OPC UA

OPC Specification

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

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

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

OPC UA

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

OPC UA is the latest revision of the OPC specification, which offers platform and vendor neutral transfer and use of industrial data. The specification plays a crucial role in Ignition, and is the primary data access specification used in the Gateway. Ignition supports connections to any number of OPC UA servers created by any manufacturer, provided that they are compliant to the specification. The data is then used to drive all aspects of the system. Creating connections to OPC UA servers is described below in 26020900.

Distributed Systems with OPC UA

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

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

Servers and Clients

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

On this page

...

Steve Hechtman's Blog

Click the link below to see the blog:

OPC UA goes mainstream

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

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

Technology

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

Additionally, Ignition supports all of the common encryption schemes. This means that Ignition connects to OPC UA servers (and allows connections from clients) over TCP/IP using encryption, and sends data by first encoding it into an efficient format defined by the OPC UA specification. This is in contrast to other schemes outlined in the specification, which can use web services and XML encoding that are not as efficient.

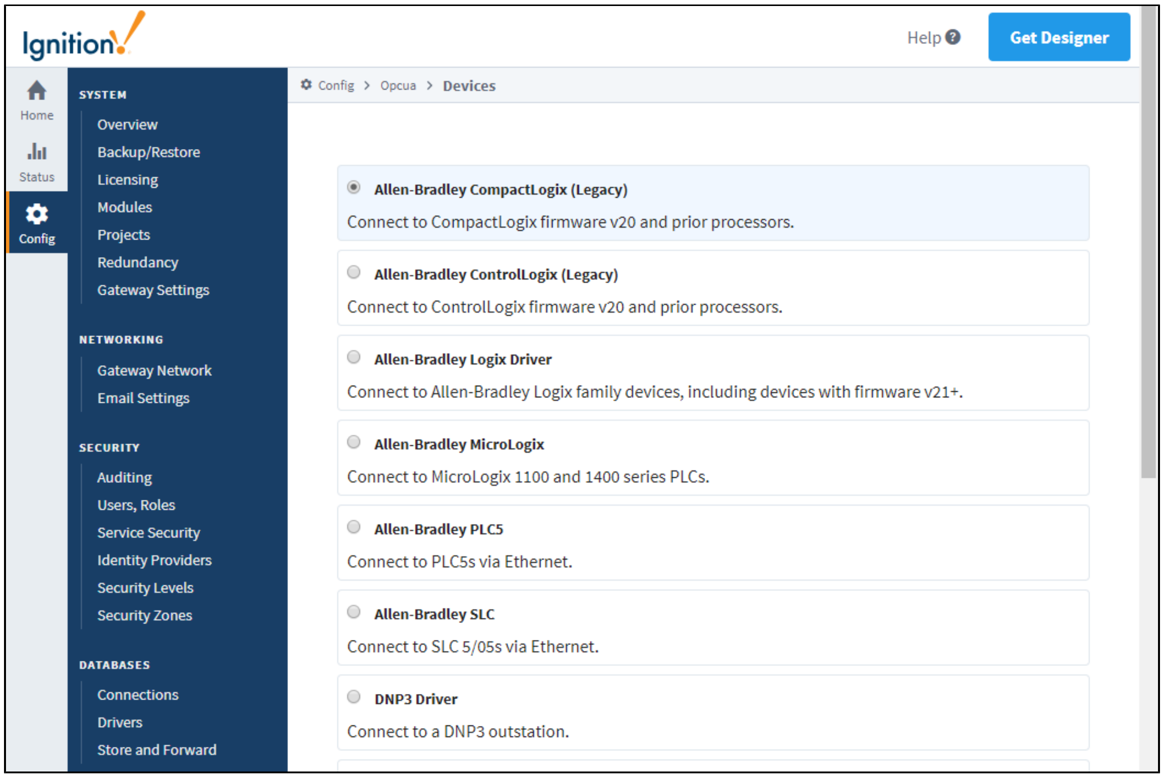
Connecting with OPC UA

Connecting to a Device

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

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

- Allen Bradley Ethernet
- Modbus
- Siemens
- UDP and TCP Driver
- DNP3
- Omron NJ Driver
- Simulators



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

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

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

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

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

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

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

Connecting to a Server

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

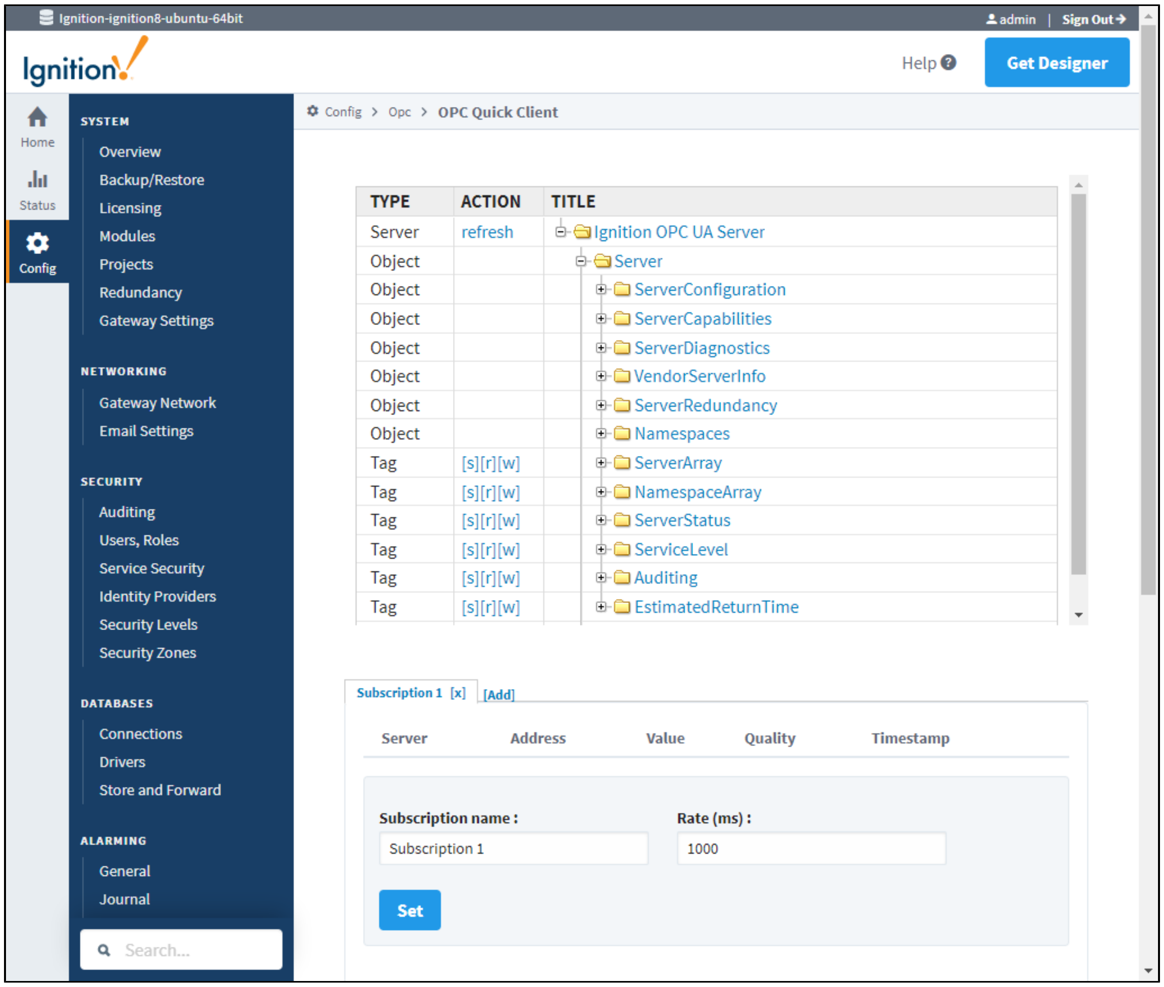
- Third Party OPC Servers
- OPC COM

OPC Quick Client

After you connected to a device, you can access the OPC Quick Client in the Gateway. It allows for quick, simple testing of any devices connected to the server.

You can find the Quick Client under the **OPC Connections** section of the Ignition Gateway **Config** tab. You can browse by expanding the tree nodes and read from or write to OPC tags by clicking on the [r] and [w] buttons next to those tags.

Subscriptions can be made by clicking on the [s] button. Clicking on the **enable live values** link will automatically refresh subscriptions and show live value changes (if there are any).



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

Server	Address	Value	Quality	Timestamp

Subscription name : Subscription 1

Rate (ms) : 1000

How Do I Get Data from My PLC?

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

1. Add a device, see 26020900.
2. Add some tags, see Creating Tags.

It requires you to touch both the Ignition Gateway and the Ignition Designer. There are also some limitations as to what kind of devices you can connect to Ignition and these are explained throughout the user manual, however, included below is an overview of what you can expect when it comes to compatibility.

Brief Summary of Device Connection in Ignition

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

Adding a Device to Ignition

Ignition Supported OPC UA Device

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

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

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

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

Adding Connection to Third Party OPC Server Via OPC UA

If your device does not have an Ignition driver, you can use a third party OPC server to connect to your device and then have Ignition connect to the server as a client. If the OPC server offers support for OPC UA, you can add a new OPC UA server connection in the Ignition Gateway. For more information, see the OPC UA Client Connection Settings page. Additionally, the Connecting to Kepware OPC UA is useful when connecting to Kepserver.

Adding Connection to Third Party OPC Server Via OPC COM

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

Related Topics ...

- Quick Start Guide

In This Section ...

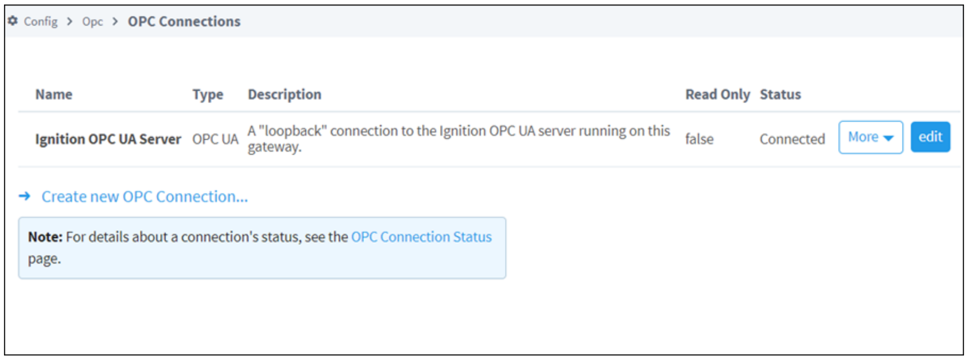
OPC UA Client Connection Settings

Configuring an OPC Client Connection

An OPC UA connection is used to communicate with an OPC UA compliant server, such as the one the OPC UA Module provides.
The following steps walk through connecting Ignition (as an OPC UA client) to a OPC UA server.

On this page ...

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



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

Note: For details about a connection's status, see the OPC Connection Status page.

3. Select **OPC UA Connection** from the list and click **Next**. The Endpoint 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:

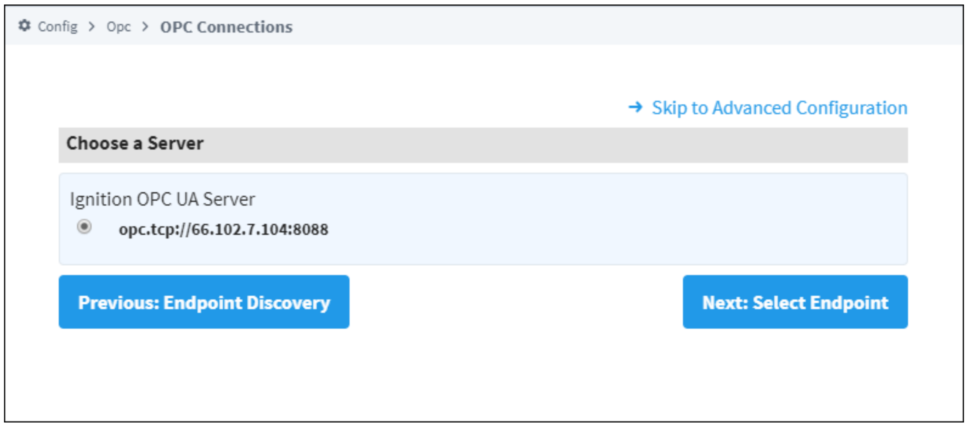
```
opc.tcp://IpAddress:Port
```

Instead of an IP address, a host name can be used:

```
 opc.tcp://myServer:12345
```

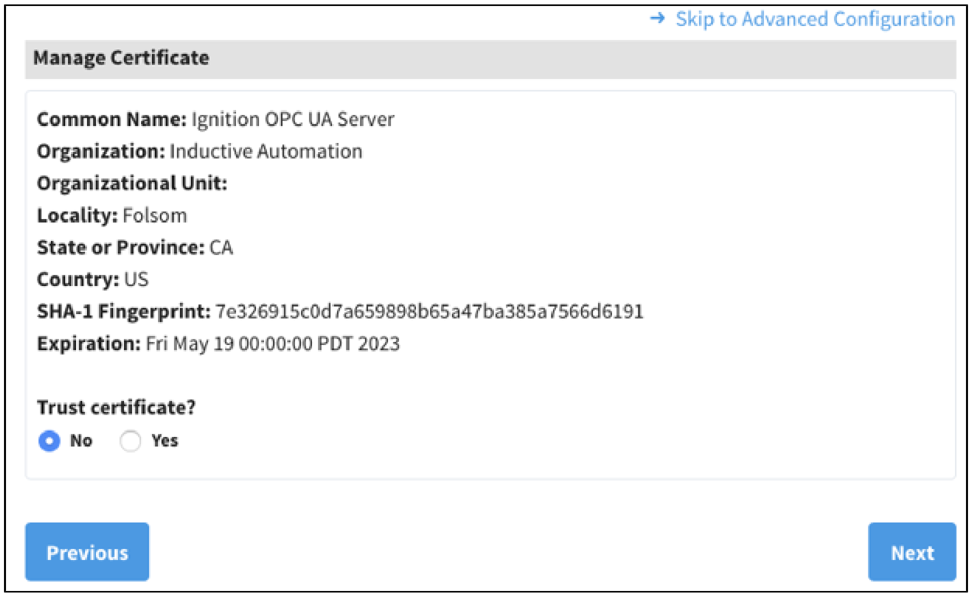
An "Advanced Configuration" link was added to this flow, allowing you to manually configure connection settings. This is useful in cases where a server doesn't allow anonymous endpoint access, but provides separate discovery endpoints.

