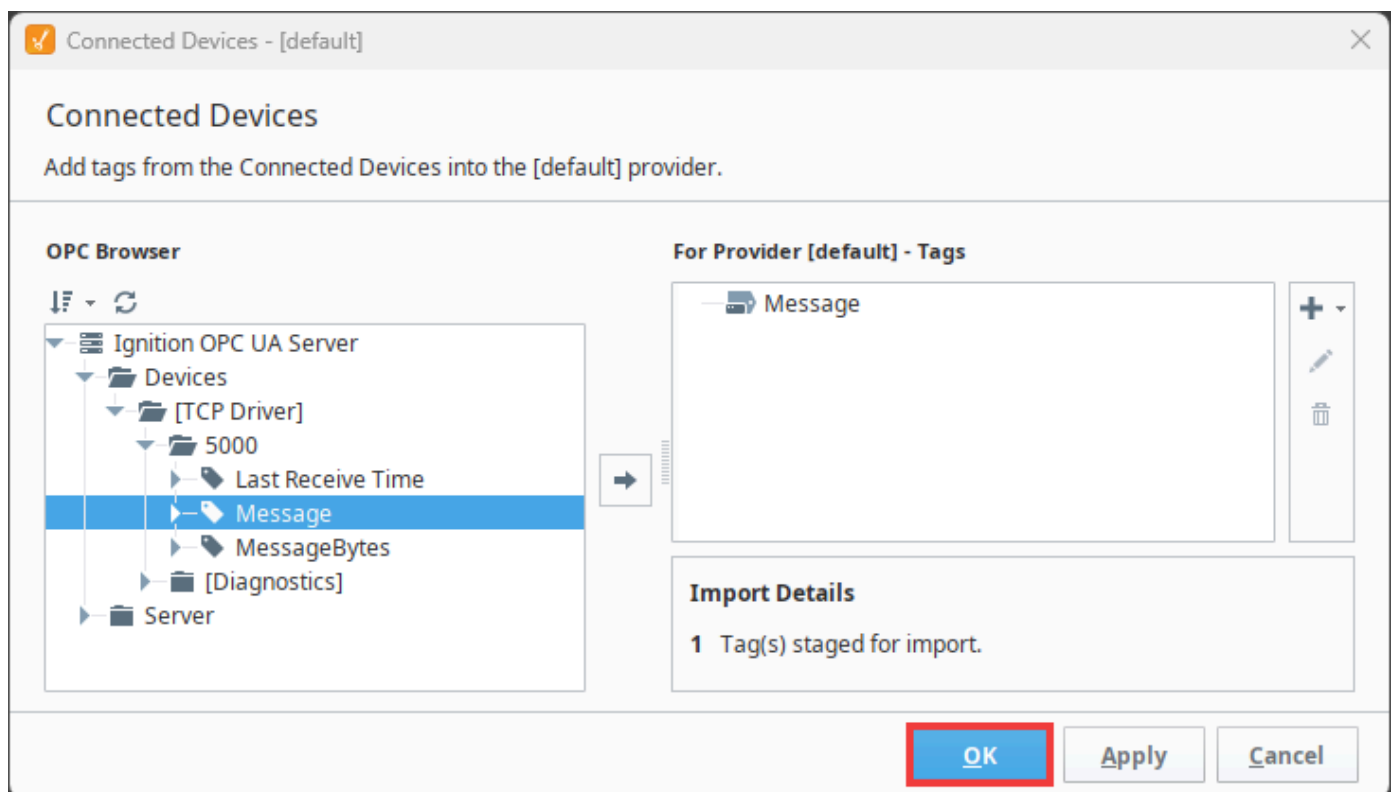
2. Navigate to the tag you want to import. In this example, since we want to add the Message in port 5000, we'll go through **Ignition OPC UA Server > Devices > [TCP Driver] > 5000 > Message**.



3. Select the tag or folder to import and add it to your Tag Provider by clicking on the Add ➡ icon. This will stage the tag for when you are ready to import, in case you want to add any other tags or folders.



4. Once you have added the tags and folders you want, click **OK** to add them to your Tag Browser.



Your data is now available to use in Ignition.

Message Delimiters

Some devices use control characters to signal the end of a message. Tag values update no faster than the scan class they use. If messages arrive more quickly than the scan class, Ignition will drop

them. Typical reliable scan rates are 100–200 ms, though TCP can theoretically scan down to 25 ms, and UDP depends heavily on network conditions. You can specify these in the driver settings.

Characters	Description
\t	Represents a tab character.
\b	Represents a backspace character.
\n	Represents a newline character.
\r	Represents a carriage return character.
\f	Represents a form feed character.



TIP

If tags show unexpected bad quality, check your delimiter and field count settings. Mismatches will result in `BAD_CONFIG_ERROR`.

DNP3

DNP3 is a communication protocol commonly used in water and electric utilities. It connects a central master station to multiple remote devices, known as outstations, creating a distributed network without overwhelming the communication infrastructure. This page covers connection details and general protocol requirements for using DNP3.

Compared to Modbus, DNP3 is a newer and more robust protocol. It supports a wider range of features but is also more complex. Always consult your device documentation to confirm whether DNP3 is supported.

NOTE

The DNP3 drivers do not support DNP3 Secure Authentication.

DNP3 Drivers

Ignition provides two DNP3 drivers:

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

The DNP3 driver is installed by default on new Ignition installations. The DNP3 Driver (Legacy) remains available as an optional module but no longer receives updates or ongoing support.

Scripting Functions

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

- The DNP3 Driver uses the [system.dnp](#) scripting namespace.
- The DNP3 Driver (Legacy) uses the [system.dnp3](#) namespace.

See the scripting functions appendix for details.

Supported Operating Systems

Both DNP3 drivers are supported on:

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

 NOTE

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

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.

Upgrading from DNP3 Driver (Legacy)

To migrate from the DNP3 Driver (Legacy):

1. Delete or rename the existing Legacy device.
2. Create a new DNP3 Driver device and in the name field, use the old DNP3 Driver (Legacy) name.
3. Update any tag references as needed.

 TIP

Review the new driver's settings carefully, since its configuration and polling model differ from the Legacy driver.

Addressing and Data Acquisition

Ignition's DNP3 drivers use either event-based polling, unsolicited messaging, or group/variation/index (GVI) addressing for explicit data reads from the PLC. See the [DNP3 Driver](#) and [DNP3 Driver \(Legacy\)](#) pages for the specific types of data acquisition each driver uses.

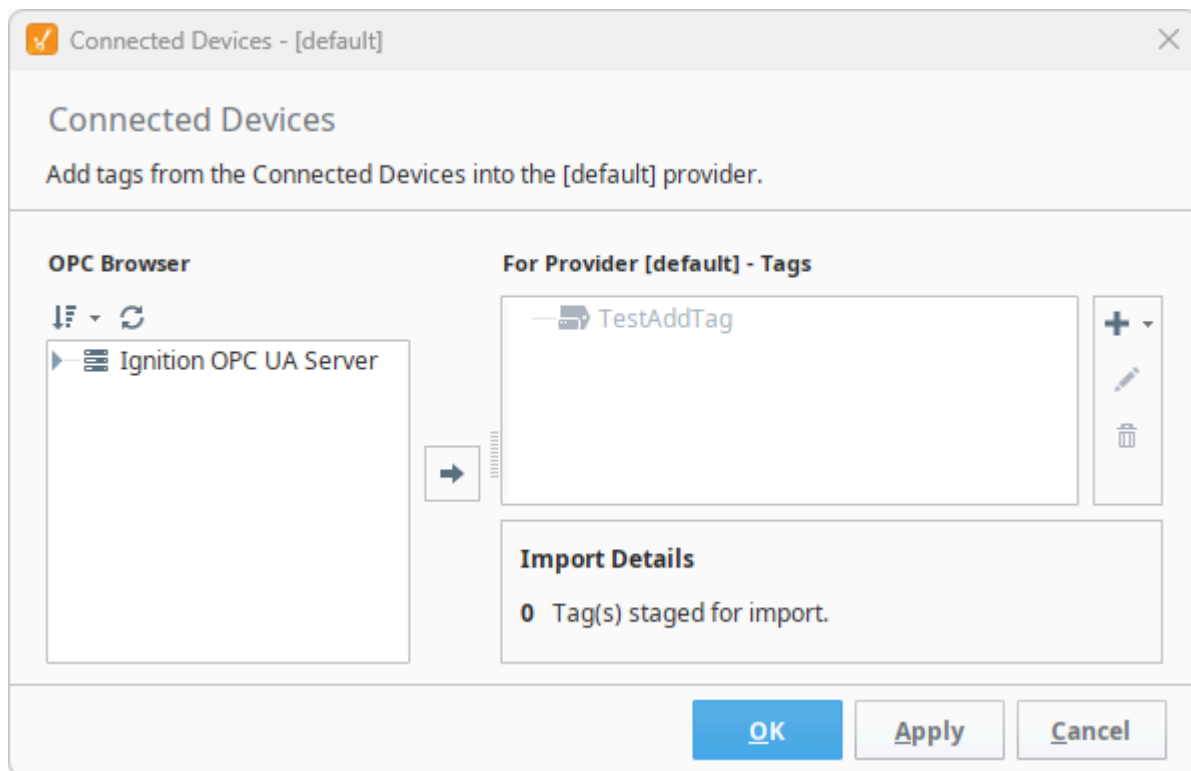
Addressing and Browsing DNP3 Points Examples

These sections assume you have already set up a DNP3 connection to a supported device. For instructions on connecting to a device using DNP3, see the [DNP3 Driver](#) page, or the [DNP3 Legacy Driver](#) page, depending on the DNP3 driver you are using.

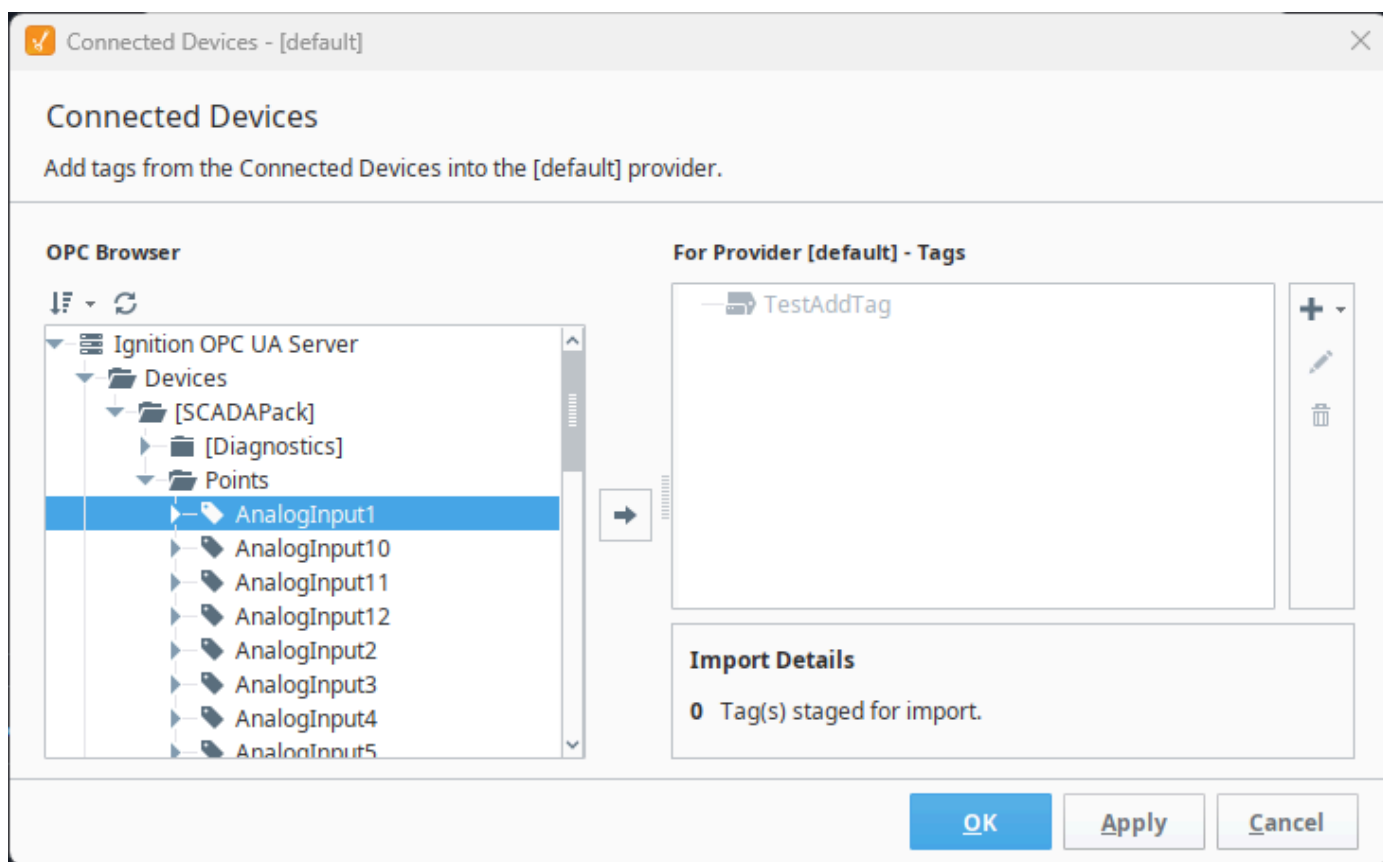
Browsing Points With DNP3

This example uses the DNP3 driver to browse for points within the Tag Browser's Connected Devices window.