5. Choose a server, then click **Next: Select Endpoint.**


Config > Opc > OPC Connections
→ Skip to Advanced Configuration
Choose a Server
Ignition OPC UA Server
opc.tcp://66.102.7.104:8088
Previous: Endpoint Discovery
Next: Select Endpoint

6. Click **Next: Select Endpoint**. A list of available Security Policies and Message Security options will appear.

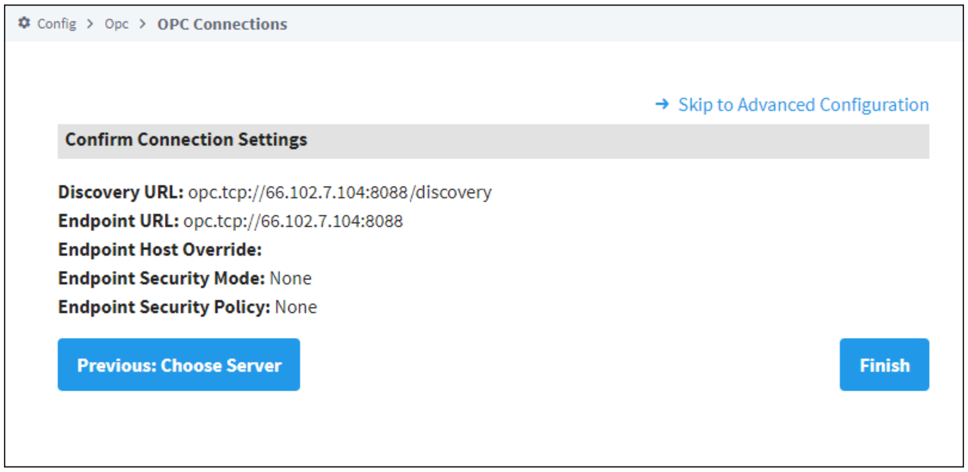
7. A Manage Certificate window will open if you haven't previously trusted a certificate. If the certificate that the Endpoint sends you to is already in Ignition's trust store, this step is skipped. Click **Next**.



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

Trust certificate?
[✓ No] Yes

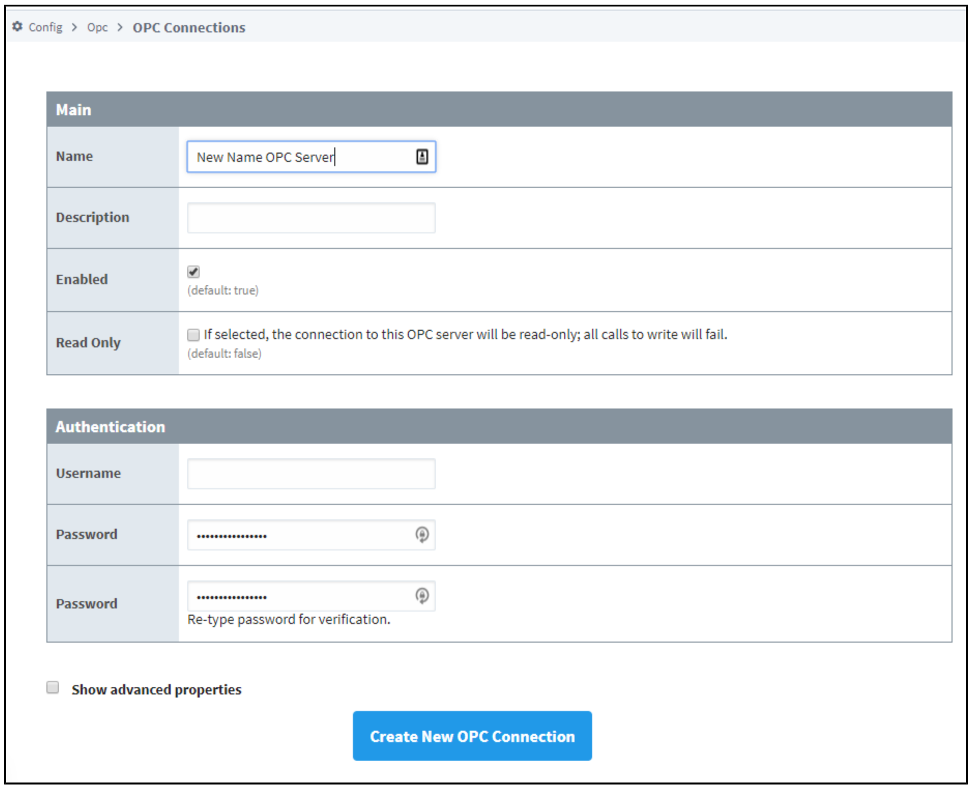
8. If you entered a discovery URL in step 4, you also have an option to enter another URL if the host is unreachable.
9. Select a Security Policy and Message Security configuration to use when connecting to the endpoint, then click **Finish**. The policies that appear here are determined by the server.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


Config > Opc > OPC Connections
Skip to Advanced Configuration
Confirm Connection Settings

Discovery URL: opc.tcp://66.102.7.104:8088/discovery
Endpoint URL: opc.tcp://66.102.7.104:8088
Endpoint Host Override:
Endpoint Security Mode: None
Endpoint Security Policy: None

Previous: Choose Server
Finish

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



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

Authentication	
Username	
Password	••••••••••••••••
Password	•••••••••••••••• Re-type password for verification.

☐ Show advanced properties

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

Ignition is now connected to the OPC UA server.

OPC UA Client Connection Settings

The following table describes all the available properties.

Main	
Name	A name used to identify this connection.

Description	Short description of this connection.
Enabled	Disable the connection to the OPC server.
Read-only	Puts the connection into read-only mode. All writes sent to this server will fail.
Authentication	
Username	A username the connection will use when authenticating with the UA server.
Password Fields	The password to use when authenticating with the UA server.
Advanced	
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 120,000.
Session Timeo ut	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 a References is simply 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 increasing 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 Publish 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.
Max Message Size	The maximum allowable size of an OPC UA application layer message. Default is 33,554,432.
Max Array Length	The maximum allowable size for arrays. Default is 2,147,483,647.
Max String Length	The maximum allowable size for strings. Default is 2,147,483,647.
Type Dictionary Fragment Size	The fragment size to request when reading the server's type dictionary. Default is 8,192.
Keep-Alive Failures Allowed	Number of consecutive failures allowed before disconnecting. Setting this to <= 0 means consecutive failures will not cause a disconnect. Default is 1.
Keep-Alive Interval	Interval, in milliseconds, between keep-alive requests.
Keep-Alive Timeout	Max duration, in milliseconds, to wait for a response to a keep-alive request. Default is 10,000.

Browser Origin	The Node that browsing should originate from. Options are OBJECTS_FOLDER or ROOT_FOLDER. Most OPC UA Servers use OBJECTS_FOLDER, but some non-standard servers may require ROOT_FOLDER to browse correctly. Ignition's OPC UA Servers uses OBJECTS_FOLDER.
Failover	
Failover Enabled	Enable failover on the connection, allowing the UA client to switch to a backup server in the event the primary server is unavailable.
Failover Threshold	The number of retry attempts before the failover connection is used. The default is 3.
Failover Discovery URL	The discovery URL for the backup server's OPC UA server. Expects the following format: `opc.tcp://hostname:port`
Failover Endpoint URL	The endpoint of the failover server. Example: `opc.tcp://192.168.1.0:62541`
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: 192.168.1.10
KeyStore	
KeyStore Alias	The alias of the certificate and private key stored in the client KeyStore.
Password Fields	The password to use when authenticating with the UA server.

Failover Versus Backup Properties

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

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

Related Topics ...

- OPC UA Server Settings
- Understanding Tags

OPC UA Server Settings

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 3rd party systems.

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

On this page

...

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 connection require the following credentials:

Username	opcuauser
Password	password

Note that 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 UR should include "/discovery" at the end:

```
opc.tcp://192.168.2.134:62541/discovery
```

OPC UA Server Settings

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

The table below represents settings on Ignition's OPC UA server. They'll only become available if the OPC UA Module is installed on the gateway.

Setting	Description	Default value
Endpoint Configuration		
Bind Port	The port the UA server will bind to.	`62,541`
Bind Addresses	The address the server will bind to. If you want to expose the OPC UA server to external sources, you need to use 0.0.0.0 or the IP address of the computer.	`localhost`

Endpoint Addresses	A comma separated list of endpoint addresses that the UA server can be reached at. It is important that this is set to addresses that can be reached by any UA clients attempting to connect to the server.
	
	When entering addresses into this property, they can be just an IP address or hostname:
	
	`10.10.10.100`
	
	Alternatively, angled brackets can be used. When applied to an address, the server attempts to find the hostname, or resolve the value to as many addresses or hostnames as it can find.
	
	`<10.10.10.100>`
		`<hostname>, <localhost>`

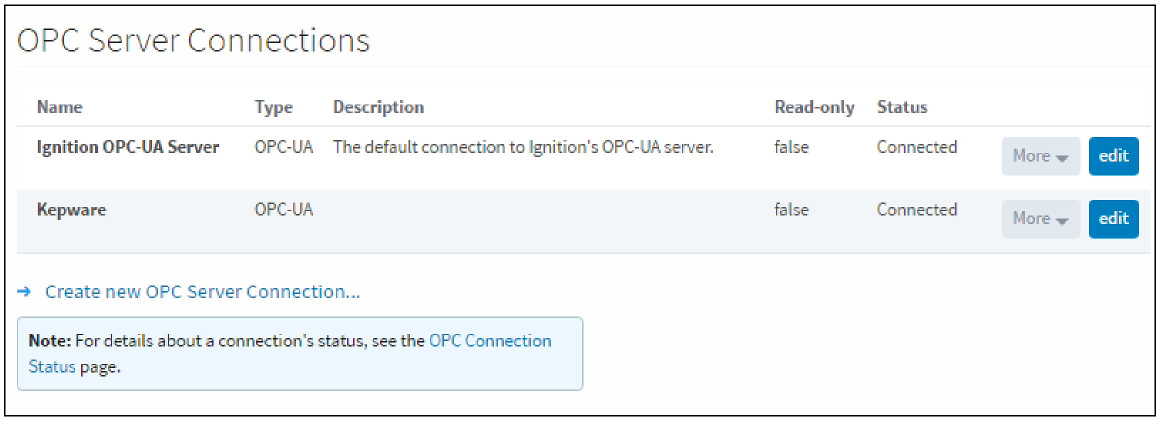
Security Policies	A comma separated list of acceptable security policies. Available policies are: • `None` • `Basic256Sha256` • `Aes128_Sha256_RsaOaep` • `Aes256_Sha256_RsaPss` In addition, the following deprecated policies are available, but not recommended: • `Basic128Rsa15 (deprecated)` • `Basic256 (deprecated)`	`Basic256Sha256`
Authentication		
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		
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`
Redundancy		
Backup Bind Addresses	The local addresses that the UA server will attempt to bind to while the backup server in a redundant pair.	`localhost`
Backup Endpoint Addresses	The endpoint addresses that the UA server can be reached at while the server is configured as the backup in a redundant pair. The notation on this property is similar to the **Endpoint Addresses** property above, in that angled brackets can be used with each hostname and IP address.	`<hostname>, <localhost>`
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`

Related Topics ...

- OPC UA Client Connection Settings
- Tag Browser

Third Party OPC Servers

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



OPC Server Connections

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

Create new OPC Server Connection...

Note: For details about a connection's status, see the OPC Connection Status page.

OPC UA

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

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

OPC COM (DA)

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

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.

In This Section ...

Connecting to Kepware OPC UA

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

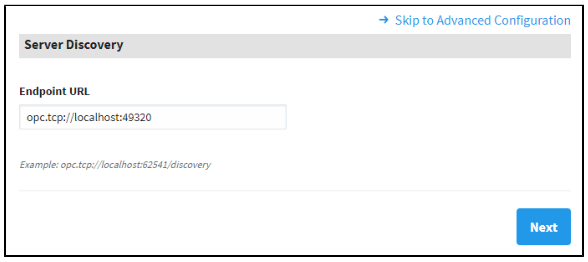
Connect to KEPServerEX from Ignition using OPC UA

1. In the **Config** section of the Gateway, go to **OPC Client > OPC Connections**. The OPC Connections page is displayed showing the OPC UA servers your Ignition is connected to.
2. Click on **Create new OPC Connection…**.
3. Choose **OPC UA** as the connection type, and click **Next**.