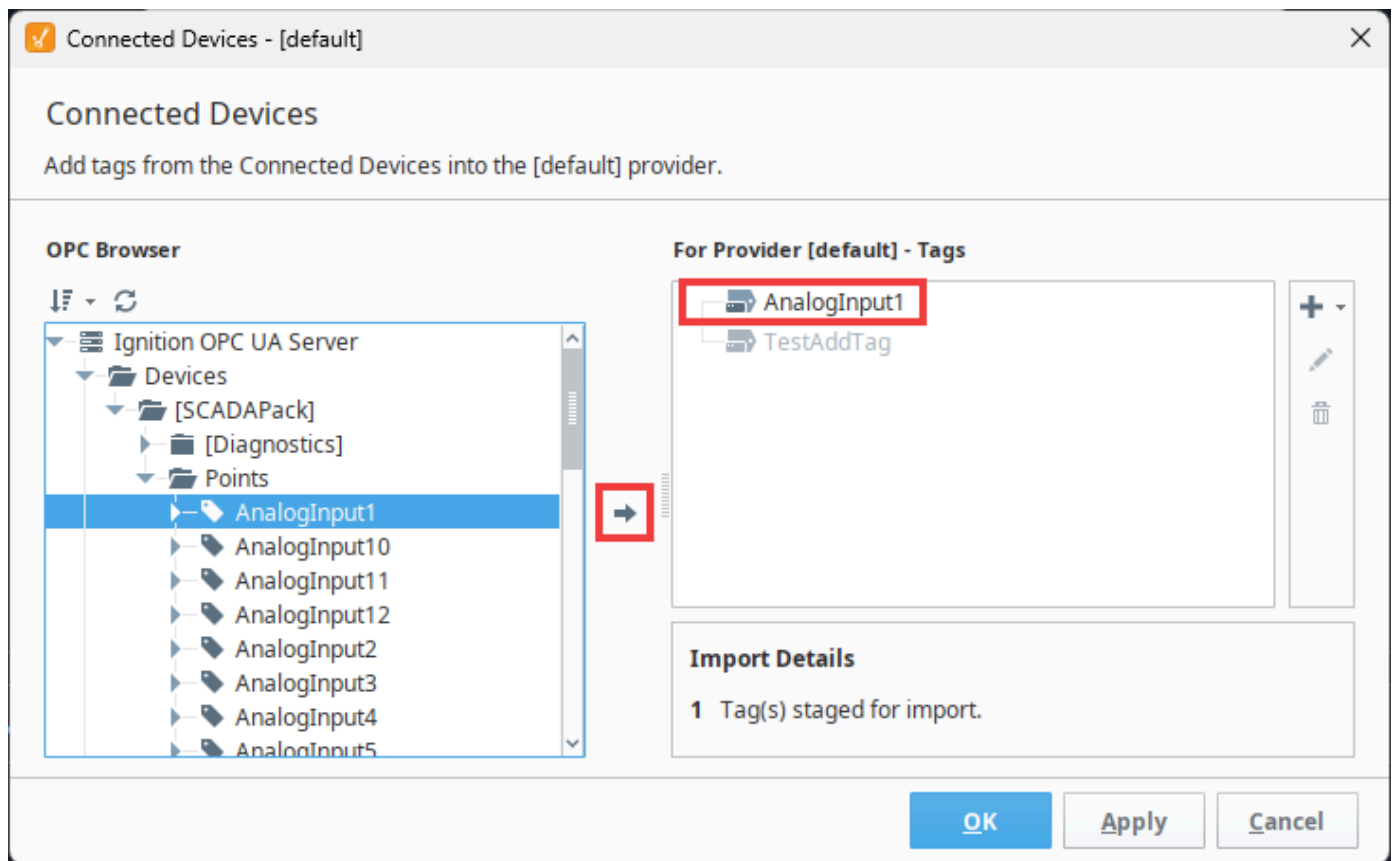
1. In your Tag Browser, click on the **Add +** icon, then click on **Browse Devices...** . The Connected Devices popup window will open.



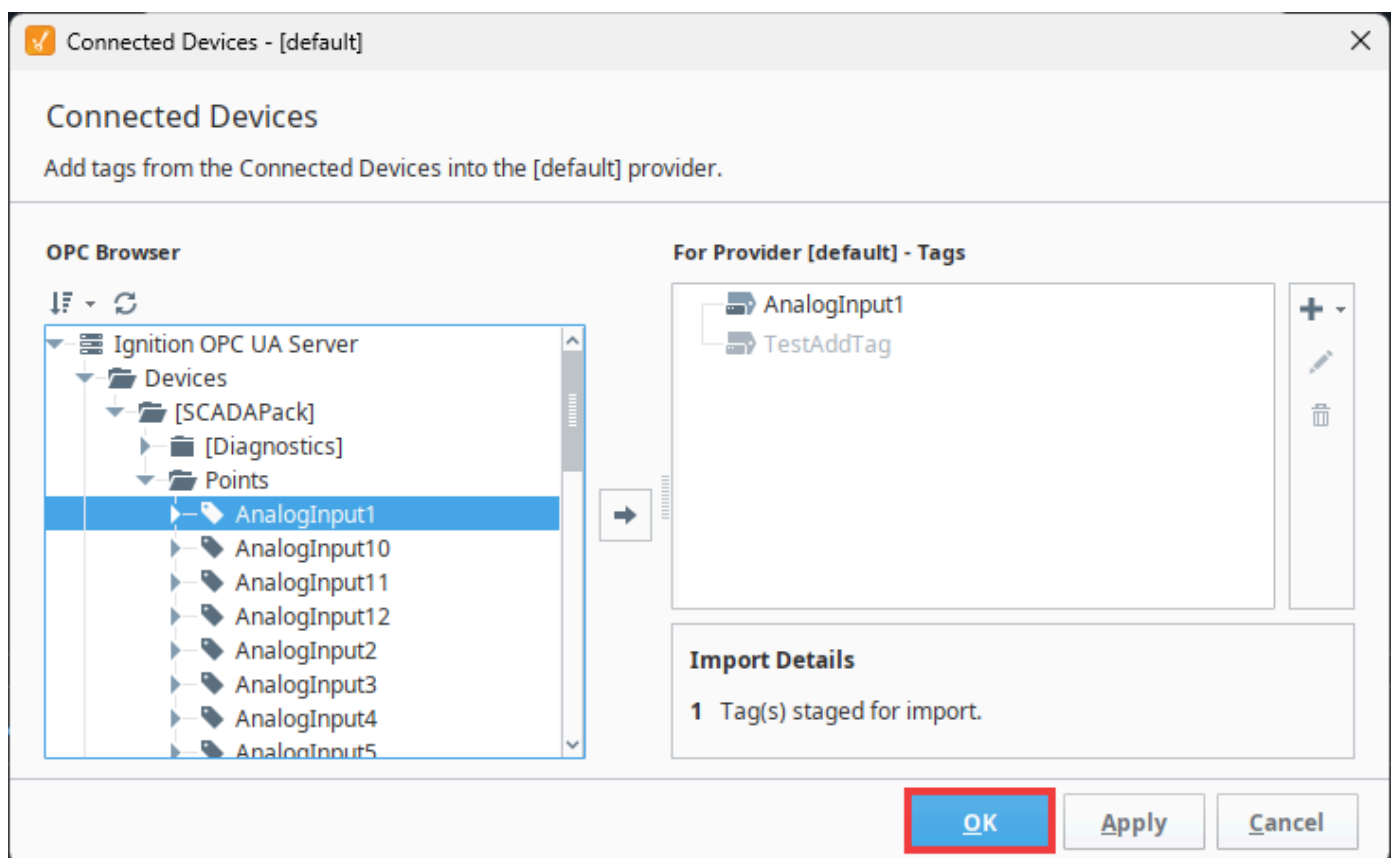
2. Navigate to the tag you want to import. In this example, since we want **AnalogInput1** within our connected device, we'll go through **Ignition OPC UA Server > Devices > [SCADAPack] > Points > AnalogInput1**.



3. Select the tag or folder to import and add it to your Tag Provider by clicking on the Add ➡ icon. This will stage the tag for when you are ready to import, in case you want to add any other tags or folders.



4. Once you have added the tags and folders you want, click **OK** to add them to your Tag Browser.



Your tags and folders are now available to use in Ignition.

Browsing Points With Legacy DNP3 (Aliased Points)

This example uses the legacy DNP3 driver to add a single aliased point. If you want to add multiple aliased points at the same time, see [Importing Global Variables - CSV File](#).

NOTE

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

- 1. Navigate to the **Gateway Webpage > Connections > Devices > Connections**.
- 2. Click the **three dots menu** of the legacy DNP3 connection you want to add a data point for, then click **Aliased Points**.

☐	SCADAPack	DNP3 Driver (Legacy)	true	CONNECTED	
☐	Modbus TCP	Modbus TCP	false	DISAB	Edit Rename Duplicate Aliased Points Roles & Permissions Disable
☐	Modbus RTU Over TCP	Modbus RTU over TCP	false	DISAB	
☐	Modbus RTU	Modbus RTU	false	DISAB	
☐	DNP3 2	DNP3 Driver	false	DISAB	
☐	DNP3	DNP3 Driver	false	DISAB	
☐	AB SLC	Allen-Bradlev SLC	false	DISAB	

- 3. The Program panel for your legacy DNP3 connection will open. Click on the **Add +** button on the top right of the panel. This will add a new entry to the table where you can specify an aliased point.

×

SCADAPack Program

Import Configuration File

Reset

Export

Remove All

Add +

POINT ADDRESS

BROWSE PATH

DESCRIPTION

- 4. Enter the information for the aliased point you want to map, then click **Save Changes** on the bottom right.

Import Configuration File Reset Export 

Remove All

Add +

POINT ADDRESS

BROWSE PATH

DESCRIPTION

g30v1i1	/DNP3Legacy/AnalogInput	AnalogInput:001	Delete
g30v1i2	/DNP3Legacy/AnalogInput	AnalogInput:002	Delete
g30v1i3	/DNP3Legacy/AnalogInput	AnalogInput:003	Delete

Showing 3 of 3 items

Cancel

Save Changes

Your aliased point is now exposed to Ignition. You can now browse for this aliased point in the Tag Browser's Connected Devices window and add it to Ignition by following the [Browsing Points with DNP3 Example](#) above.

Importing Global Variables - CSV File

This example describes how to import CSV configuration files to define aliased points for the DNP3 legacy driver.

1. On the Gateway, go to **Connections > Devices > Connections**.
2. Locate your DNP3 device, click the **three dots menu**, and choose **Aliased Points**.
3. Click **Import Configuration File** and upload your CSV.
4. Review the imported entries, then click **Save Changes**.

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.

DNP3 IIN Bit	Indicator	Description
IIN1.0	Broadcast Message Received	A broadcast message was received.
IIN1.1	Class 1 Events	Additional Class 1 event data is available.
IIN1.2	Class 2 Events	Additional Class 2 event data is available.
IIN1.3	Class 3 Events	Additional Class 3 event data is available.
IIN1.4	Need Time (Time Synchronization Required)	The outstation requires time synchronization.
IIN1.5	Local Control (Digital Outputs Paused)	One or more outputs are in local (manual) control mode.
IIN1.6	Device Trouble	An abnormal condition exists within the device.
IIN1.7	Device Restart	The outstation has restarted.
IIN2.0	No Func Code Support	The requested function code is not implemented.
IIN2.1	Object Unknown/Unknown Type ID	The requested object is unknown or unsupported.
IIN2.2	Parameter Error	There was an error with one or more request parameters.
IIN2.3	Event Buffer Overflow	The outstation's event buffer has overflowed.
IIN2.4	Already Executing	A previously requested operation is still executing.

DNP3 IIN Bit	Indicator	Description
IIN2.5	Configuration Corrupt	The outstation's configuration is corrupted.
IIN2.6	Reserved	This bit is reserved to ensure future compatibility.
IIN2.7	Reserved	This bit is reserved to ensure future compatibility.

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 and Alarm Journals
- Tag History (in On Change mode)
- Tag Event Scripts

Once playback completes, the tag resumes showing live values.



TIP

This Buffered Events feature can only be used by devices that support Sequence of Events (SoE). The SoE data type may differ from the standard data type returned by subscriptions. This may be due to a configuration on the Default Variation property in the device.

DNP3 Driver

The DNP3 Driver provides communication with DNP3 outstations over Ethernet, supporting event-based polling, unsolicited messaging, and explicit reads using the **DNP3** protocol. For information about the older version, see [DNP3 Driver \(Legacy\)](#).

NOTE

The DNP3 Driver does not support DNP3 Secure Authentication.

Data Acquisition

The DNP3 Driver supports three modes of acquiring data.

- **Event-based polling:** The primary method, using configured polling intervals for Class 1, 2, and 3 events. Events that occur between polls are preserved and processed sequentially. OPC UA clients with a sampling interval of 0 will receive all events in order.
- **Unsolicited messaging:** Available on supported outstations, where the device pushes events as they occur. The driver processes these as they arrive. Sampling interval behavior for OPC UA clients applies similarly to event-based polling.
- **Explicit reads:** Used with group/variation/index (GVI) addressing, where static values are explicitly requested at each poll. No events are processed in this mode. This can approximate Legacy driver behavior if GVI tags are configured.

Connecting to a Device



INDUCTIVE
UNIVERSITY

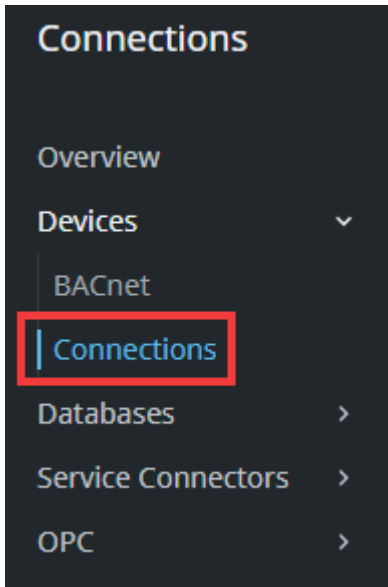
Connection to DNP3 Devices

[Watch the video](#)

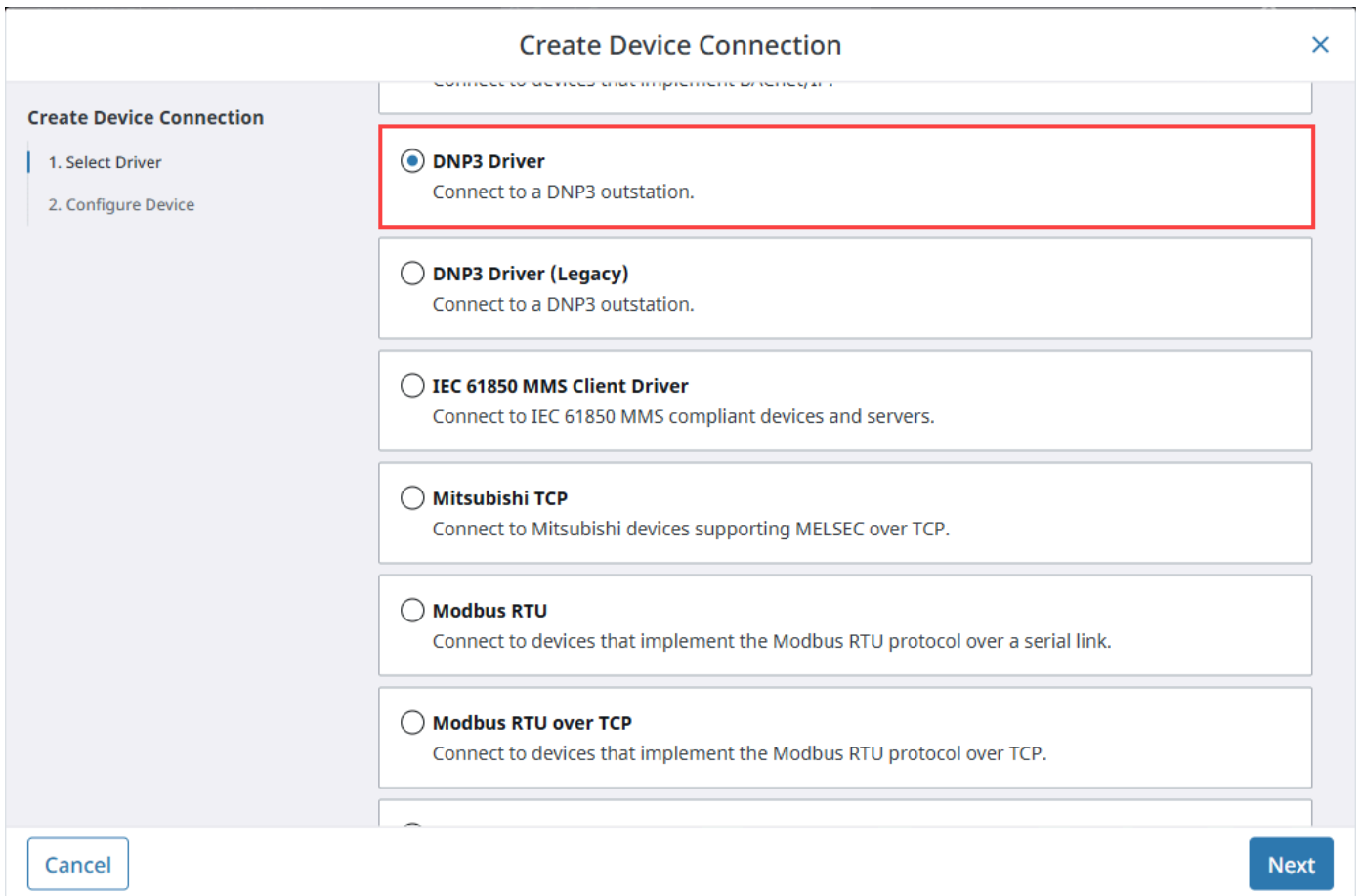
This driver requires the OPC UA module to be installed and enabled. If the module is missing or disabled, the device will fault with a “Missing Dependency” error on the **Platform System > Modules** page.

Ignition's DNP3 driver can connect directly to any devices that support Ethernet communication through the master station. It is important to make a new device connection for each of the outstations (remote devices), setting the source and destination addresses for each in Ignition's device connection.

1. On the Gateway, go to **Connections > Devices > Connections**.



2. Click **Create Device Connection +**.
3. Select **DNP3 Driver** from the list and click **Next**.



4. On the **Configure Device** screen, enter the required fields:

- **Name:** The desired name for the device (e.g., DNP3)
- **Hostname:** The IP address of the PLC (e.g., 10.20.4.55)

MAIN

*Required

Name *

DNP3

The name of this device connection.

Description

Enabled

☒

 Set whether this device connection is enabled or disabled within the Gateway.

CONNECTIVITY

*Required

Hostname *

10.20.3.45

The hostname or IP address component of the outstation endpoint.

5. Click **Create Device Connection**.

The device will be listed now on the Connections page. The status will initially show Disconnected, but will move to a Connected state.

Device Connections

Connections

Filter device connections...

View20

Create Device Connection +

☐	NAME	DRIVER	ENABLED	STATUS	
☐	DNP3	DNP3 Driver	true	CONNECTED	

Showing 1 of 1 items

Connection Settings

Main

Name	Description
Name	A user-defined name for the device. This name appears in OPC item paths and the Devices list. Name must begin with an underscore or a letter, and cannot include special characters.
Description	Optional description to help identify the device.
Enabled	When selected, the device is active and available for use.

Connectivity

Property	Description
Hostname	The hostname or IP address of the outstation device.
Port	The port component of the outstation endpoint. 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.

Advanced Properties

Analog Operation

Property	Description
Command Mode	The command mode to use when issuing commands for analog outputs; DIRECT_OPERATE or SELECT_BEFORE_OPERATE. Default is DIRECT_OPERATE.
Read After Operate	Whether an explicit read should be issued after operating analog outputs. Useful when an outstation does not support generating analog output events or assigning them to a class. Default is false.
Read After Operate Delay	Delay, in milliseconds, the read operation after operating to the Analog outputs. Default is 0.

Binary Operation

Property	Description
Command Mode	The command mode to use when issuing commands for binary outputs; DIRECT_OPERATE or SELECT_BEFORE_OPERATE. Default is DIRECT_OPERATE.
Read After Operate	Whether an explicit read should be issued after operating binary outputs. Useful when an outstation does not support generating binary output events or assigning them to a class. Default is false.
Read After Operate Delay	Delay, in milliseconds, the read operation after operating to the Binary outputs. Default is 0.
Trip Close Code	The trip close code to use when issuing commands for binary outputs; NUL, CLOSE, or TRIP. Default is NUL.
Op Type	The operation type to use when issuing commands for binary outputs; LATCH or PULSE. Default is LATCH.
Count	The count to use when issuing commands for binary outputs. Default is 1.
On Time (ms)	The on-time to use when issuing commands for binary outputs. Default is 0.
Off Time (ms)	f-time to use when issuing commands for binary outputs. Default is 0.

Logging

Property	Description
Auto Time Sync	Enable automatic time synchronization when an outstation signals that it needs a time update by setting the NEED_TIME indicator bit. Default is true.
Application Layer Debug Logging	Whether to enable debug logging for the DNP3 application layer. Default is false.
Physical Layer Debug Logging	Whether to enable debug logging for the DNP3 physical layer. Default is false.

Addressing and Browsing DNP3 Points

You can **browse returned points** under the Tag Browser's Connected Devices window in the Designer.

Sequence of Events

You can preserve the sequence of events by tuning your Tag Group:

- **OPC UA Queue Size:** increase to handle the maximum number of anticipated events. This is typically much larger than the buffer in a physical outstation.
- **OPC UA Sampling Interval:** set to 0 to allow true event-based reporting. Nonzero sampling intervals may collapse multiple events into a single update.



TIP

Sequence of events is best used with devices that support high-frequency event reporting and maintain their own internal event buffers.

DNP3 Driver (Legacy)

The DNP3 Driver (Legacy) uses the **DNP3** protocol for explicit reads and unsolicited messaging to acquire data. This driver is no longer actively supported with updates or fixes. For the latest option, see **DNP3 Driver**.

NOTE

The DNP3 Driver (Legacy) does not support DNP3 Secure Authentication.

Data Acquisition

The DNP3 Driver (Legacy) driver supports two modes of acquiring data.

- **Explicit reads:** Uses the DNP3 Read function code to retrieve static (current) values. Events are not processed in this mode. Recommended when points are addressed with group/variation/index (GVI) syntax.
- **Unsolicited messaging:** Supports event-driven updates reported directly from the outstation. The driver processes these updates as they arrive. Sampling interval behavior for OPC UA clients still applies. Not all devices support unsolicited messaging.

Connecting to a Device



INDUCTIVE
UNIVERSITY

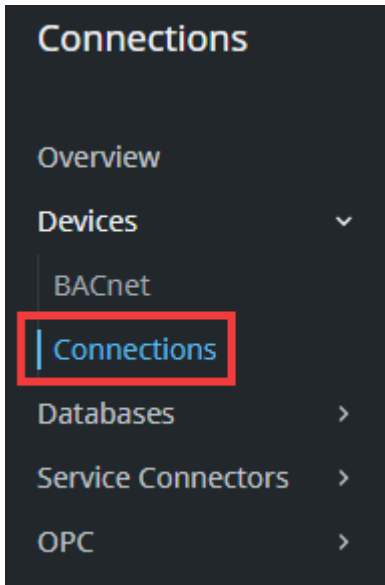
Connection to DNP3 Devices

[Watch the video](#)

This driver requires the OPC UA module to be installed and enabled. If the module is missing or disabled, the device will fault with a “Missing Dependency” error on the **Platform System > Modules** page.