4. On the Server Discovery page,enter the endpoint of the OPC UA server Ignition should connect to. For example:

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

Click the **Next** button to continue.



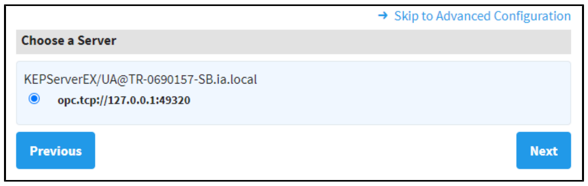
Endpoint URL: opc.tcp://localhost:49320

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

Connecting to Kepware OPC-UA

Watch the Video

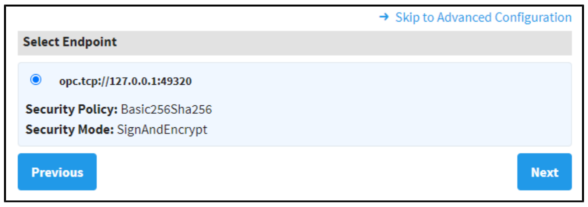
5. Select the Server you want and click **Next**.



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

● opc.tcp://127.0.0.1:49320

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



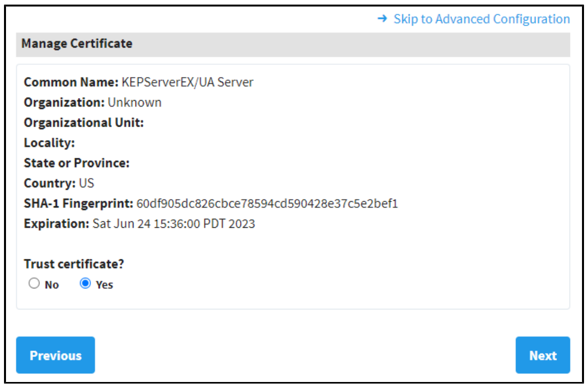
● opc.tcp://127.0.0.1:49320

Security Policy: Basic256Sha256

Security Mode: SignAndEncrypt

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

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



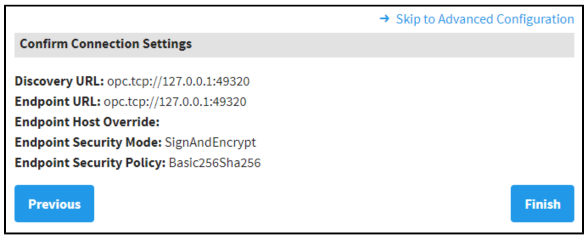
→ Skip to Advanced Configuration

Manage Certificate

Common Name: KEPServerEX/UA Server
Organization: Unknown
Organizational Unit:
Locality:
State or Province:
Country: US
SHA-1 Fingerprint: 60df905dc826cbce78594cd590428e37c5e2bef1
Expiration: Sat Jun 24 15:36:00 PDT 2023

Trust certificate?
No [✓ Yes]

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



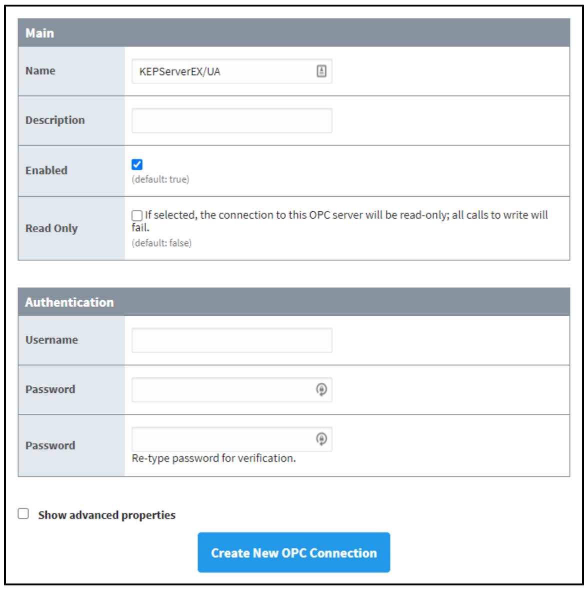
→ Skip to Advanced Configuration

Confirm Connection Settings

Discovery URL: opc.tcp://127.0.0.1:49320
Endpoint URL: opc.tcp://127.0.0.1:49320
Endpoint Host Override:
Endpoint Security Mode: SignAndEncrypt
Endpoint Security Policy: Basic256Sha256

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

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



Main

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

Authentication

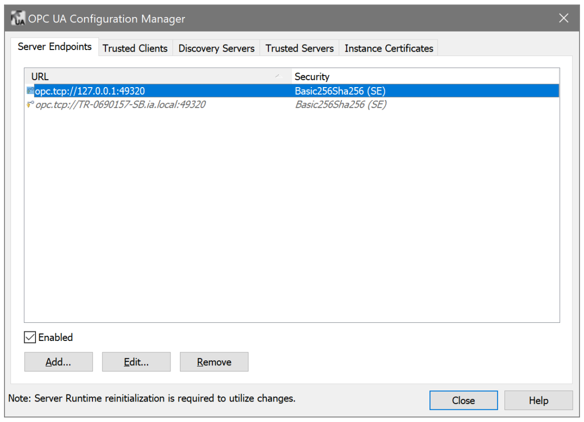
Username	
Password	
Password	Re-type password for verification.

☐ Show advanced properties

10. Click **Create New 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.

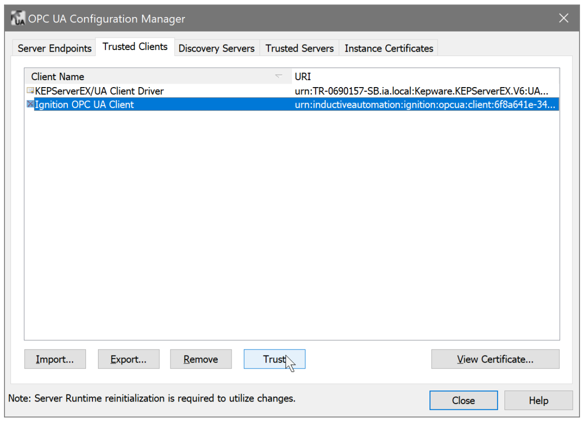
12. On the computer that KEPServer is installed on, right-click on the **KEPServerEX** icon on the desktop KEPServerEx is installed on, and from the menu select **OPC UA Configuration**. The **OPC UA Configuration Manager** will appear.



URL	Security
opc.tcp://127.0.0.1:49320	Basic256Sha256 (SE)
opc.tcp://TR-0690157-SB.ia.local:49320	Basic256Sha256 (SE)

Note: Server Runtime reinitialization is required to utilize changes.

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



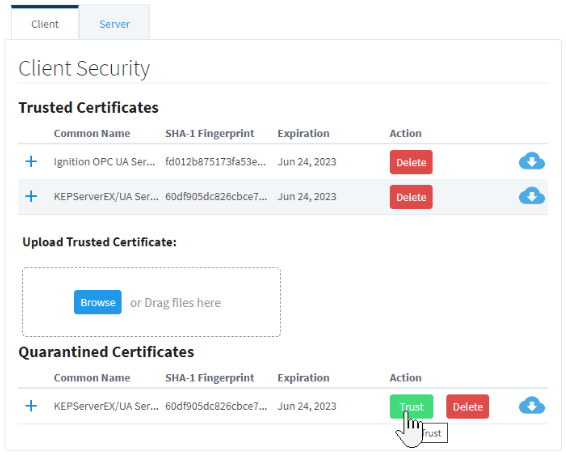
Client Name	URI
KEPServerEX/UA Client Driver	urn:TR-0690157-SB.ia.local:Kepware.KEPServerEX.V6:UA...
Ignition OPC UA Client	urn:inductiveautomation:ignition:opcua:client:6f8a641e-34...

Note: Server Runtime reinitialization is required to utilize changes.

15. Again, right-click on the **KEPServerEX** icon on the desktop KEPServerEx is installed on, and from the menu select **Reinitialize**.

16. Go back to the Ignition Gateway Webpage. In the **Config** section, go to the **OPC UA > Security** page.

17. Under the Client Security tab, you will find your new connection listed as Quarantined. Click on the **Trust** button on the far right.



Client Security

Trusted Certificates

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

Upload Trusted Certificate:

Quarantined Certificates

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

18. Go back to the **Config** section of the Gateway, to OPC Connections > Servers. The Status of your KEPServer connection should be **Connected**.

19. To test your tag connections, go to the OPC Connections > Quick Client in the Configure section of the Gateway. Expand the **KEPServer** object until you find tags.

Troubleshooting

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

Failover

The failover Kepware OPC UA server works the same as the OPC UA server with the exception that you need to have two copies of Kepware setup, preferably on different servers. The failover Kepware OPC UA server will be used in the event the primary Kepware server goes down. To enable failover, check the box to **Show advanced properties** in the New OPC UA Connection Settings, set the **Failover Enabled** property to **'true,'** and specify the **Failover Endpoint**.

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

For additional information on Failover, refer to OPC UA Client Connection Settings.

No Anonymous Token Policy Found

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

Other UA Servers

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

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

Download a Client Certificate Manually

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

Related Topics ...

- OPC COM

OPC COM

Connecting to OPC Classic (COM)

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

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

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

Install OPC Core Components

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

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

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

4. (Remote) Install Core Components on remote machine running the OPC-DA server.
   Once you have the correct package downloaded you can extract the contents of the .zip file and then run the installer. With the core components installed you can now proceed to setting up your OPC-DA server connection in Ignition.

Connecting to OPC-DA Server

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

Install OPC-DA Server Connection

1. Go to the Ignition Gateway **Config** section (http://localhost:8088/main/web/config).

2. Go to **OPC Connections > Servers** and then select **Create new OPC Server Connection...**.

3. Choose the **OPC-DA COM Connection** and then select whether you want to make a **Local** connection or if the OPC server resides on a **Remote** machine. For the most part, setting up a local or remote connection to an OPC-DA server is the same. There are only a couple of differences for a remote connection that will be highlighted along the way.

   Local - Selecting a local connection takes you to a screen that contains a list of the available and running OPC servers located on the local machine.
   Remote - For a remote connection you first have to specify the host name or IP address of the machine the the OPC server resides on and then (as of Ignition 7.4) you are redirected to the available servers list.

4. Select the OPC server that you wish to connect to from the list. In the case where your server is not listed, see the **OPC server is not listed...** the Troubleshooting OPC-COM Connection section.

Unique Remote Connection Settings - Remote connections have a few unique settings that you can specify. You can get to these settings by selecting the **Show advanced properties** check box. As of Ignition 7.4 these should all be set for you (except for the CLSID which should no longer be necessary but is still available for you to set if you wish).
Remote Server - Specifies that the server is remote and that a DCOM connection will be used.
Host Machine - The computer name or IP address of the machine on which the remote server is running.
CLSID - This is no longer required as of Ignition 7.4, but it is still made available for you. It can be used in place of the ProgId because the ProgId is really just used to lookup the CLSID in the registry. This id can be found in the registry of the machine hosting the server under:
HKEY_CLASSES_ROOT\OPCServerName\CLSID

5. All of the settings for the server connection are rather straight forward and each property has a description of its functionality. Most of these settings should be fine when left at their default values. The only setting that could possibly give you some trouble is the ProgId. If you selected your OPC server from the list on the **Choose OPC-DA Server** page, this will be filled in for you. However, if for whatever reason your server wasn't listed and you choose the **Other Server** option, you will have to know the ProgId for your server and specify it here. The ProgId is used to look up the CLSID of the OPC Server in the Windows Registry and without this a connection cannot be made.

6. When you are finished fine tuning these settings click **Create new OPC Server Connection**. You will be redirected to the OPC Server Connections page and your new server connection should be listed. The status of your connection will read **Connected** if Ignition was able to successfully connect to the third-party OPC server.

Connection Is Faulted

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

Troubleshooting OPC-COM Connections

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

Common Problems

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

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

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

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

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

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

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

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

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

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

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

Set the OPC Server to Run as Interactive User

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

Faulted Status with E_CLASSNOTREG Error Reported on OPC Connections Status Page

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

DCOM Settings

Ignition Server DCOM Settings

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

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

OPC Server DCOM Settings

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

1. Open up **Windows Component Services**, located in the **Administrative Tools** section of the **Control Panel**.
2. Browse down through the **Component Services** tree until get to the DCOM Config folder.
3. Locate the entry for your OPC server that you wish to make a connection to, right-click and select properties.