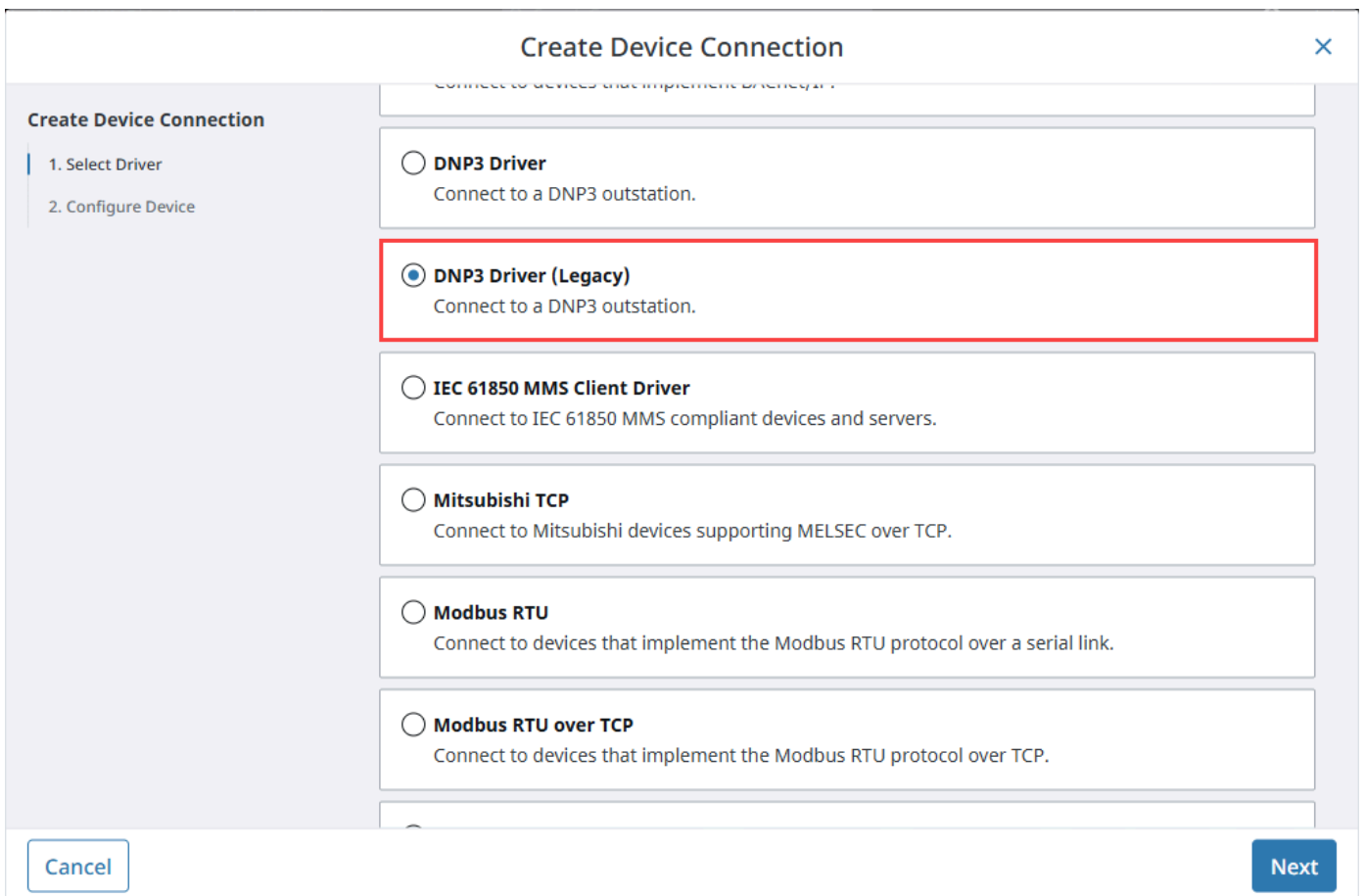
Ignition's DNP3 Driver (Legacy) can connect directly to any devices that support Ethernet communication through the master station. It is important to make a new device connection for each of the outstations (remote devices), setting the source and destination addresses for each in Ignition's device connection.

1. On the Gateway, go to **Connections > Devices > Connections**.



2. Click **Create Device Connection +**.

3. Select **DNP3 Driver (Legacy)** from the list and click **Next**.



4. On the **Configure Device** screen, enter the required fields:

- **Name:** The desired name for the device (e.g., LegacyDNP3)
- **Hostname:** The IP address of the PLC (e.g., 10.20.4.55)

MAIN

*Required

Name *

LegacyDNP3

The name of this device connection.

Description

Enabled

☒ Set whether this device connection is enabled or disabled within the Gateway.

CONNECTIVITY

*Required

Hostname *

10.20.3.45

The IP Address of the device.

5. Click **Create Device Connection**.

The device will be listed now on the Connections page. The status will initially show Disconnected, but will move to a Connected state.

Device Connections

Connections

Filter device connections...		View	20		Create Device Connection +
☐	NAME	DRIVER	ENABLED	STATUS	
☐	LegacyDNP3	DNP3 Driver	true	CONNECTED	

Showing 1 of 1 items

Device Connection Settings

Main

Name	Description
Name	A user-defined name for the device. This name appears in OPC item paths and the Devices list. Name must begin with an underscore or a letter, and cannot include special characters.
Description	Optional description to help identify the device.
Enabled	When selected, the device is active and available for use.

Connectivity

Property	Description
Hostname	The IP Address of the device.
Port	The port to use when connecting to a DNP3 device. The default port is 20000.
Local Address	The local address to connect from when establishing a TCP connection. If left blank, then the driver will simply pick an available address.
Source Address	Master station data link address, default 3.
Destination Address	Outstation data link address. Should be opposite of the source on the outstation. (e.g., Source Address is 3 and Destination Address is 4 or Source Address is 4 and Destination Address is 3)

Advanced

Property	Description
Message Fragment Size	The maximum size of a message fragment in the application layer, default is 249.
Message Timeout	The amount of time to wait for a message response from the outstation, default is 5,000.
Retries	The number of retries on a message timeout, default is 0.
Time Synchronization Enabled	When true, automatically synchronizes the gateway time to the outstation when requested, default is true.
Default Outstation Conformance Level	The default DNP3 Application Layer level subset to use when communicating with the outstation.
Direct Operate Enabled	When true, the Direct-Operate function code is used on a write, otherwise Select-Operate is used, default is true.
Unsolicited Messages Enabled	When true, the outstation may send unsolicited messages for Class 1, 2, and 3 data, default is false. Ignition will connect to the outstation, but not request any data from it. Ignition waits for the outstation to send data. Not all devices support this option; those that do need to be configured to use it. Please refer to your device's documentation for more information.
Integrity Poll Interval	The interval at which to perform an integrity poll, in milliseconds, default is 3,600,000.

Default Value Types

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

Addressing and Browsing DNP3 Points

When the driver connects to an outstation, it performs an integrity poll to map available points to the OPC UA server with group, variation, and index addressing. You can review these mapped points in the Designer under the [Connected Devices window](#).



INDUCTIVE
UNIVERSITY

About DNP3 Addressing

[Watch the video](#)

Point Types

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

For variation details, consult your device’s documentation.

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.

The options within the Aliased Points panel are described in the table below:

Button	Description
Import Configuration File	Allows users to import a CSV file to add many data points at once.
Reset	Resets the program to its default state before any changes were saved.
Export	Allows users to export the configuration file as a CSV.
Remove All	Removes all entries from the program.
Add +	Allows users to manually add individual data points.

When adding individual aliased points or importing configuration files, it is helpful to know what fields Ignition is expecting. These fields are detailed by the table below:

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	A user-defined description of the point. [Optional]

** NOTE**

If you change a point address after creating tags, you must update the OPC Item Path for those tags.

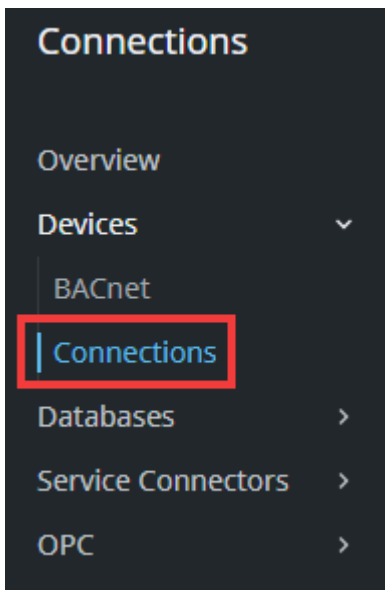
For a walkthrough of how to add an aliased point, see [Browsing Points With Legacy DNP3 Example \(Aliased Points\)](#).

Omron NJ Driver

Connect to an Omron NJ Device

This driver requires the OPC UA module to be installed and enabled. If the module is missing or disabled, the device will fault with a “Missing Dependency” error on the **Platform > System > Modules** page.

1. On the Gateway, go to **Connections > Devices > Connections**.



2. Click **Create Device Connection +**.
3. Select **Omron NJ Driver** and click **Next**.

Create Device Connection

Create Device Connection

1. Select Driver

2. Configure Device

☒ **Omron NJ Driver**
Connect to Omron NJ series PLCs.

☐ **Programmable Device Simulator**
A simulator device that can be configured with a user-defined hierarchy of static or function-driven values.

☐ **Siemens S7 Symbolic**
Connect to Siemens S7-300, S7-400, S7-1200, and S7-1500 with symbolic tag access.

☐ **Siemens S7-1200**
Connect to Siemens S7-1200 PLCs over Ethernet.

☐ **Siemens S7-1500**
Connect to Siemens S7-1500 PLCs over Ethernet.

☐ **Siemens S7-300**
Connect to Siemens S7-300 PLCs over Ethernet.

☐ **Siemens S7-400**

Cancel

Next

4. On the **Configure Device** screen, enter the required fields:
- **Name:** The desired name for the device (e.g., ONJ)
 - **Hostname:** The IP address of the PLC (e.g., 10.20.3.45)

GENERAL

*Required

Name *

ONJ

The name of this device connection.

Description

Enabled

☒ Set whether this nj device is enabled or disabled within the Gateway.

CONNECTIVITY

*Required

Hostname *

10.20.3.45

The hostname or IP address of the device.

5. Click **Create Device Connection**.

The device will be listed now on the Connections page. The status will initially show Disconnected, but will move to a Connected state.

Device Connections

Connections

Filter device connections...

View20

Create Device Connection +

☐	NAME	DRIVER	ENABLED	STATUS	
☐	ONJ	Omron NJ Driver	true	IDLE	

Showing 1 of 1 items

Device Settings

General

Name	Description
Name	A user-defined name for the device. This name appears in OPC item paths and the Devices list. Name must begin with an underscore or a letter, and cannot include special characters.
Description	Optional description to help identify the device.
Enabled	When selected, the device is active and available for use.

Main

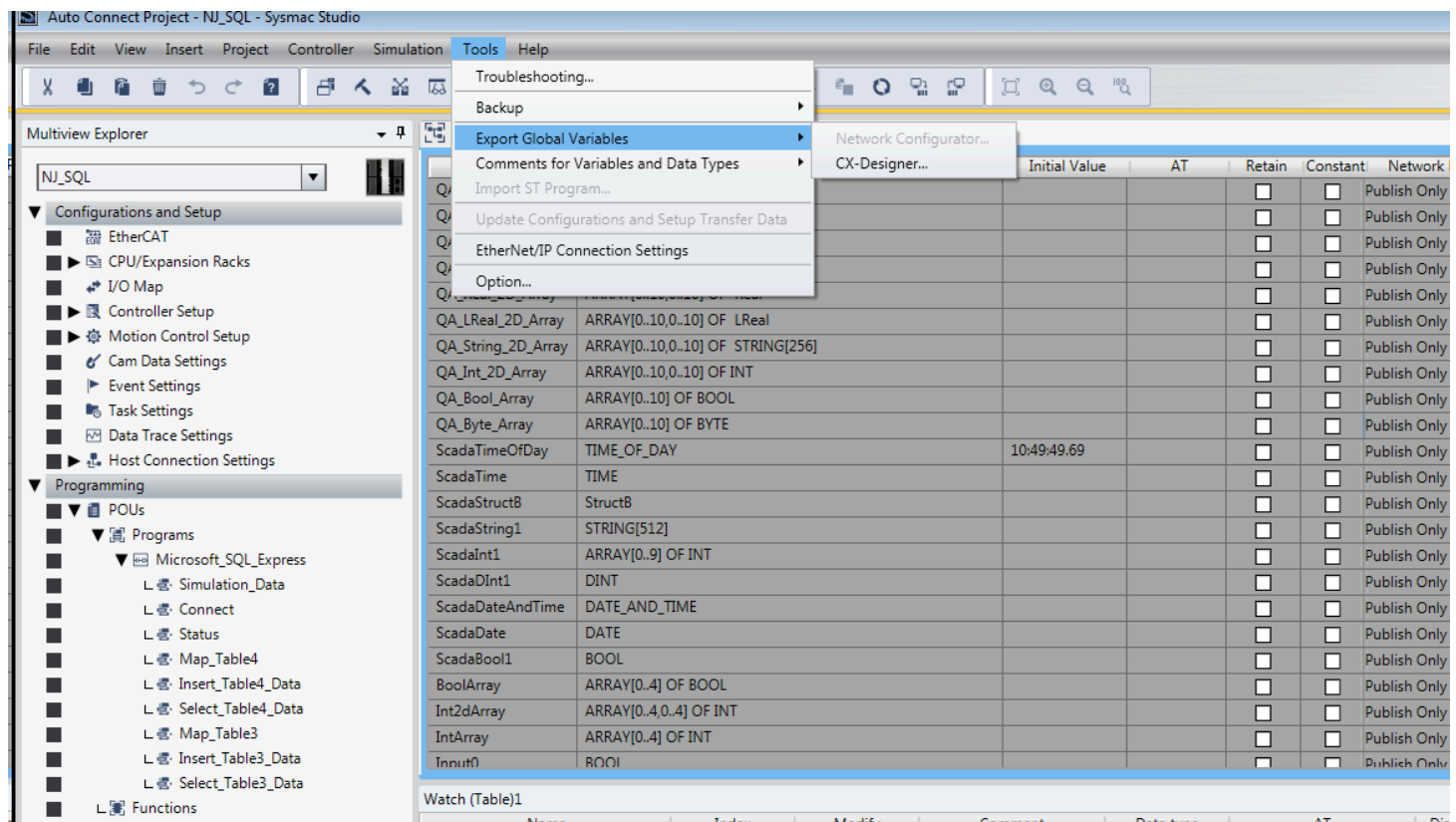
Name	Description
Hostname	Specifies the IP address of the device.
Local Address	Specifies the local address to connect from when establishing a TCP connection. If left blank, the driver will automatically select an available address.
Timeout	Sets the request timeout in milliseconds. The default value is 2,000.