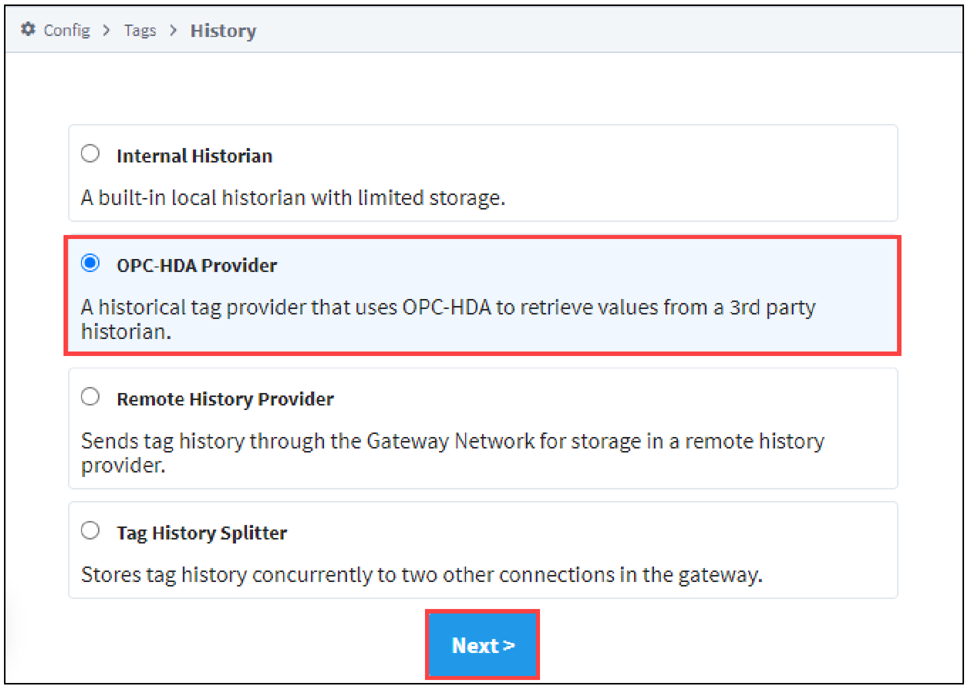
4. Click the **Security** tab and you will see three sections, **Launch and Activation Permissions**, **Access Permissions**, and **Configuration Permissions**. There are two options to choose from for each section. If you already added the ANONYMOUS and Everyone accounts to the COM Security section from the **Ignition Server DCOM Settings** section then you can go ahead and just select the **Use Default** option for each of the three areas. The second option is to edit each of the groups that have **Customize** selected. You will have to add both the **ANONYMOUS** and **Everyone** accounts with all privileges.

5. Now select the **Identity** tab. You will notice that you can choose which account you want to run the OPC server under. Select the **Interactive User** option. This ensures that if Ignition launches an instance of the OPC server, it will run under whichever user is currently logged into the system.

Creating an OPC-HDA Connection

The process of connecting to an OPC HDA server is similar to that of a DA server. Instead of going to the "OPC Connections" section, however, you define the server as a Tag History Provider.

1. Navigate to the Gateway Config Webpage, as outlined above.
2. Under Tags, History, select "Create new Historical Tag Provider..."
3. Select OPC HDA Provider and click **Next**.



○ **Internal Historian**
A built-in local historian with limited storage.

● **OPC-HDA Provider**
A historical tag provider that uses OPC-HDA to retrieve values from a 3rd party historian.

○ **Remote History Provider**
Sends tag history through the Gateway Network for storage in a remote history provider.

○ **Tag History Splitter**
Stores tag history concurrently to two other connections in the gateway.

4. Follow the step outlined above, for DA connections.
5. Once complete, the status on the Tags>History screen will show the state of the connection. If Connected, you should now be able to browse and query the server through the Ignition designer.

OPC-HDA Properties

OPC-HDA Provider

Provides a connection to an OPC-HDA server.

Provider Name	Name of the Tag History Provider.
Enabled	If the check box is selected (enabled), the provider is turned on and will expose historical data
Description	A description of the provider.
ProgId	A description of the provider.

Use Flat Browsing	Flat browsing returns all items at once. This is less efficient than normal browsing, but if a server only supports flat browsing, then this needs to be checked.
Remote Server	If selected, DCOM will be used to connect to the server on the specified Host with the given ProgId or CLSID.
Host Machine	The name or IP address of the machine hosting the server. Leave empty for local machine.
CLSID	The CLSID of the server. If not specified, will be obtained using the ProgId.

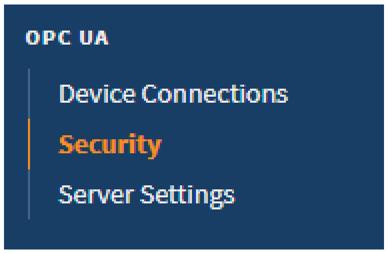
Example - Adding OPC-HDA data to a chart

1. Open the designer, and create or open a project.
2. Create a window, and add an Easy Chart component.
3. Double-click on the chart, or right click and select Customizers>Easy Chart Customizer to bring up the chart customization window.
4. Next to the "Tag History Pens" table, select the first button, "Browse for tags". This will display a tree for browsing all historical tags.
5. Browse through your defined HDA server. Once you find a tag, select "ok" to add it to the chart.
6. You may edit the tag to alter its aggregation mode, though the HDA provider will select a supported mode automatically if the specified mode does not exist in the server.
7. Once you save the configuration, the chart should update with the requested data.

A similar procedure can be used anywhere Tag History can be bound or used.

OPC UA Security

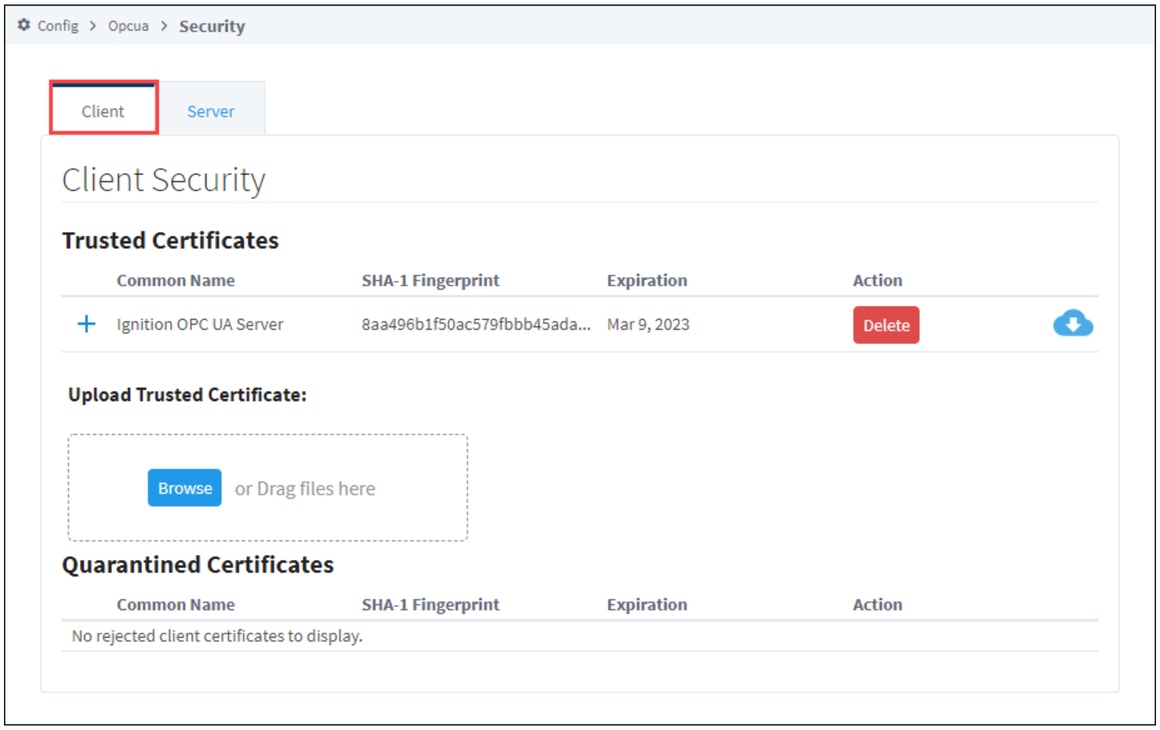
On the OPC UA security page you can manage OPC UA certificates for the client and server. Trusted certificates can be imported and quarantined certificates can be marked as trusted.



On this page ...

Trusted Certificates on the Client

When viewing the **Client** tab, you're viewing the certificates trusted by the Gateway, as a UA client. In the screenshot below, you can see that this client trusts the certificate named "Ignition OPC UA Server."



Client Security

Trusted Certificates

Common Name	SHA-1 Fingerprint	Expiration	Action
Ignition OPC UA Server	8aa496b1f50ac579fbbb45ada...	Mar 9, 2023	Delete

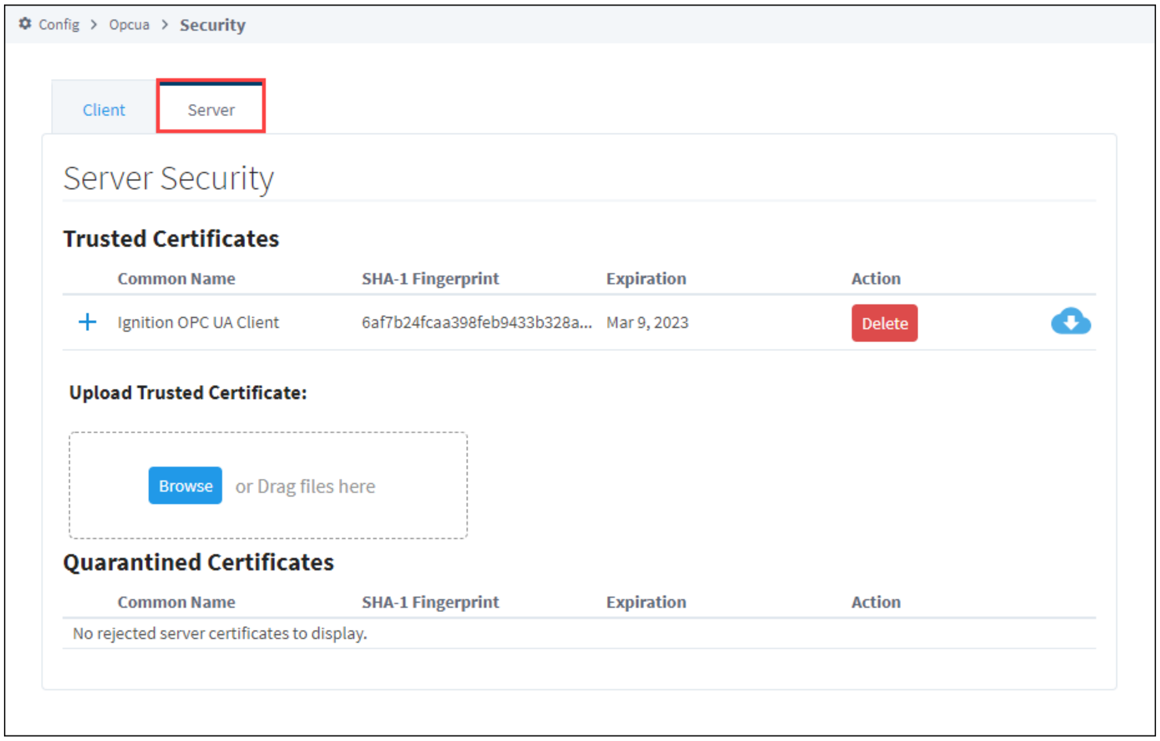
Upload Trusted Certificate:

Quarantined Certificates

Common Name	SHA-1 Fingerprint	Expiration	Action
No rejected client certificates to display.			

Trusted Certificates on the Server

When viewing the **Server** tab, you're viewing the certificates trusted by the server, meaning the Gateway's OPC UA server. In the screenshot below, you can see that this server trusts the certificate named "Ignition OPC UA Client."



Config > Opcua > Security

Client | Server

Server Security

Trusted Certificates

	Common Name	SHA-1 Fingerprint	Expiration	Action	
+	Ignition OPC UA Client	6af7b24fcaa398feb9433b328a...	Mar 9, 2023	Delete	

Upload Trusted Certificate:

Browse or Drag files here

Quarantined Certificates

Common Name	SHA-1 Fingerprint	Expiration	Action
No rejected server certificates to display.			

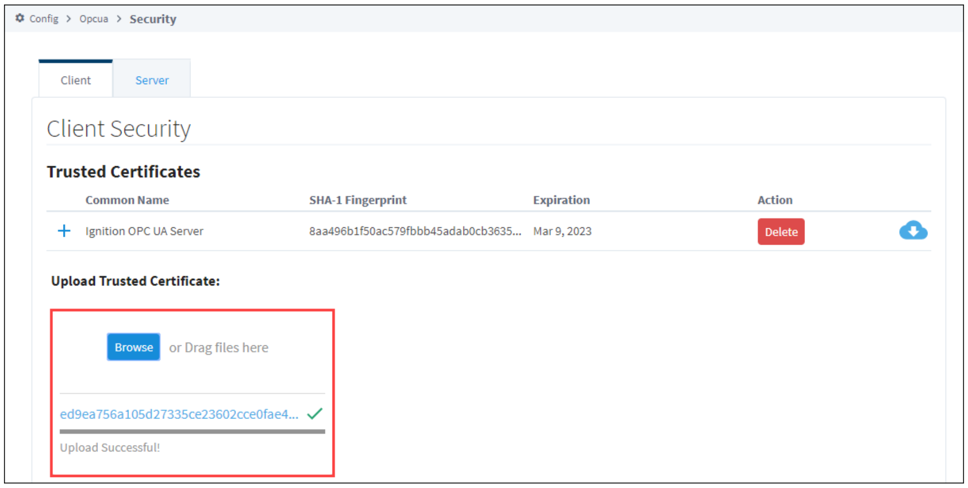
Managing Certificates

The steps for managing trusted certificates are the same whether you're on the Client tab or the Server tab.

Upload a Trusted Certificate

To upload a trusted Certificate, do the following.

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



Config > Opcua > Security

Client | Server

Client Security

Trusted Certificates

	Common Name	SHA-1 Fingerprint	Expiration	Action	
+	Ignition OPC UA Server	8aa496b1f50ac579fbbb45adab0cb3635...	Mar 9, 2023	Delete	

Upload Trusted Certificate:

Browse or Drag files here

ed9ea756a105d27335ce23602cce0fae4... ✓

Upload Successful!