Advanced

Name	Description
Connection Size	Specifies the CIP connection size to use. The default and maximum value is 1,994 bytes.
Date/Time Offset	Sets the offset, in hours, applied to date/time values when reading or writing. The default value is 0.
Slot Number	Specifies the slot number in the backplane where the CPU is located. The default value is 0.
Concurrency	Defines the number of concurrent requests the driver is allowed to issue. There is a 1:1 correlation between this value and the number of CIP connections used. The default is 2.

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. You can access this by clicking the three dot menu and selecting **Addresses**.

Device Connections

Connections

View 20 ▾ Create Device Connection +

☐	NAME	DRIVER	ENABLED	STATUS	
☐	ONJ	Omron NJ Driver	true	IDLE	

Showing 1 of 1 items

Edit
 Rename
 Duplicate
 Addresses
 View Details
 Roles & Permissions

Importing Tags

To manage tags, you can manually enter the tags, or import them from a tab-separated file. When importing, first click the **Import Configuration File** button, then choose a file to append tags to the table.

× ONJ Program

Import Configuration File Reset Export Remove All Add +

BROWSE NAME	DATA TYPE	CHARACTERS	ARRAY ELEMENTS	READ WRITE
-------------	-----------	------------	----------------	------------

Showing 0 of 0 items

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 table of tags, we have five columns of configuration per tag.

Column	Description
Browse Name	The address of the variable in the Omron device. Struct members are separated with periods.
Datatype	The datatype of the variable as found in the Omron device.
Characters	The maximum number of characters allowed for a String tag.
Elements	Indicates whether the tag is a scalar or an array. Refer to the section on specifying element counts for more details.
R/W	Specifies read/write access permissions for 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.

×

ONJ Program

Import Configuration File

Reset

Export

Remove All

Add

BROWSE NAME	DATA TYPE	CHARACTERS	ARRAY ELEMENTS	READ WRITE	
Scarlaint1	INT ▼		0..9	R ▼	Delete

Showing 1 of 1 items

❗ 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.

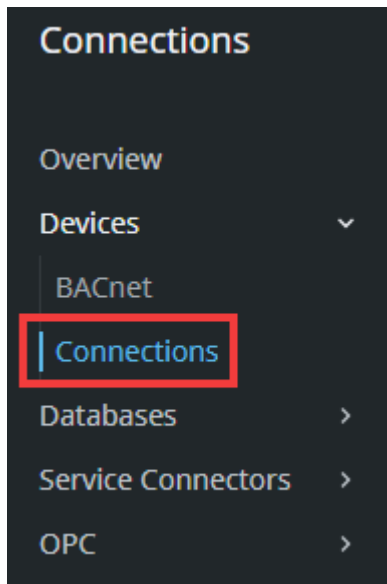
Omron FINS Driver

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

Connect to an Omron FINS Device via TCP

This driver requires the OPC UA module to be installed and enabled. If the module is missing or disabled, the device will fault with a “Missing Dependency” error on the **Platform > System > Modules** page.

1. On the Gateway, go to **Connections > Devices > Connections**.



2. Click **Create Device Connection +**.
3. Select **Omron FINS/TCP** and click **Next**.

Create Device Connection

Create Device Connection

1. Select Driver

2. Configure Device

SERVER

☒ **Omron FINS/TCP**
Connect to devices that implement FINS over TCP.

☐ **Omron FINS/UDP**
Connect to devices that implement FINS over UDP.

☐ **Omron NJ Driver**
Connect to Omron NJ series PLCs.

☐ **Programmable Device Simulator**
A simulator device that can be configured with a user-defined hierarchy of static or function-driven values.

☐ **Siemens S7 Symbolic**
Connect to Siemens S7-300, S7-400, S7-1200, and S7-1500 with symbolic tag access.

☐ **Siemens S7-1200**
Connect to Siemens S7-1200 PLCs over Ethernet.

Cancel

Next

4. On the **Configure Device** screen, enter the required fields:

- **Name:** The desired name for the device (e.g., FINSTCP)
- **Hostname:** The IP address of the PLC (e.g., 74.125.224.72)

GENERAL

*Required

Name *

FINSTCP

The name of this device connection.

Description

Enabled

☒ Set whether this fins tcp device is enabled or disabled within the Gateway.

CONNECTIVITY

*Required

Hostname *

74.125.224.72

Hostname or IP address of the target device.

5. Click **Create Device Connection**.

The device will be listed now on the Connections page. The status will initially show Disconnected, but will move to a Connected state.

Device Connections

Connections

🔍 Filter device connections...

View 20 ▾

Create Device Connection +

☐	NAME	DRIVER	ENABLED	STATUS	⚙
☐	FINSTCP	Omron FINS/TCP	true	RECONNECTWAIT	⋮

Showing 1 of 1 items

Omron FINS/TCP Settings

General

Name	Description
Name	A user-defined name for the device. This name appears in OPC item paths and the Devices list. Name must begin with an underscore or a letter, and cannot include special characters.
Description	Optional description to help identify the device.
Enabled	When selected, the device is active and available for use.

Connectivity

Name	Description
Hostname	The hostname or IP Address of the device.
Port	The port your Omron device is listening on. The default is 9600.
Local Address	Address of network adapter to connect from.
Timeout	The request timeout, specified in milliseconds. The default is 2,000.

FINS Settings

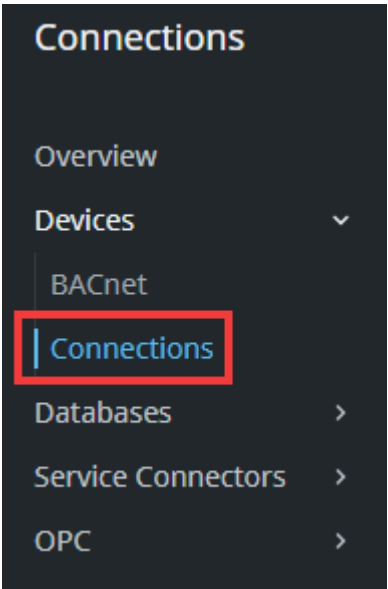
Name	Description	Valid Value
FINS Source Network	The address number of the source network (the Ignition Gateway).	Valid value is an integer between 0 and 127. The default setting is 0.
FINS Source Node	The number of the source node (the Ignition Gateway), typically the last octet of the IP address.	Valid 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

Name	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.

Connect to an Omron FINS Device via UDP

1. On the Gateway, go to **Connections > Devices > Connections**.



2. Click **Create Device Connection +**.
3. Select **Omron FINS/UDP** and click **Next**.

Create Device Connection

Create Device Connection

1. Select Driver

2. Configure Device

☒ **Omron FINS/UDP**
Connect to devices that implement FINS over UDP.

☐ **Omron NJ Driver**
Connect to Omron NJ series PLCs.

☐ **Programmable Device Simulator**
A simulator device that can be configured with a user-defined hierarchy of static or function-driven values.

☐ **Siemens S7 Symbolic**
Connect to Siemens S7-300, S7-400, S7-1200, and S7-1500 with symbolic tag access.

☐ **Siemens S7-1200**
Connect to Siemens S7-1200 PLCs over Ethernet.

☐ **Siemens S7-1500**
Connect to Siemens S7-1500 PLCs over Ethernet.

Cancel

Next

4. On the **Configure Device** screen, enter the required fields:

- **Name:** The desired name for the device (e.g., FINSUDP)
- **Bind Address:** The hostname or IP address of the Gateway (e.g., 74.125.224.10)
- **Remote Address:** The hostname or IP address of the device(e.g., 74.125.224.72)
- **FINS Source Node:** The number of the source node (the Ignition Gateway), typically the last octet of the IP address (e.g., 10)
- **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 (e.g., 72)
- **FINS Destination Unit:** The unit number of the destination device (the FINS device)

GENERAL

*Required

Name *

FINSUDP

The name of this device connection.

Description

Enabled



Set whether this fins udp device is enabled or disabled within the Gateway.

CONNECTIVITY

*Required

Bind Address *

74.125.225.10

The hostname or IP address of the Gateway.

Bind Port *

9600

The port you wish to create a UDP connection on for the Gateway.

Remote Address *

74.125.224.72

The hostname or IP address of the device.

Remote Port *

9600

The port your Omron device is listening on.

Timeout *

2000

The request timeout, specified in milliseconds.

FINS SETTINGS

*Required

FINS Source Network *

0

The address number of the source network (the Ignition Gateway).

FINS Source Node *

[Restore Default](#)

10

The number of the source node (the Ignition Gateway), typically the last octet of the IP address.

FINS Source Unit *

0

The unit number of the source device (the Ignition Gateway).

FINS Destination Network *

The address number of the destination network (the FINS device).

FINS Destination Node *

[Restore Default](#)

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.

FINS Destination Unit *

The unit number of the destination device (the FINS device).

5. Click **Create Device Connection**.

The device will be listed now on the Connections page. The status will initially show Disconnected, but will move to a Connected state.

Device Connections

Connections

View

20

Create Device Connection +

☐	NAME	DRIVER	ENABLED	STATUS	
☐	FINSUDP	Omron FINS/UDP	true	UNBOUND	

Showing 1 of 1 items

Omron FINS/UDP Settings

General

Name	Description
Name	A user-defined name for the device. This name appears in OPC item paths and the Devices list. Name must begin with an underscore or a letter, and cannot include special characters.
Description	Optional description to help identify the device.
Enabled	When selected, the device is active and available for use.

Connectivity

Name	Description
Bind Address	The hostname or IP address of the Gateway
Bind Port	The port you wish to create a UDP connection on for the Gateway. The default is 9600.
Remote Address	The hostname or IP address of the device.
Remote Port	The port your Omron device is listening on. The default is 9600.
Timeout	The request timeout, specified in milliseconds. The default is 2,000.

FINS Settings

Name	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 is 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.

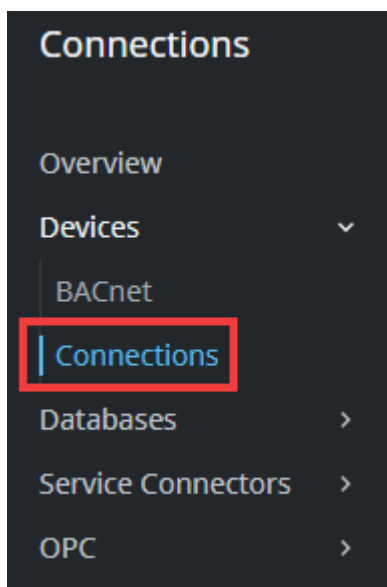
Request Optimization

Name	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. On the Gateway, go to **Connections > Devices > Connections**.



2. On the **Device Connections** page, click the **three dots menu** next to the Omron FINS device and select **Addresses**.

Device Connections

Connections

View 20

Create Device Connection +

☐	NAME	DRIVER	ENABLED	STATUS	
☐	FINSUDP	Omron FINS/UDP	true	BOUND	

Showing 1 of 1 items

Edit
 Rename
 Duplicate
 Addresses
 View Details
 Roles & Permissions

3. In the pop out panel, click the **Import Configuration File** button and upload the addresses from a CSV, TSV, or CXR file.

FINSUDP Program

Import Configuration File

Reset

Export

Remove All

Add +

BROWSE PATH	ADDRESS	DESCRIPTION
Showing 0 of 0 items		

4. After uploading a configuration file, click **Import**.

Import Program

×

You can find a few examples of CSV files on our website.

To see the examples, [visit our website](#).

Drag a file here or [click to upload](#)

Accepts CSV,TSV,CXR files.

Cancel

Import

5. The tags and their values will be displayed. Click **Save Configuration**.

×

FINSUDP Program

Import Configuration File

Reset

Export

Remove All

Add +

BROWSE PATH	ADDRESS	DESCRIPTION	
BrowseName	Datatype	Chars	Delete
Temperature	Float		Delete
Pressure	Float		Delete
Status	Boolean		Delete
Counter	Int32		Delete
ErrorCode	Int16		Delete
ProductionArray	Int16		Delete

Showing 7 of 7 items

Cancel

Save Changes

6. Once any changes are saved, 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.

NOTE

Valid banks are E00 through E18 (hex), totaling 25 banks.