Download a Trusted Certificate

To download a trusted certificate, do the following.

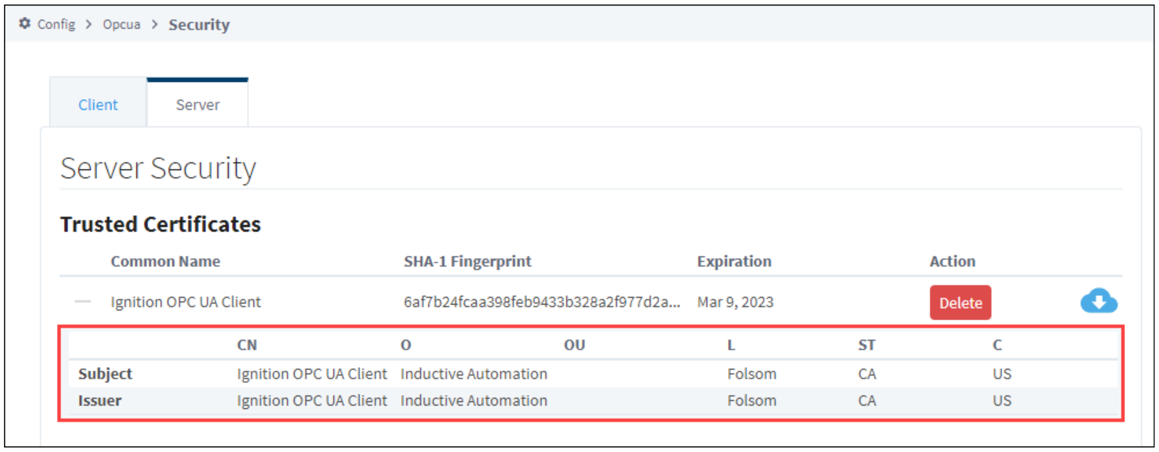
1. Next to the certificate name, click the **Download** icon.
2. The certificate is downloaded to your system by your web browser.

Delete a Trusted Certificate

To delete a trusted certificate, do the following.

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

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



Server Security

Trusted Certificates

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

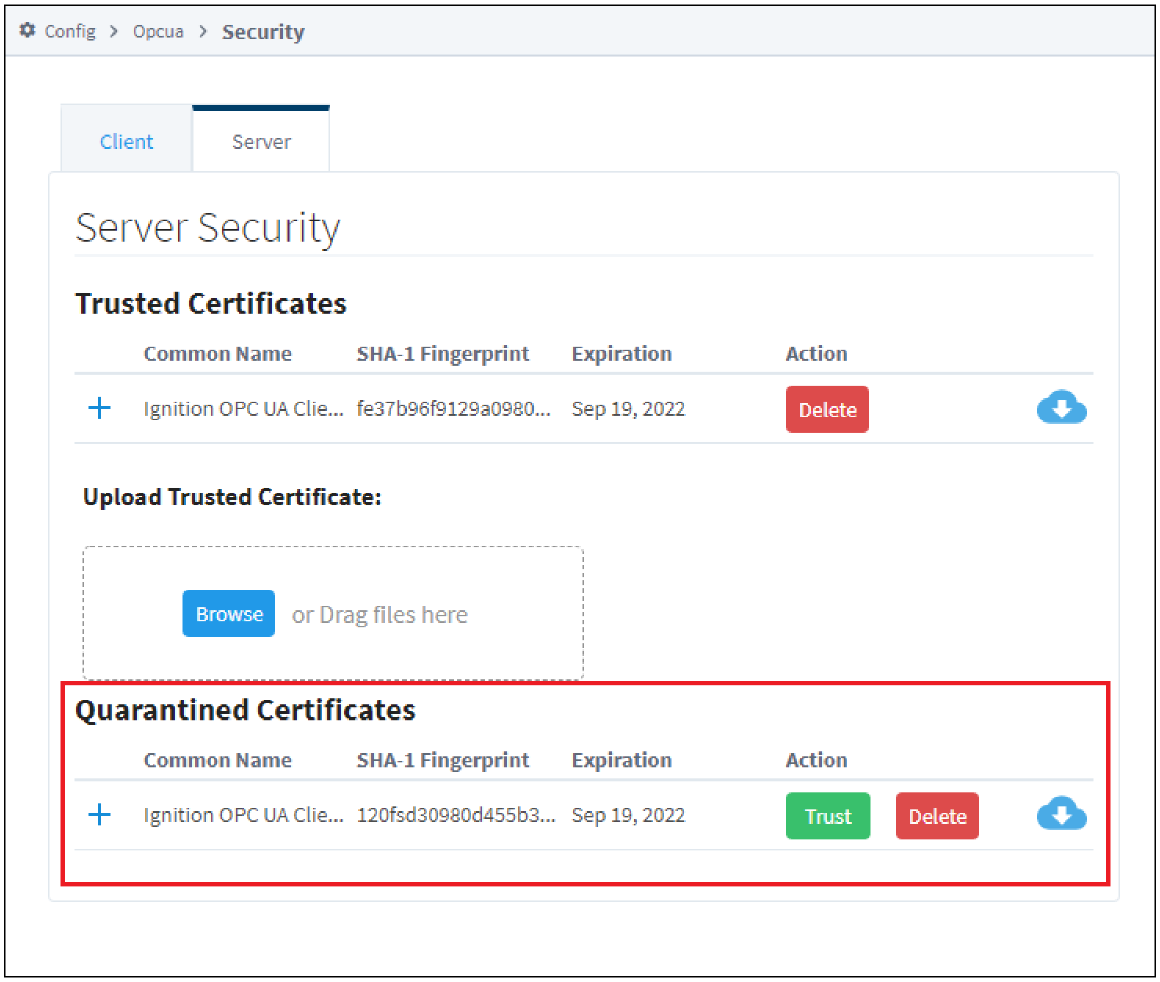
	CN	O	OU	L	ST	C
Subject	Ignition OPC UA Client	Inductive Automation		Folsom	CA	US
Issuer	Ignition OPC UA Client	Inductive Automation		Folsom	CA	US

OPC UA Security Page Details

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

Quarantined Certificates

If you import a certificate that is not trusted, it will appear on the Quarantined Certificates list.


Config > Opcua > Security

Server Security

Trusted Certificates

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

Upload Trusted Certificate:

Quarantined Certificates

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

Accept a Quarantined Certificate

To accept a quarantined certificate, do the following:

1. Next to the certificate name, click the **Trust** action button.
2. The certificate is accepted and will appear in the Trusted Certificates list.

OPC UA Drivers

Ignition has a few drivers 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.

On this page ...

Allen Bradley Drivers

The Allen Bradley Driver module and Logix Driver module provides the ability to use any of our Allen Bradley suite of drivers. One of the more popular options, it contains drivers to connect to:

- Logix family devices with firmware versions 21 and newer
- MicroLogix 1100 and 1400
- PLC5
- SLC 5/05
- CompactLogix and ControlLogix with legacy firmware 20.18 and prior

Modbus Drivers

The Modbus Driver module allow you to connect to a wide range of devices using the Modbus protocol. While this typically means figuring out the Modbus specific addresses for each register, address mapping makes this process simpler by mapping a range of addresses for you.

Siemens Drivers

The Siemens Drivers module allows you to connect to a number of Siemens S7 devices. The available device drivers are:

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

UDP and TCP Drivers

The UDP and TCP Drivers module are very basic drivers that can connect to one or more ports of a given IP address and read in data. This is typically used for barcode scanners or scales.

DNP3 Driver

The DNP3 Driver module allows you to connect to a DNP3 outstation. Many different types of devices support DNP3.

Omron NJ Driver

The Omron Driver module allows you to connect to the Omron NJ series of devices.

The following feature is new in Ignition version **8.0.8**
Click here to check out the other new features

Omron FINS Driver

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

Simulator Drivers

There are a number of drivers that do not connect to any device, but instead provide simulated values for testing. These simulators are a part of the Allen Bradley Driver module.

In This Section ...

Allen Bradley Ethernet

Connecting to Allen Bradley Devices

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

- **Logix** ControlLogix and CompactLogix firmware v21 and newer
- **ControlLogix** firmware v20.18 and prior
- **CompactLogix** firmware v20.18 and prior
- **PLC-5**
- **MicroLogix** 1100, 1200, 1400, 1500
- **SLC** 5/05

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

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

Verifying Device Connectivity

Device connectivity can be verified in the following places:

- In the **Config** page of the Gateway under **OPC UA > Device Connections**.
- In the **Status** page of the Gateway under **Overview >** click the **Devices** box.
- In the **Status** page of the Gateway under **Connections > Devices**.
- In the **Status** page of the Gateway under **Connections > OPC Connections**.

Allen Bradley Connection Paths Explained

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

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

Follow the Path

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

Connection Path Steps

1. Move to the backplane.
2. Specify the slot number of the module you are moving to.

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

Connection Path Entries for Different Module Types

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

ENET, ENBT, and EN2T:
Exiting
  1 = Backplane
  2 = Ethernet Port
Entering
  IP Address

CNB:
Exiting
  1 = Backplane
  2 = ControlNet Port
Entering
  ControlNet Address

DHRIO
Exiting
  1 = Backplane
  2 = DH+ Channel A
  3 = DH+ Channel B
Entering
  DH+ Station Number (an octal value between 0-77)

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

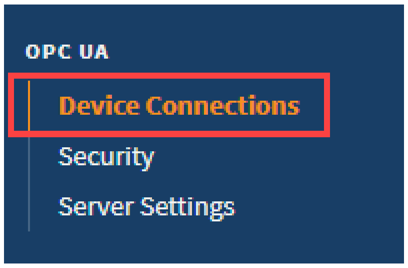
In This Section ...

Connecting to CompactLogix

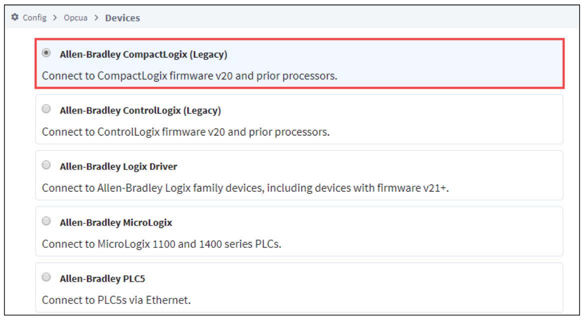
This driver is made for CompactLogix firmware up to version 20.18.

Connect to an Allen-Bradley CompactLogix Device

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



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



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

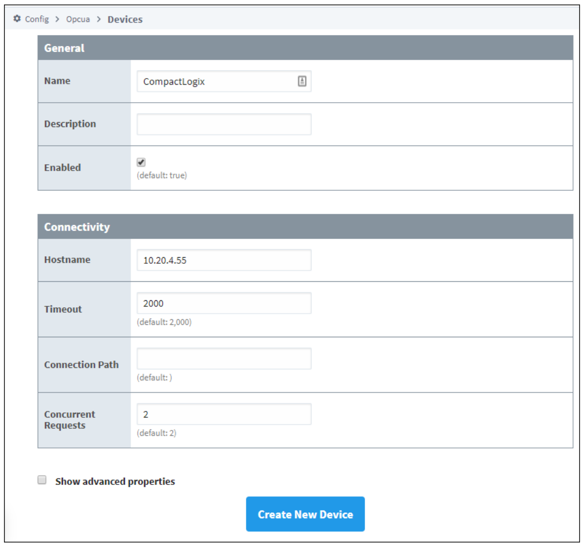
( ) **Allen-Bradley ControlLogix (Legacy)**
Connect to ControlLogix firmware v20 and prior processors.

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

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

( ) **Allen-Bradley PLC5**
Connect to PLC5s via Ethernet.

5. On the **New Device** page, leave all the default values and type in the following fields:
   Name: **CompactLogix**
   Hostname: type the **IP address** for the PLC, for example 10.20.4.55.



General	
Name	CompactLogix
Description	
Enabled	✓ (default: true)

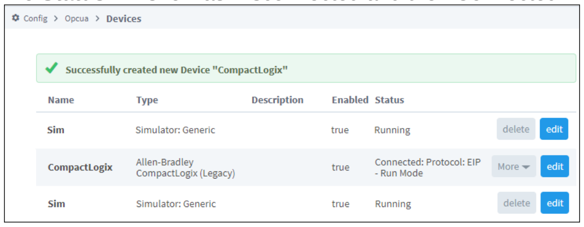
Connectivity	
Hostname	10.20.4.55
Timeout	2000 (default: 2,000)
Connection Path	(default:)
Concurrent Requests	2 (default: 2)

☐ Show advanced properties

Connecting to CompactLogix

Watch the Video

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



Config > Opcua > Devices

Successfully created new Device "CompactLogix"

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

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

Device Connection Settings

The general settings are common to all Allen Bradley devices. The connectivity and advanced settings are device dependent.

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

Connectivity	
Host name	This is the IP Address of the Ethernet module to route through to connect a CompactLogix processor.
Time out	The timeout settings refer to the communication between the device driver and the OPC-UA server and usually can be left at their default values.
Connection Path	The Connection Path value is used to define the route to connect to the processor. The Connection Path format contains 4 numbers separated by commas. The first number is always 1 and tells the Ethernet module to route through the backplane. The rest of the numbers vary by device type, for a more in depth explanation of connection paths, see Allen Bradley Connection Paths Explained section.
Concurrent Requests	The number of requests that Ignition will try to send to the device at the same time. Increasing this number can sometimes help with your request throughput, however increasing this too much can overwhelm the device and hurt communications with the device.