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 using the settings in the table below.

Name	Value
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 also 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, which means users can simulate devices with the tag structure they are building for their project. Additionally, there are a number of built-in functions that will result in unique tag outputs rather than static values. For instance, this connection provides the ability to set specific values at different points of the program should the functions not provide exactly what you need.

Connecting to a Programmable Device Simulator

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

1. On the Gateway, go to the **Connections > Devices > Connections** page.
2. Click on **Create Device Connection**.
3. Select **Programmable Device Simulator** and click **Next**.

Create Device Connection

- 1. Select Driver
- 2. Configure Device

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.

☐ **Omron FINS/UDP**
Connect to devices that implement FINS over UDP.

☐ **Omron NJ Driver**
Connect to Omron NJ series PLCs.

☒ **Programmable Device Simulator**
A simulator device that can be configured with a user-defined hierarchy of static or function-driven values.

☐ **Siemens Driver**
Connect to Siemens S7-300, S7-400 PLCs with absolute addressing, or connect to S7-1200, S7-1500 PLCs with absolute addressing or symbolic tag access.

Cancel

Next

4. Give the device a name (i.e., GenSim), and click **Create Device Connection**.

Create Device Connection

- 1. Select Driver
- 2. Configure Device

Configure Device

MAIN

*Required

Name*

Description

Enabled

☒ Set whether this device connection is enabled or disabled within the Gateway.

Simulator Program*

Generic Program

SIMULATOR PROGRAM SETTINGS

*Required

Repeat Program

☒

Back

Create Device Connection

5. The page 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 expanding the three dots menu and

selecting **Edit Program**.

Device Connections

Connections

Filter device connections...

View 20

Create Device Connection +

☐	NAME	DRIVER	ENABLED	STATUS	
☐	GenSim	Programmable Device Simulator	true	🟢 RUNNING	Edit Rename Duplicate Edit Program Roles & Permissions Disable Delete
☐	Sample_Device	Programmable Device Simulator	true	🟢 RUNNING	

Showing 2 of 2 items

6. Edit instructions by using the Import, Export, Delete All, and Add buttons. These instructions determine what tags get made, and what their values are. Additionally, you can load a predefined program from the **Program** dropdown and click **Load Simulator Program**.

×

GenSim Program

Program

Select

Load Simulator Program

Import

Export

Delete All

Add +

TIME INTERVAL

BROWSE PATH ↑

VALUE SOURCE

DATA TYPE

7. Once you have some instructions added, click **Save Changes**.

Device Properties

General

Setting	Description
Name	The name of this connection.
Description	Device connection description (optional). Can be used to provide any useful information or comments about this connection.
Enabled	If checked, the connection is enabled within the Gateway. If unchecked, the connection is disabled.
Simulator Program	Upon creation, it is required to select a program, which eliminates the need to load a program after the connection is running. If you do not know what program you want to load, you can select None. Programs can be loaded and changed after creation using the Edit Program panel. The Generic Program is selected by default.

Simulator Program Settings

Setting	Description
Repeat Program	If checked, 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.
Legacy Mode	When Legacy Mode is enabled, simulators created in previous versions of Ignition can be imported without having to recreate tags. See the Preloaded Programs prefix note for more information.

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 Browse 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 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) 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 are fully 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 described 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. 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 import 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 previous versions 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 writable.

Import

The Import 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, and 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 <code>True</code> will stop the program at its current Time Interval. Setting it back to <code>False</code> 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 <code>False</code> .

Export Instruction File

The ability to create custom instructions in the driver provides 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. **Delete All** removes all instructions in the program.

×

GenSim Program

Program

Gen...

▼

Load Simulator Program

Import

📄

Export

📄

Delete All

🗑️

Add

+

TIME INTERVAL	BROWSE PATH	VALUE SOURCE	DATA TYPE	
0	Ramp/Ramp0	ramp(0.0, 10.0, 100.0, true)	Double ▼	Delete
0	Ramp/Ramp1	ramp(5.0, 123.0, 230.0, true)	Double ▼	Delete
0	Ramp/Ramp2	ramp(1.0, 2.0, 50.0, true)	Double ▼	Delete
0	Ramp/Ramp3	ramp(-10.0, 10.0, 300.0, true)	Double ▼	Delete
0	Ramp/Ramp4	ramp(0.0, 500.0, 1500.0, true)	Double ▼	Delete
0	Ramp/Ramp5	ramp(0.0, 700.0, 2000.0, true)	Double ▼	Delete

BACnet

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 **Platform > System > Modules** page.

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

Ignition 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 properties described in the table below.

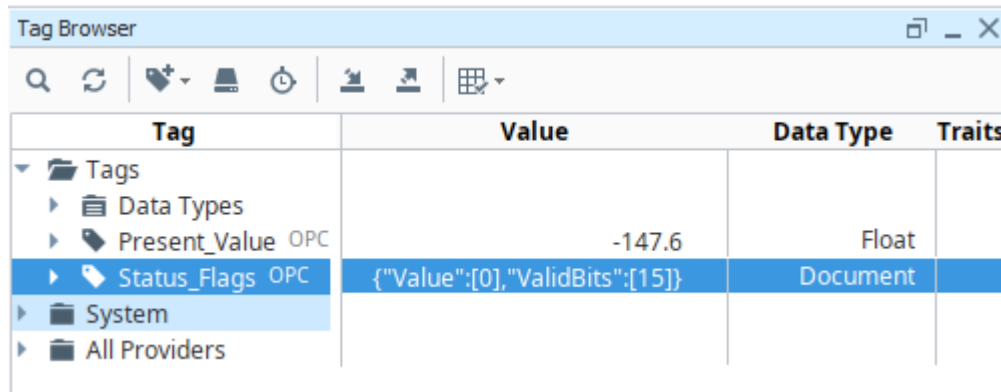
Name	Type	Read/Write
Elapsed_Active_Time	Long	Writable
Time_Of_Active_Time_Reset	Document	Read Only

Object Identifier

The numerical identifier of the object.

Status_Flags and Document Tags

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



The screenshot shows a 'Tag Browser' window with a tree view on the left and a table on the right. The tree view shows a hierarchy: Tags > Data Types > Present_Value OPC > Status_Flags OPC. The table has four columns: Tag, Value, Data Type, and Traits. The row for Status_Flags OPC is highlighted, showing its value as a JSON object and its data type as Document.

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

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

The ([system.bacnet.readRaw](#) and [system.bacnet.writeRaw](#)) scripting functions 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

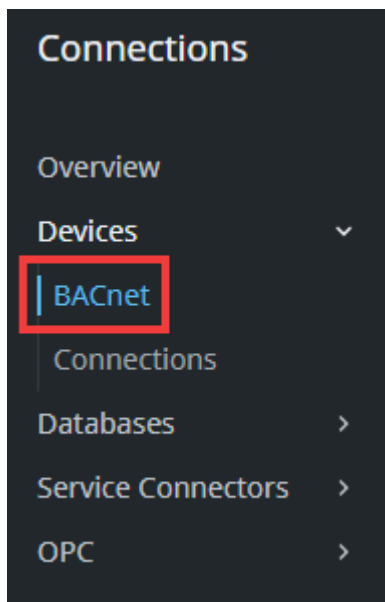
One strategy for configuring local devices is to configure one per network adapter that will be used for communication between the Ignition Gateway and remote BACnet devices.

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.

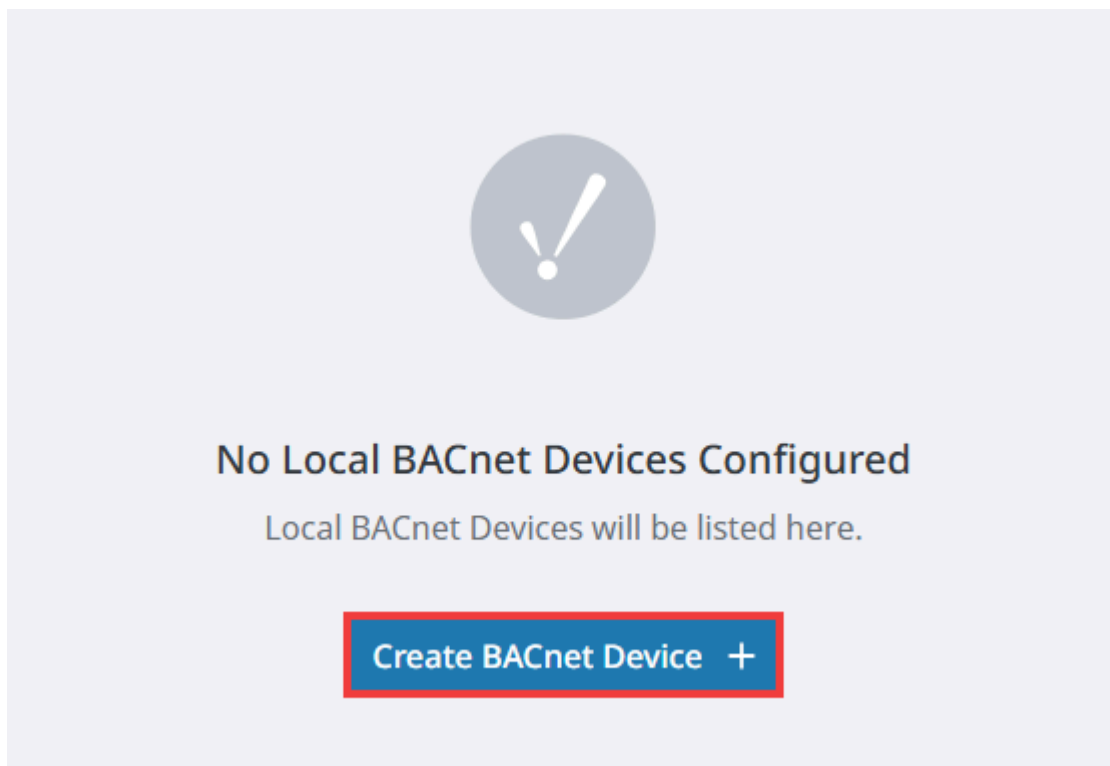
Another situation where you may need multiple local devices is when registration as foreign device with a BACnet Broadcast Management Device (BBMD) is necessary to bridge traffic between networks. Each local device instance can only be registered with one BBMD.

Configure a Local BACnet Device

1. On the Gateway webpage, go to **Connections > Devices > BACnet**.



2. Click on **Create BACnet Device**.



3. 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 47808.
- Broadcast Address: Enter the **IP address**, for example 10.10.###.##

4. Leave the default values in the remaining fields.

5. Click **Create BACnet Device**. Your local BACnet device should now be ready to use.

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

Device Settings

General

Local Device Setting	Description
Name	Device name.
Enabled	Enable or disable the BACnet device. Default is enabled.
Bind Address	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 47808.

Advanced

Ignition Redundancy Setting	Description
Backup Device Number	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 0.

BACnet Remote Device

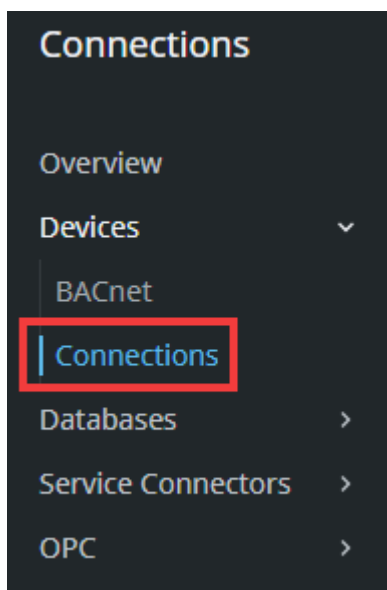
Once a **local device** has been configured a remote BACnet device can be added. You must know the IP address and device number of the remote device to configure a connection.

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 three dots menu for a given device (**Connections > Devices > Connections**) in the Ignition Gateway.

Connect to a Remote BACnet Device

1. On the Gateway webpage, go to **Connections > Devices > Connections**.



2. Click on **Create Device Connection**.
3. Select **BACnet/IP** and click **Next**.

Create Device Connection

Create Device Connection

1. Select Driver

2. Configure Device

☐ **Allen-Bradley Logix Driver**
Connect to Allen-Bradley Logix family devices. Optimized for devices with firmware v21+ but supports earlier firmware versions with significantly reduced performance.

☐ **Allen-Bradley Micro800 Driver**
Connect to Micro800 series PLCs

☐ **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.

☒ **BACnet/IP**
Connect to devices that implement BACnet/IP.

☐ **DNP3 Driver**

Cancel

Next

4. Fill in the following fields:

- Name: **BACnet Remote**
- Local Device: Select the local device from the dropdown. If no local devices are shown, you'll need to **set one up** first.
- Remote Address: Enter the **IP address**, for example 10.10.###.##
- Remote Device Number: The instance number of the remote device you're setting up.