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

Related Topics ...

- Connecting to MicroLogix

Connecting to ControlLogix

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

On this page ...

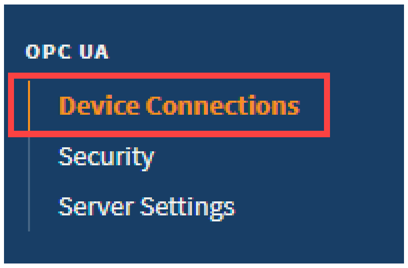
INDUCTIVE UNIVERSITY

Connecting to ControlLogix

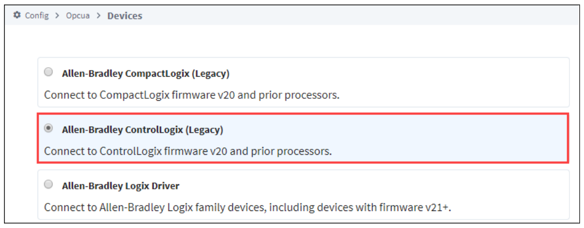
Watch the Video

Connect to an Allen-Bradley ControlLogix Device

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



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



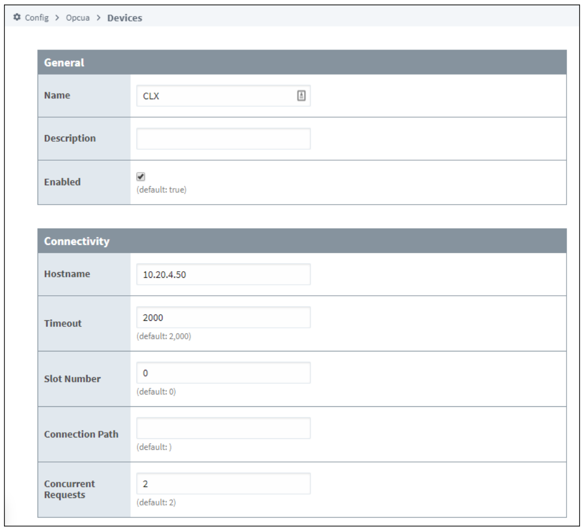
Config > Opcua > Devices

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

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

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

5. On the **Devices** page, leave all the default values and type in the following fields:
   Name: **CLX**
   Hostname: Enter the **IP address** for the PLC, for example 10.20.4.50.

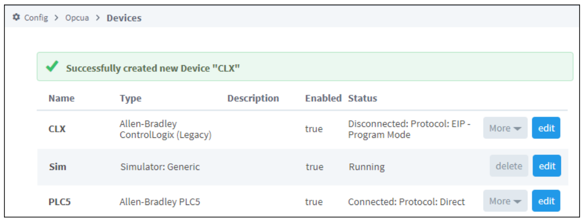


Config > Opcua > Devices

General	
Name	CLX
Description	
Enabled	✓ (default: true)

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

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



Successfully created new Device "CLX"

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

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

Device Connection Settings

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

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

Connectivity	
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.
Timeout	After sending a request to the ControlLogix processor, the Communication Timeout setting is the amount of time in msec to wait for a response before treating it as a failure.
Slot Number	The Slot Number value is the zero based ControlLogix chassis slot number of the ControlLogix processor to connect to.
Connection Path	The Connection Path value is used to define the route of the PLC-5 processor to connect to. Currently routing through the ControlLogix Ethernet Communication Interface Module (1756-ENET) to the ControlLogix Data Highway Plus-Remote I/O Communication Interface Module (1756-DHRIO) and on to a PLC-5 processor of the DH+ network is supported.
Concurrent Requests	The number of requests that Ignition will try to send to the device at the same time. Increasing this number can sometimes help with your request throughput, however increasing this too much can overwhelm the device and hurt your communications with the device.

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

Supported Connection Methods

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

Related Topics ...

- Connecting to CompactLogix

Connecting to Logix

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

On this page ...

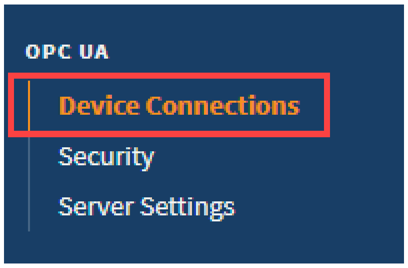
INDUCTIVE UNIVERSITY

Connecting to ControlLogix v21

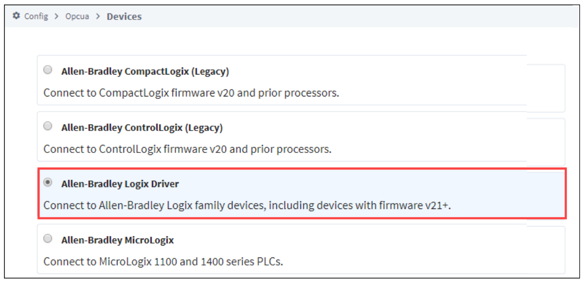
Watch the Video

Connect to an Allen-Bradley Logix Device

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



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



Config > Opcua > Devices

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

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

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

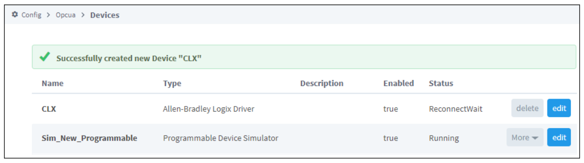
Allen-Bradley MicroLogix
Connect to MicroLogix 1100 and 1400 series PLCs.

5. Fill in the following fields:

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

   Leave the default values in the remaining fields.

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



Config > Opcua > Devices

✓ Successfully created new Device "CLX"

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

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

Driver Implementation

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

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

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

Device Connection Settings

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

General

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

Connectivity

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	The following feature is new in Ignition version **8.0.13** Click here to check out the other new features Port to connect to the remote device.
Timeout	After sending a request to the ControlLogix processor, the Communication Timeout setting is the amount of time in msec to wait for a response before treating it as a failure.
Max Concurrent Requests	The number of requests that Ignition will try to send to the device at the same time. Increasing this number can sometimes help with your request throughput, however increasing this too much can overwhelm the device and hurt your communications with the device.
Slot Number	The Slot Number value is the zero based ControlLogix chassis slot number of the ControlLogix processor to connect to.

Advanced

Automatic Rebrowse	Monitor for tag additions and UDT changes and automatically initiate a re-browse when detected. If this is disabled tags will only be browsed when connecting and reconnecting. (default: true)

CIP Connection Size	The CIP connection size to use during Forward Open requests. (default: 500)

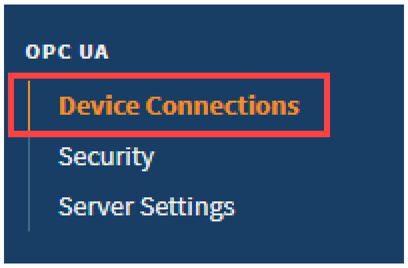
Related Topics ...

- Connecting to ControlLogix

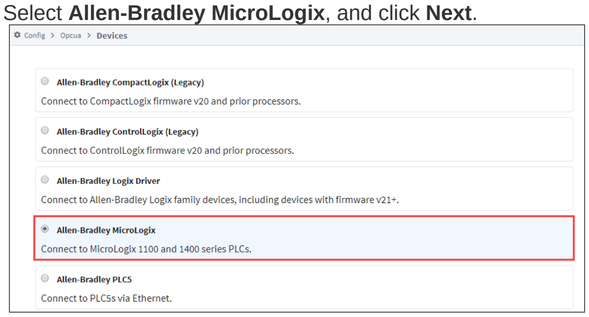
Connecting to MicroLogix

Connect to a Device

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



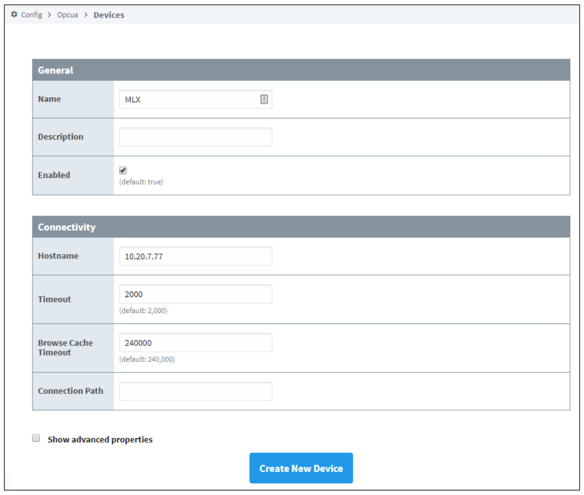
3. On the **Devices** page, find the blue arrow and click on **Create new Device**.
4. Select **Allen-Bradley MicroLogix**, and click **Next**.



Config > Opcua > Devices

- Allen-Bradley CompactLogix (Legacy)
  Connect to CompactLogix firmware v20 and prior processors.
- Allen-Bradley ControlLogix (Legacy)
  Connect to ControlLogix firmware v20 and prior processors.
- Allen-Bradley Logix Driver
  Connect to Allen-Bradley Logix family devices, including devices with firmware v21+.
- [✓ Allen-Bradley MicroLogix]
  Connect to MicroLogix 1100 and 1400 series PLCs.
- Allen-Bradley PLC5
  Connect to PLC5s via Ethernet.

5. On the **New Device** page, leave all the default values and type in the following fields:
   Name: **MLX**
   Hostname: Enter **IP address** for the PLC, for example 10.20.7.77



Config > Opcua > Devices

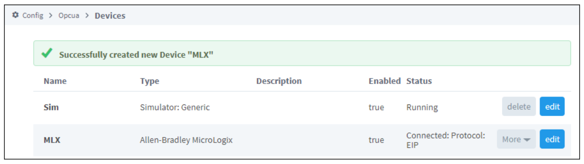
General	
Name	MLX
Description	
Enabled	✓ (default: true)

Connectivity	
Hostname	10.20.7.77
Timeout	2000 (default: 2,000)
Browse Cache Timeout	240000 (default: 240,000)
Connection Path	

Show advanced properties

Create New Device

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



Config > Opcua > Devices

✓ Successfully created new Device "MLX"

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

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

Device Connection Settings

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

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

Connectivity	
Hostname	The Hostname value is the IP Address of the MicroLogix 1100 processor, MicroLogix 1400 processor or 1761-NET-ENI Ethernet interface. EthernetIP protocol on TCP port 44818 (0xAF12) is used to communicate to the listed devices.
Communication Timeout	After sending a request to the MicroLogix processor, the Communication Timeout setting is the amount of time in msec to wait for a response before treating it as a failure.
Browse Cache Timeout	When the data table layout is read from the MicroLogix processor, the Browse Cache Timeout value is the amount of time in msec to cache the results.

Supported MicroLogix Connection Methods

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

Important

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

Additionally, the MicroLogix driver can not access Input Parameters.

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

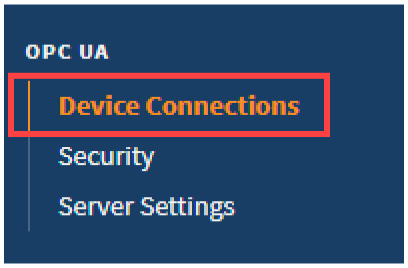
Related Topics ...

- Connecting to PLC5

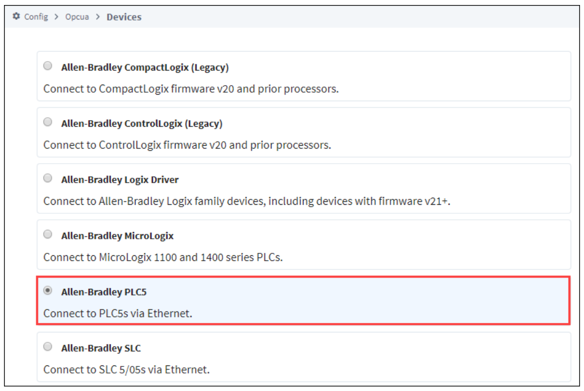
Connecting to PLC5

Connect to a Device

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



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



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

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

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

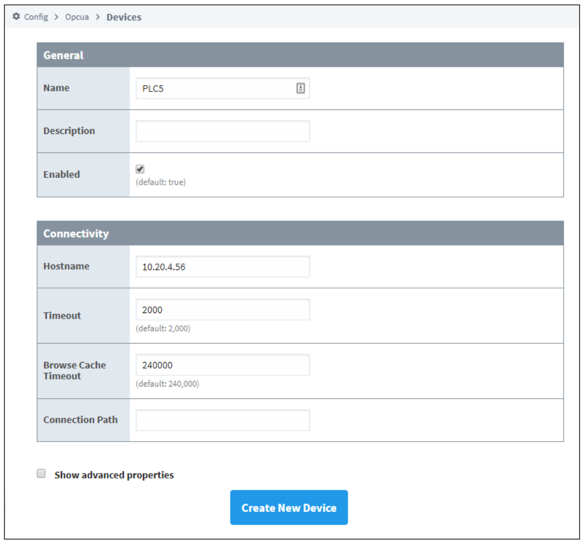
Allen-Bradley MicroLogix
Connect to MicroLogix 1100 and 1400 series PLCs.

Allen-Bradley PLC5
Connect to PLC5s via Ethernet.

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

5. Fill in the following fields:
   Name: **PLC5**
   Hostname: type the **IP address** for the PLC, for example 10.20.4.56.

   Leave the default values for the remaining fields.



General	
Name	PLC5
Description	
Enabled	✓ (default: true)

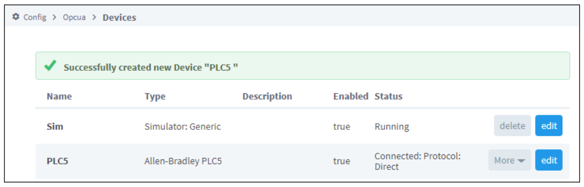
Connectivity	
Hostname	10.20.4.56
Timeout	2000 (default: 2,000)
Browse Cache Timeout	240000 (default: 240,000)
Connection Path	

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

INDUCTIVE UNIVERSITY

Connecting to PLC5

Watch the Video



Config > Opcua > Devices

✓ Successfully created new Device "PLC5 "

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

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

Device Connection Settings

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

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

Connectivity	
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).
Communication Timeout	After sending a request to the PLC-5 processor, the Communication Timeout setting is the amount of time in milliseconds to wait for a response before treating it as a failure.
Browse Cache Timeout	When the data table layout is read from the PLC-5 processor, the Browse Cache Timeout value is the amount of time in milliseconds to cache the results.
Connection Path	The Connection Path value is used to define the route of the PLC-5 processor to connect to. Currently routing through the ControlLogix Ethernet Communication Interface Module (1756-ENET) to the ControlLogix Data Highway Plus-Remote I/O Communication Interface Module (1756-DHRIO) and on to a PLC-5 processor of the DH+ network is supported.

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

More Information On Connection Path

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

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

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

Supported PLC-5 Connection Methods

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

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

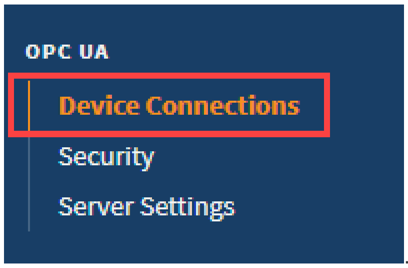
Related Topics ...

- Connecting to SLC

Connecting to SLC

Connect to an Allen-Bradley SLC Device

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

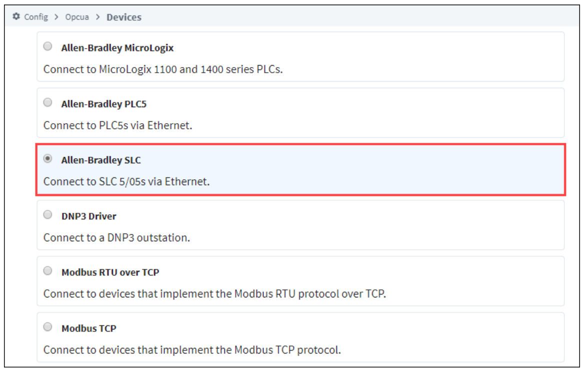


.

3. On the **Devices** page, click on **Create new Device**.

4. Select **Allen-Bradley SLC**, and click **Next**.

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



Config > Opcua > Devices

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

Allen-Bradley PLC5
Connect to PLC5s via Ethernet.

[✓ Allen-Bradley SLC]
Connect to SLC 5/05s via Ethernet.

DNP3 Driver
Connect to a DNP3 outstation.

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

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

5. Fill in the following fields:

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

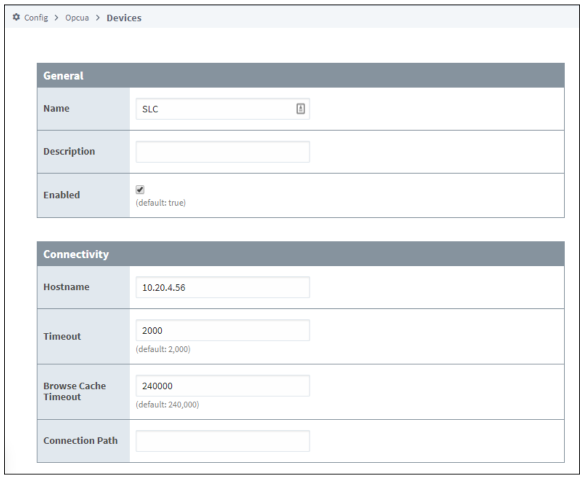
   Leave the default values in the remaining fields.

On this page ...

INDUCTIVE UNIVERSITY

Connecting to SLC

Watch the Video



Config > Opcua > Devices

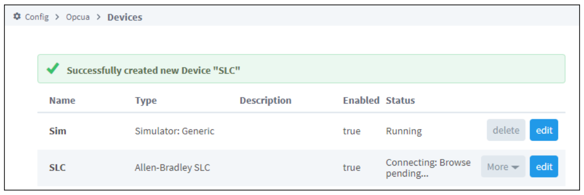
General

Name	SLC
Description	
Enabled	✓ (default: true)

Connectivity

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

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



Config > Opcua > Devices

Successfully created new Device "SLC"

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

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

Device Connection Settings

The general settings are common to all Allen Bradley devices, and the connectivity and advanced settings are device dependent.

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

Connectivity	
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).
Communication Timeout	After sending a request to the SLC processor, the Communication Timeout setting is the amount of time in milliseconds to wait for a response before treating it as a failure.

Browse Cache Timeout	When the data table layout is read from the SLC processor, the Browse Cache Timeout value is the amount of time in milliseconds to cache the results.
Connection Path	The Connection Path value is used to define the route of the SLC processor to connect to. Currently routing through the ControlLogix Ethernet Communication Interface Module (1756-ENET) to the ControlLogix Data Highway Plus-Remote I/O Communication Interface Module (1756-DHRIO) and on to a SLC processor of the DH+ network is supported.

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

More Information On Connection Path

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

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

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

Supported SLC Connection Methods

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

Allen-Bradley Connection Paths

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

On this page ...

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 the 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:

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

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

Examples

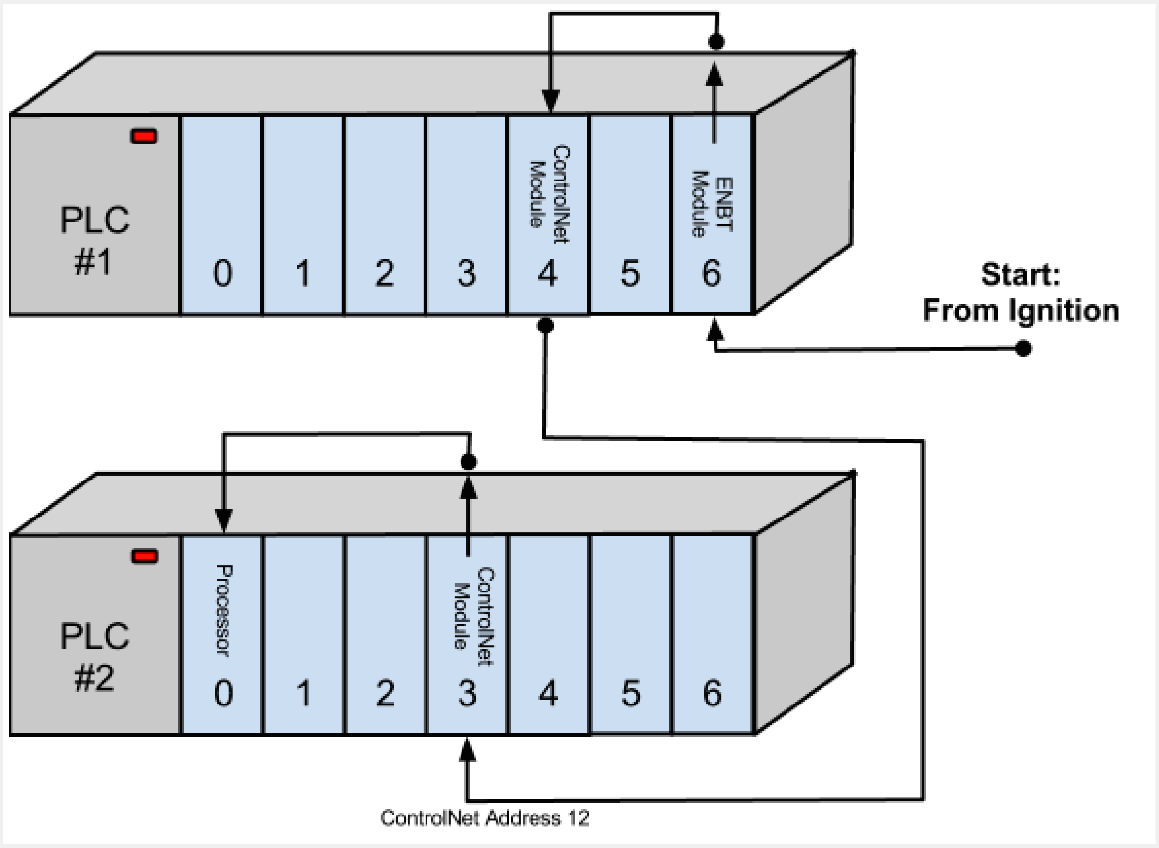
Below there are a few examples that go over the differences with using specific modules to route the connection. Note that typically, the end result PLC does not matter. So while the example may show connecting to a PLC5 through ControlNet, the the PLC5 could easily be swapped out with another PLC, and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

⚠ 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

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).

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



INDUCTIVE UNIVERSITY

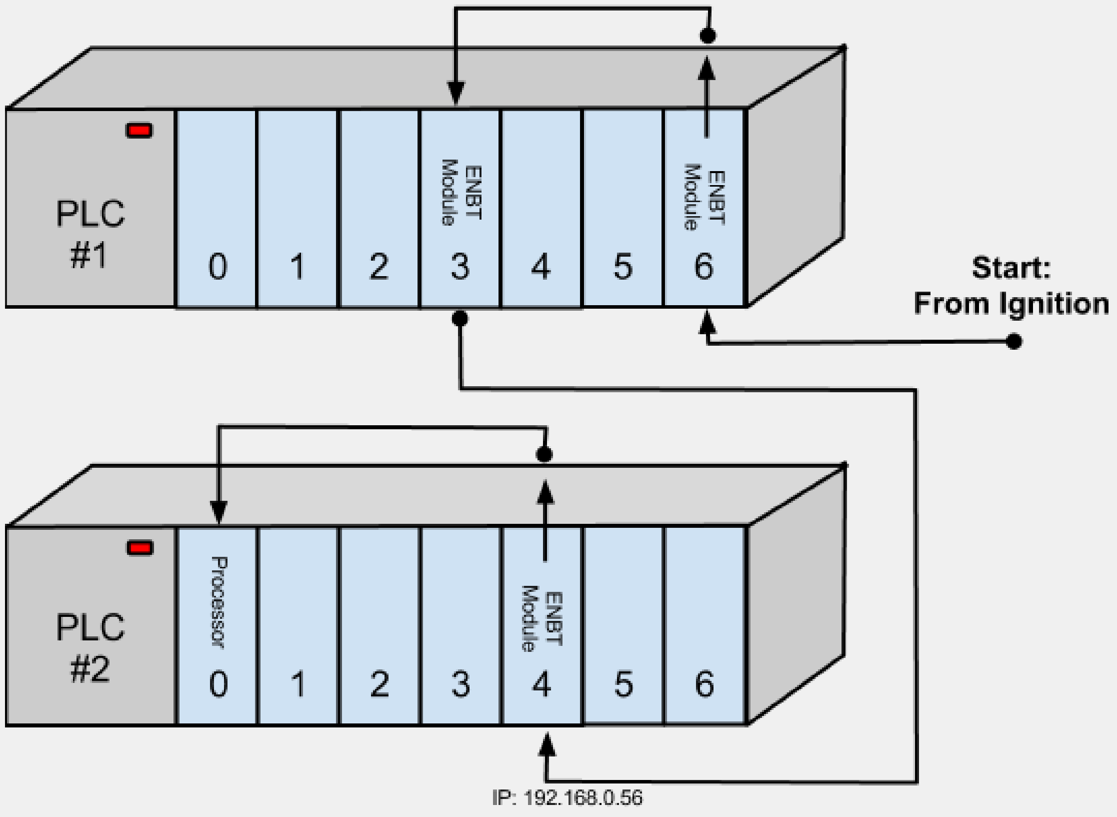
ControlNet Example

Watch the Video

Connecting to a PLC using ENBT

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).