Create Device Connection

Create Device Connection

1. Select Driver

2. Configure Device

GENERAL

*Required

Name *

BACnet Remote

The name of this device connection.

Description

Enabled

☒ Set whether this bacnet device is enabled or disabled within the Gateway.

CONNECTIVITY

*Required

Local Device *

Select

BACnet Local

Remote Address

Back

Create Device Connection

5. Leave the default values in the remaining fields.

6. Click **Create Device Connection**.

The Device Connections page is displayed, showing the **BACnet/IP** device is successfully created and added to Ignition.

Device Settings

General

Properties	Description
Name	Device name.
Description	Device description.
Enabled	Enable or disable the BACnet device. Default is enabled.

Connectivity

Properties	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 47808.
Remote Device Number	The instance number of the remote device you're setting up. Default is 1.
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	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	Duration, in seconds, to wait for the remote device discovery process to complete. Default is 5.
Write Priority	The priority to use when writing to commandable properties. Default is 8.

IEC 61850 Driver

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

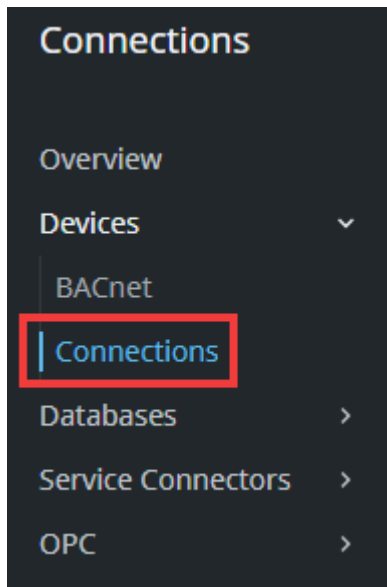
[Watch the video](#)

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 **Platform > System > Modules** page.

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.

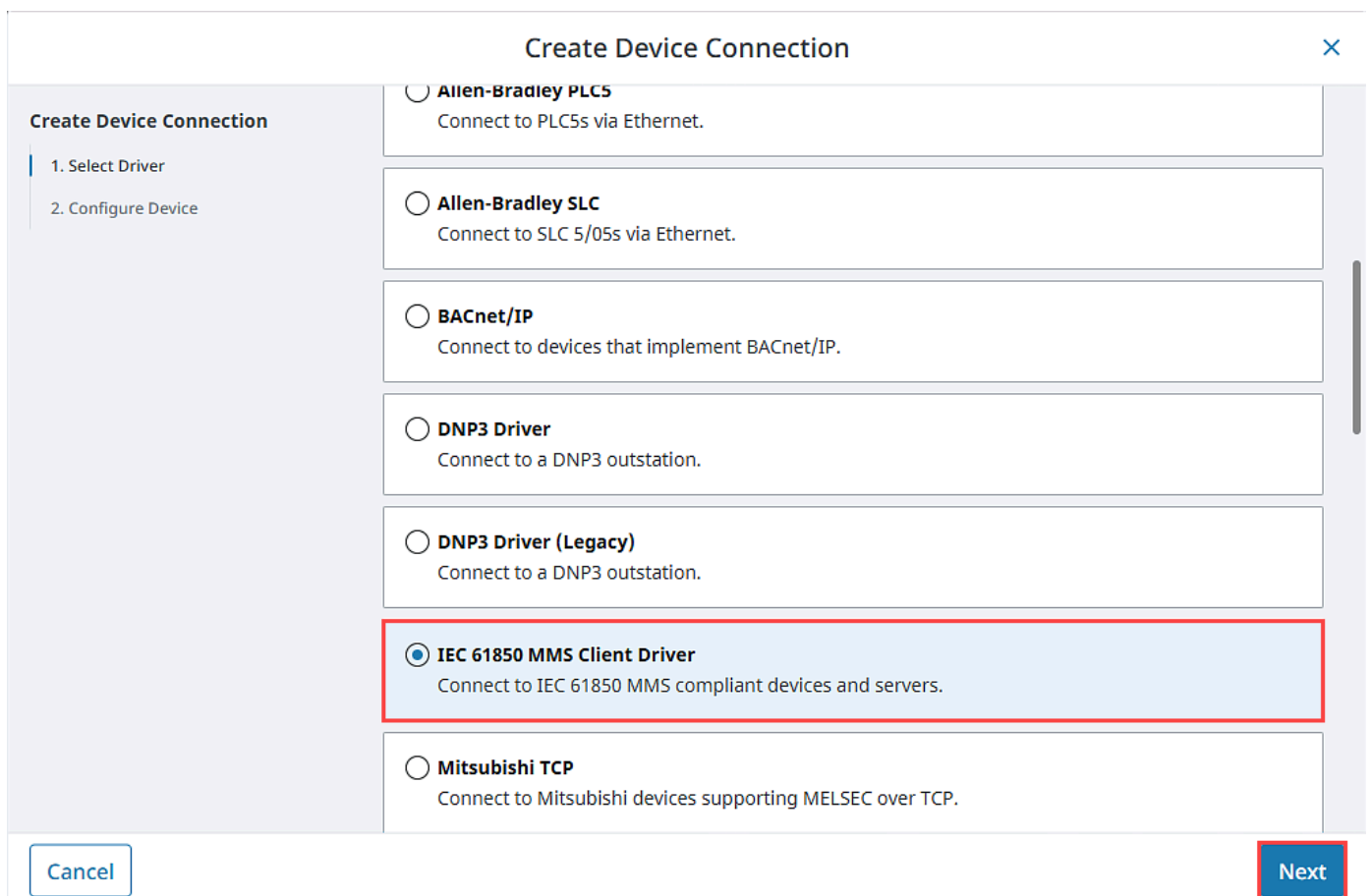
1. Go to the **Connections** section of the Gateway Webpage.

2. Select **Devices > Connections**.



3. Click on **Create Device Connection**.

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



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

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 Device Connection**. The **Device 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, **test** in this example, and expand the three dots menu and select **Manage SDL File**.

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.

The screenshot shows the 'Device Connections' interface. On the left, a sidebar lists 'Overview', 'Databases', 'Devices', 'Connections' (selected), and 'Settings'. The main panel is titled 'Device Connections' and contains a 'Connections' section. This section has a search bar, a 'View' dropdown set to '20', and a 'Create Device Connection' button. Below this is a table with columns: NAME, DRIVER, ENABLED, STATUS, and a settings icon. The table contains one entry: 'test' with driver 'com.inductiveautomation.Iec 61850DeviceType', enabled 'true', and status 'CONNECTING'. A context menu is open for the 'test' entry, showing options: Edit, Rename, Duplicate, Manage SDL File (highlighted with a red box), Disable, and Delete.

NAME	DRIVER	ENABLED	STATUS
test	com.inductiveautomation.Iec 61850DeviceType	true	CONNECTING

9. Locate your SCD file and drag it into the upload field or click the link to upload your file directly.
10. Click **Next**.

Manage SCD File

Manage SCD File

1. Upload SCD File

2. Select IED

Upload SCD File

Upload an SCD File to proceed. After the upload, you will select an IED to load.


SEL3505_SIM.scd
195.7 KB
[Remove](#)

Cancel

Next

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

Manage SCD File

Manage SCD File

1. Upload SCD File

2. Select IED

Select IED

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

☒ SUBA_DCU
Access Point: S1
Sub Network Name: W01

Back

Save Changes

12. Click **Save Changes**.

! INFO

The Edit panel will now show the saved SCD file 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.

SCD FILE SETTING

Use SCD File
☒ Determines if a SCD file upload will be used when requesting from the device.

Use Configured Hostname
☐ Override any hostname that may have been set by the SCD file with the configured hostname.

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

IED Name

Access Point Name

Use the OPC Quick Client to Confirm Report Statuses

Once you have established a device connection, you may want to verify report availability before opening the Designer. The **OPC Quick Client** provides a convenient way to inspect IEC 61850 report statuses and troubleshoot report configuration directly from the Gateway at **Connections > OPC > Quick Client**. From there, browse the **Ignition OPC UA Server** and locate your IED under the **Devices** folder, where configured IEC 61850 device data is organized into the following folders:

- **Model:** Contains logical nodes that can be directly polled for information.
- **Reports:** Contains a folder for each report control block, and exposes report-related attributes that can be subscribed to.
- **Diagnostics:** Contains diagnostic information that can help identify the status of various aspects of the configured device.

i NOTE

Subscribing to any report control block attribute causes Ignition to attempt to enable the associated report using the **RptEna** attribute. If a report cannot be enabled, the subscription will return a quality of `Bad_OutOfService: The source of the data is not operational`.

If you are unsure if a report is enabled, you can use the **[R]** read action on the **RptEna** attribute within the appropriate report control block folder under Reports. A value of false indicates the report is not currently enabled, while a value of true indicates the report is already enabled elsewhere and cannot be enabled again.

OPC Quick Client

✓ Read Complete

Server: [Ignition OPC UA Server]ns=1;s=[testDevice]SUBA_DCUCFG/LLN0.BR_SUBA_DCU_52FAxx_PMS1.RptEna

Value: false

Quality: Good

Timestamp: 24Dec2025 12:55:46

Quick Client

NAME	TYPE
Reports	Object
BR_SUBA_DCU_52FAxx_PMS1	Object
PurgeBuf	Tag [S] [R] [W]
RptEna	Tag [S] [R] [W]
RptID	Tag [S] [R] [W]
SqNum	Tag [S] [R] [W]

⚠ CAUTION

The IEC 61850 standard permits very few write operations, and write actions should not be performed from the OPC Quick Client. See [Device Operations](#) for recommended practices when writing values 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.

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.
Password	The password selections available for the device connection. Options include None, Embedded, and Referenced. See the Secrets Management page for more information on using Referenced passwords.

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.

NOTE

If you are using a buffered report, be aware the config.idb file will be frequently updated with the last report_id received. The autobackup default time, set by the `localdb.autobackup.delay` property, is 60 minutes. 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:

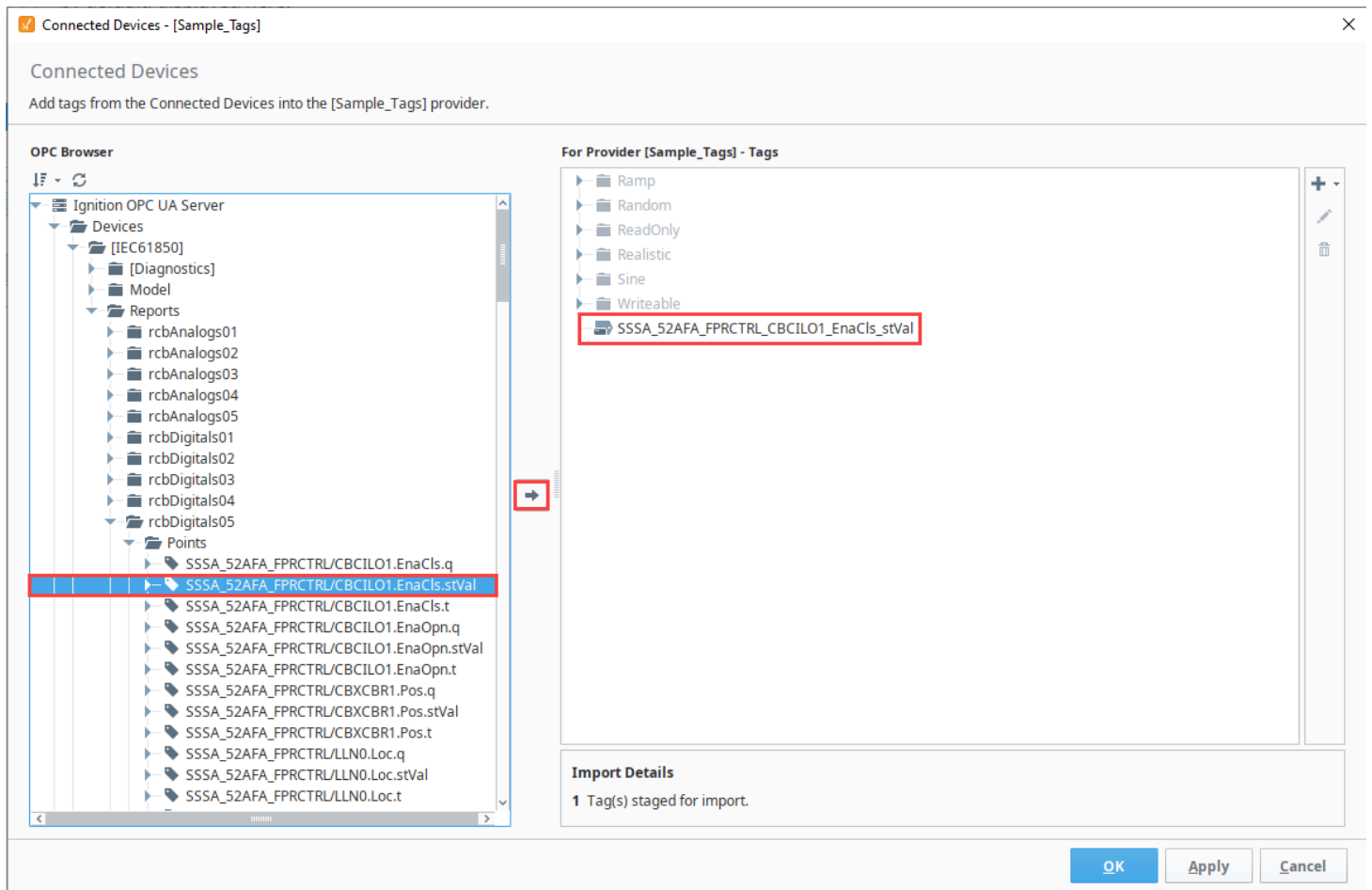
```
<entry key="localdb.autobackup.delay">60</entry>
```

Changed in 8.3.8 Autobackups are now disabled by default. This means that changes to the `localdb.autobackup.delay` property will only apply when autobackups are enabled. Automatic backups are handled with `localdb.autobackup.count`.

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

FUNCTIONAL CONSTRAINTS IN OPC ADDRESSES

Changed in 8.3.9 An attribute that is defined under more than one Functional Constraint (FC), such as a settable attribute exposed under **SP**, **SE**, and **SG**, is now browsed as a separate OPC node for each FC, rather than being collapsed into a single node. These colliding attributes are addressed with an **[FC]** suffix appended to the attribute, and reads and writes are directed to the FC named in the address:

```
[SSSA_52AFA_FPR]LD0/LLN0.rcbDigitalS05$SSSA_52AFA_FPRCTRL/CBCILO1.EnaCls.stVal[S
```

On IEDs where these collisions exist, upgrading replaces the previous unsuffixed NodeIds with the **[FC]**-suffixed form. Existing tags addressed to the unsuffixed NodeId will report a quality of **Bad_NodeIdUnknown** after the upgrade, and must be re-addressed to the appropriate suffixed form. Attributes that are defined under only one FC are unaffected and keep their existing addresses.

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

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
- F (FX3)

Connecting to a Mitsubishi Device



INDUCTIVE
UNIVERSITY

**Connecting To Mitsubishi
Devices**

[Watch the video](#)

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 **Platform > System > Modules** page.

Before you can connect your Mitsubishi device to this driver, your GX Works software settings must also 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 **Connections > Devices > Connections** on the Gateway.
2. On the Device Connections page, click **Create Device Connection**.
3. Select **Mitsubishi TCP** and click **Next**.
4. Fill in the Name field. Entering a description is optional.

5. Enter the device hostname in the **Hostname** field.
6. Enter the device port in the **Port** field.
7. Enter the **Local Address** of the network adapter to connect from if you have multiple network cards and you need to specify the particular network card that has PLC access.
8. Select the MELSEC series you will be connecting to using the **Series** dropdown under Device Settings. 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.

Create Device Connection

Create Device Connection

1. Select Driver

2. Configure Device

DEVICE SETTINGS

*Required

Series

iQR

If disabled all registers will be read in individual read requests. Disable with caution.

PC Number *

255

The network module station number of an access target.

Network Number *

0

The network number of an access target.

Request Destination Module I/O Number *

1023

The start input/output number of a multi-drop connection source module, or the CPU module of a multiple CPU system.

Request Destination Module Station Number *

0

The station number of an access target module when accessing a multi-drop connection station.

Back

Create Device Connection

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	The name of the device connection.
Description	The optional device description.
Enabled	Determines whether this device connection is enabled or disabled within the Gateway.

Connectivity

Property	Description
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.

Device Settings

Property	Description
Series	The series of the Mitsubishi device. Default is iQR. If disabled all registers will be read in individual read requests.
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.

Request Optimization

Property	Description
Max Gap Size	Changed in 8.3.2 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. Any negative value will use the max request size for the request type, optimizing for the fewest requests. Default is -1.
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.

Adding Addresses

You can add addresses through a few different options on the Gateway or in the Designer.

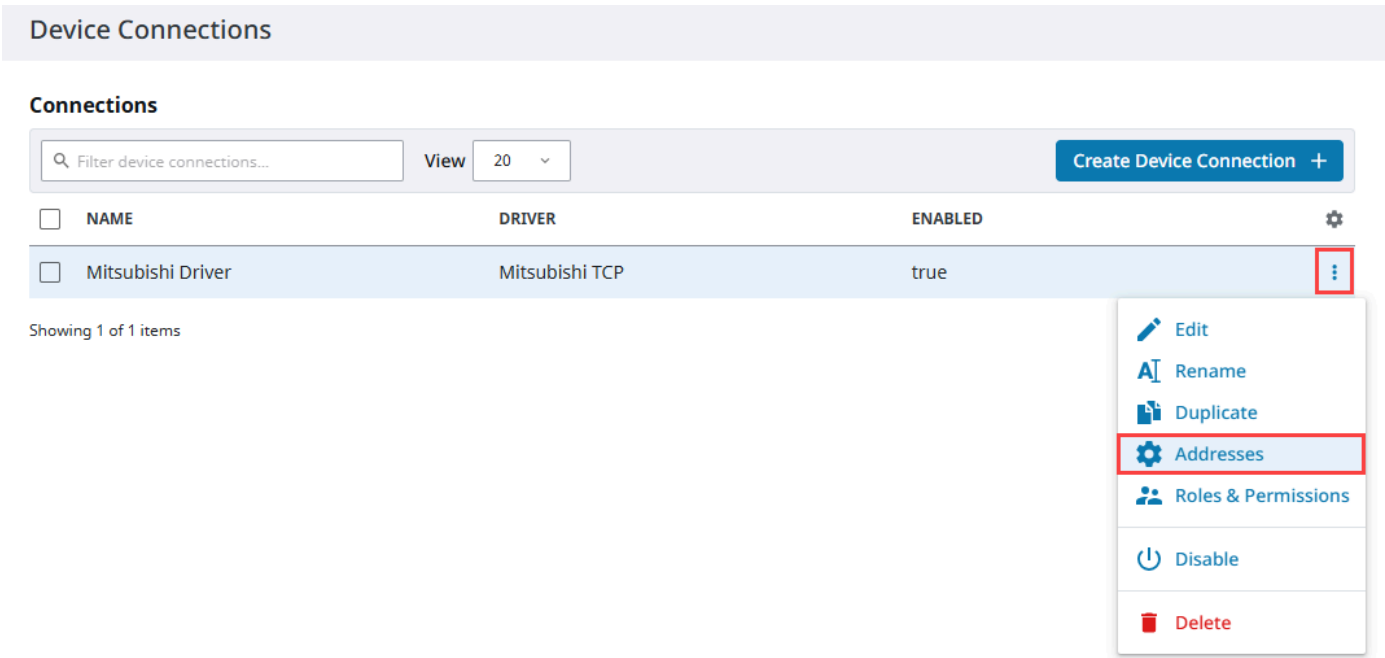
Addressing on the Gateway

The Mitsubishi Addresses panel 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 panel 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 **Connections > Devices > Connections**.
2. Expand the three dots menu on your Mitsubishi device and choose **Addresses**.



Adding Individual Address Rows

To manually enter addresses:

1. Click **Add +** to populate rows for address information.

X

Mitsubishi Driver Program

Import Configuration File

Reset

Export

Remove All

Add

BROWSE PATH	ADDRESS	DESCRIPTION	
			Delete

Showing 1 of 1 items

Cancel

Save Changes

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 Changes** when you are finished.

Importing and Exporting

To import a CSV file:

1. Select **Import Configuration File** on the Addresses panel.
2. Drag your CSV file onto the Import popup or select **click to upload** to locate the CSV file you want to upload and select it to open.

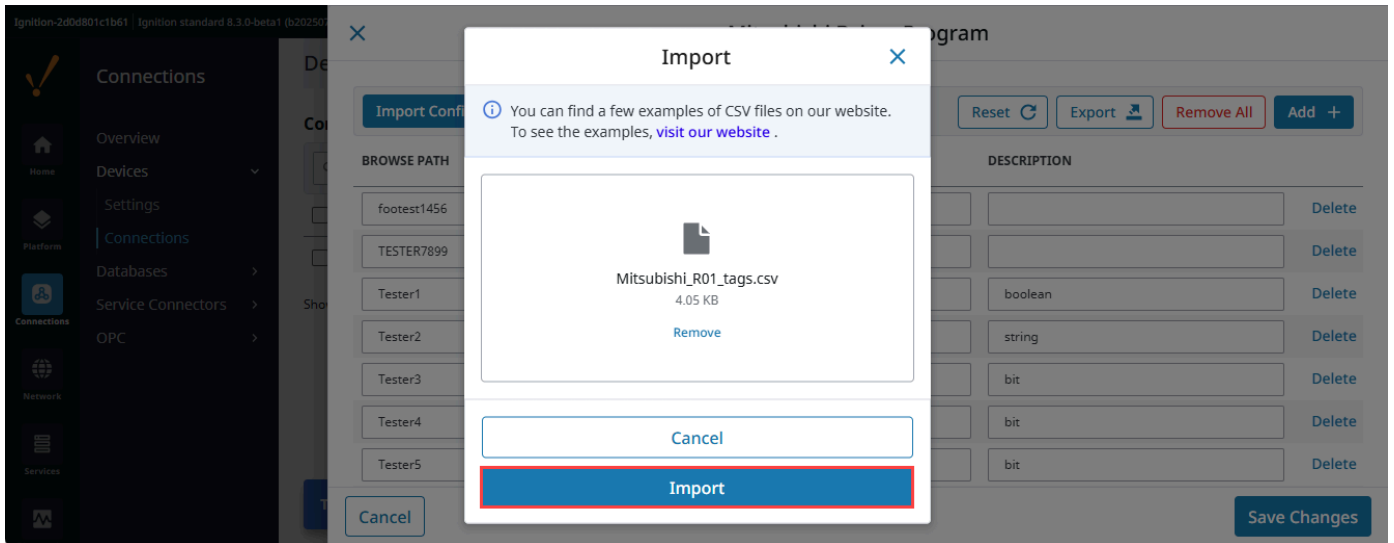
CAUTION

Make sure a comma separates column information, and if your Description column includes commas, use quotes to prevent misinterpretations when uploading.

3. Click **Import**.

⚠ 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.



4. Click **Save Changes**.

To export the address list, use the **Export** button to download.

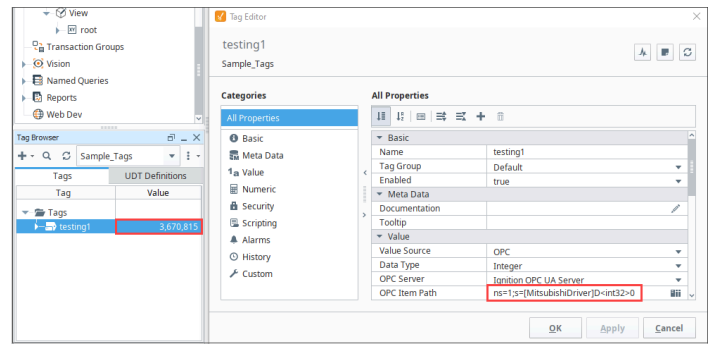
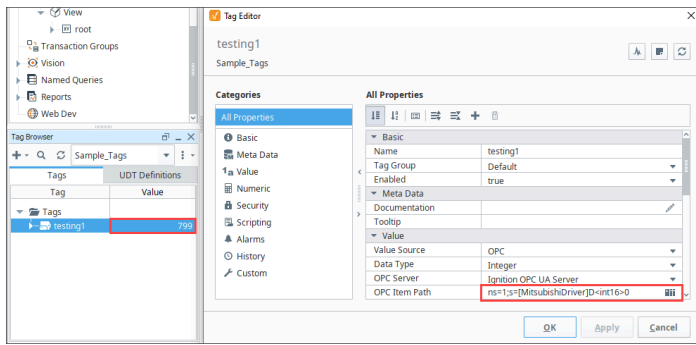
Addressing in the Designer

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	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	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
Retentive timer (Coil)	STC	Bit

Device Name	Keyword	Type
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
File register (Serial number access method)	ZR	Word
Refresh data register*	RD	Word

*Not available for Q/L series

MELSEC-F

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
- String (to use for ASCII characters)
- 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`.

<code>D[0][0] = D<int16>0</code>	<code>D[0][1] = D<int16>1</code>	<code>D[0][2] = D<int16>2</code>
<code>D[1][0] = D<int16>3</code>	<code>D[1][1] = D<int16>4</code>	<code>D[1][2] = D<int16>5</code>
<code>D[2][0] = D<int16>6</code>	<code>D[2][1] = D<int16>7</code>	<code>D[2][2] = D<int16>8</code>

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