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



INDUCTIVE UNIVERSITY

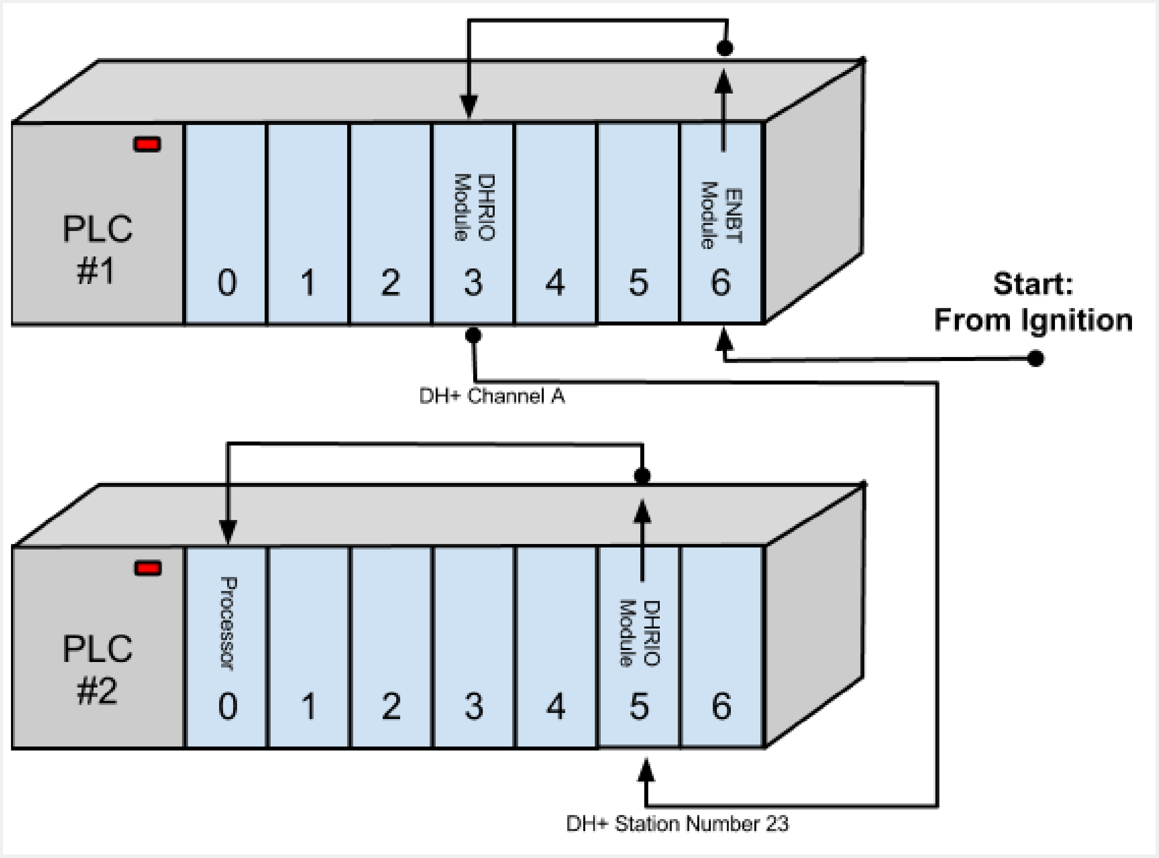
An ENBT Example

Watch the Video

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).

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+)

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 Allen-Bradley ControlLogix, and the hostname is the IP address of the ControlLogix Gateway (PLC #1).

SLC X

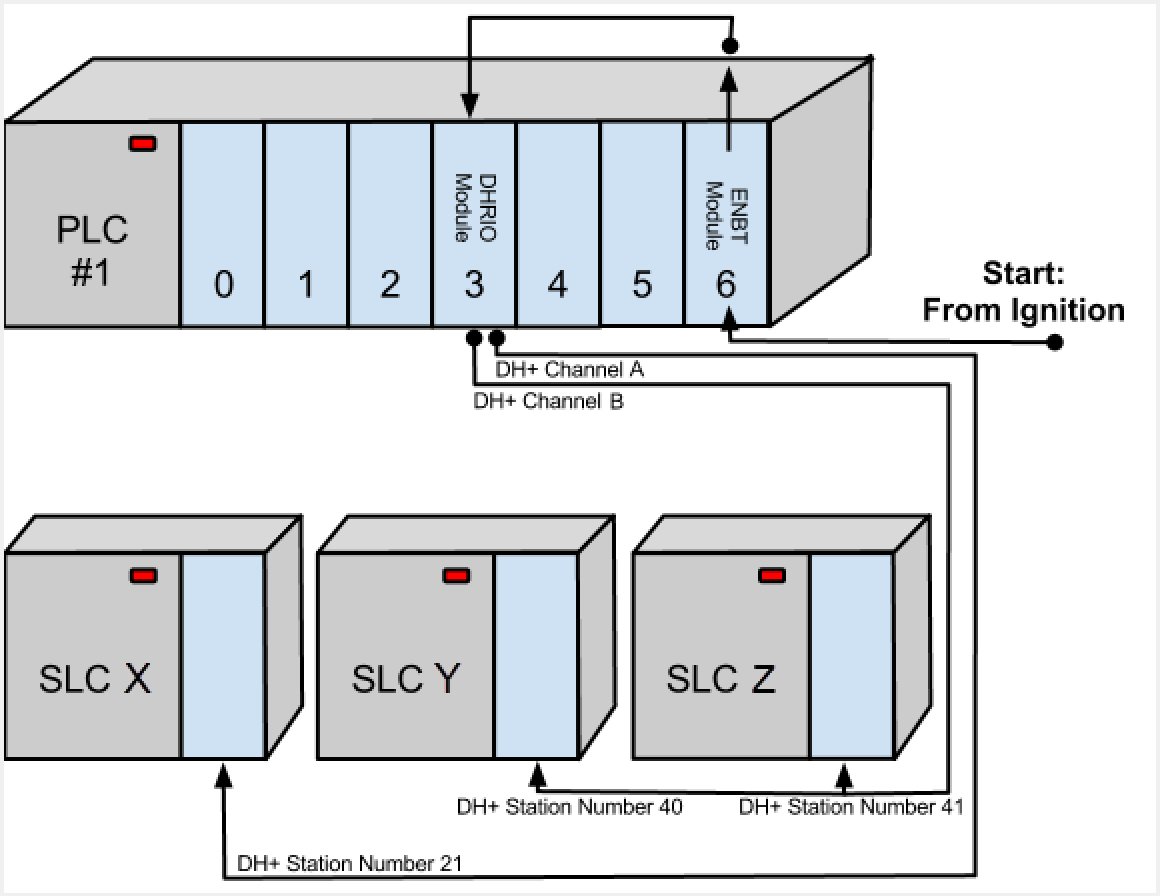
1	Move out of ENBT Module to the backplane
3	Move to DHRIO Module in Slot 3
2	Move out DH+ Channel A
21	Move in DH+ Station Number 23

SLC Y

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

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



A DH+ Example

Watch the Video

Modbus

Modbus Devices

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

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

Connecting to a Device

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

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

Modbus Addresses

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

One important setting to note is the **Max Holding Registers Per Request**. This setting can cause all of your test tags to go to bad quality when only one is bad. Change this setting (under the Advanced properties) to 1 while you are testing, and set it back when you are done. If you leave it at 1, this will cause a huge strain on your system after you have added hundreds or thousands of tags from your device.

Manually Addressing

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

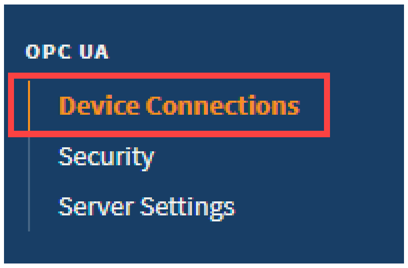
Address Mapping

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.

In This Section ...

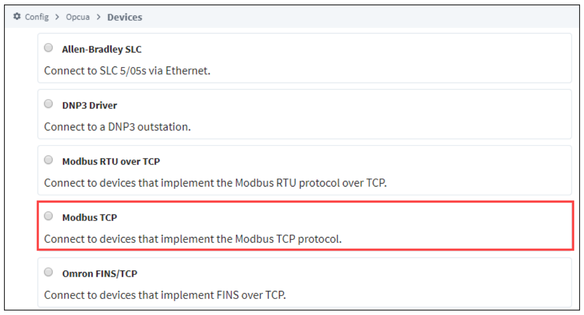
Connecting to Modbus Device

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



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

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



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

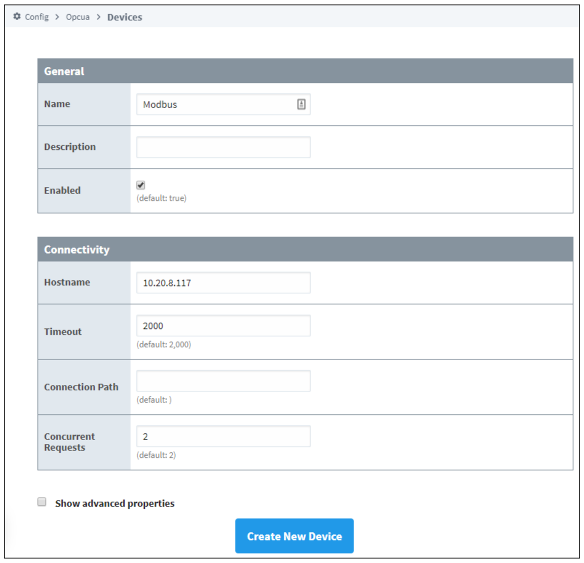
DNP3 Driver
Connect to a DNP3 outstation.

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

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

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

5. On the **New Device** page, leave all the default values and type in the following fields:
   Name: **Modbus**
   Hostname: type the **IP address**, for example 10.20.8.117.



General	
Name	Modbus
Description	
Enabled	✓ (default: true)

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

Show advanced properties

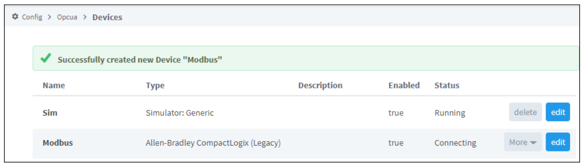
INDUCTIVE UNIVERSITY

Connecting to Modbus Device

Watch the Video

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

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



Config > Opcua > Devices

Successfully created new Device "Modbus"

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

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

Modbus Drivers using Ethernet

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

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

Supported Functions Codes

The Modbus driver supports the functions codes listed below. In some cases, a device may not allow certain function codes. To remedy this, advanced properties on the device connection can restrict or force specific functions codes. See the **Driver Properties** table on this page for more details.

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

Device Connection Settings

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

Connectivity (Modbus RTU over TCP and Modbus TCP connections)	
Hostname	Is the IP Address of the Modbus device.
Port	Is the port to use when connecting to a Modbus device. The Modbus TCP port specified in the Modbus specification is 502, but it can be changed to a different port.
Communication Timeout	Is the amount of time in milliseconds to wait for a response before treating it as a failure, after sending a request to the Modbus device. When working with a Modbus RTU over TCP connection, each "device" would be an individual Modbus device on the Modbus network.

Connectivity (Modbus RTU connections)	
Serial Port	The name of the Serial Port (i.e. COM1)
Bit Rate	The bit rate for the connection.
Data Bits	Data bits for the connection.
Parity	Parity configuration for the connection.
Stop Bits	Determines the stop bits for the connection.
Handshake	Determines the handshake for the connection.
Communication Timeout	Determines the maximum amount of time the connection will wait for a response.

Advanced	
Max Holding Registers per Request	Is the the maximum number of Holding Registers the device can handle. Because some Modbus devices cannot handle the default of requesting 125 Holding Registers in one request, to accommodate this limitation you can change this setting.
Max Input Registers per Request	Is the the maximum number of Input Registers the device can handle. Because some Modbus devices cannot handle the default of requesting 125 Input Registers in one request, to accommodate this limitation you can change this setting.
Max Coils per Request	Is the the maximum number of Coils the device can handle. Because some Modbus devices cannot handle the default of requesting 2000 Coils in one request, to accommodate this limitation you can change this setting.
Max Discrete Inputs per Request	Is the the maximum number of Discrete Inputs the device can handle. Because some Modbus devices cannot handle the default of requesting 2000 Discrete Inputs in one request, to accommodate this limitation you can change this setting.
Reverse Word Order	When reading and writing 32bit values from/to a Modbus device, the high word comes before the low word. By checking this option, the low word comes before the high word. The Modbus specification does not include a section for reading and writing 32bit values and as a result device manufacturers have implemented both methods.
One-based Addressing	This feature was removed from Ignition in version 8.0.6
Zero-based Addressing	

	The following feature is new in Ignition version **8.0.6** Click here to check out the other new features When this option is checked, the address range for each area starts at 0. If unchecked, the range starts at 1. The Modbus specification states that Modbus addresses are to be zero based. Meaning Modbus addresses start at 0 instead of 1. To read a value from Modbus address 1024, 1023 is sent to the device. When connecting to devices that do not adhere to zero based addressing, make sure this option is not selected.
Span Gaps	When this option is checked, it spans address gaps when optimizing requests, reducing the number of requests but increasing the amount of data requested at once. If unchecked, it does not span the address gaps.
Allow Write Multiple Registers Request	Enable or disable Modbus function code 0x10, Write Multiple Registers. Some devices may not support this function code. **Caution** Disabling this option will break the ability to write 32-bit and String values correctly to registers.
Force Multiple Register Writes	Force the use of Modbus function code 0x10, Write Multiple Registers, on write requests.
Allow Write Multiple Coils Request	Enable or disable Modbus function code 0x0F, Write Multiple Coils. Some devices may not support this function code.
Allow Read Multiple Registers Request	If disabled all registers will be read in individual read requests. Disable with caution.
Allow Read Multiple Coils	If disabled all coils will be read in individual read requests. Disable with caution.
Allow Read Multiple Discrete Inputs	If disabled all discrete inputs will be read in individual read requests. Disable with caution. Note: Function code 0x02 is always used to read Discrete Inputs, regardless what this property is set to.
Reconnect After Consecutive Timeouts	When checked forces a reconnect after 3 consecutive timeouts.

String Handling

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

Related Topics ...

- Modbus Addressing

Modbus Addressing

Manually Addressing a Modbus Device

Modbus doesn't support tag browsing, this means you can not view the Tags in the **OPC Browser** or **Tag Browser** of the Designer or from the **OP C Connections > Quick Client** in the **Configure** section of the Gateway.

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

1. **By manually specifying each address**
   This is done from the Designer by entering Modbus Specific Addresses into the **OPC Item Path** of an OPC Tag. See below for detailed information.

2. **By specifying the address mapping**
   This is done from the Gateway, see the Modbus Address Mapping section.

On this page ...

INDUCTIVE UNIVERSITY

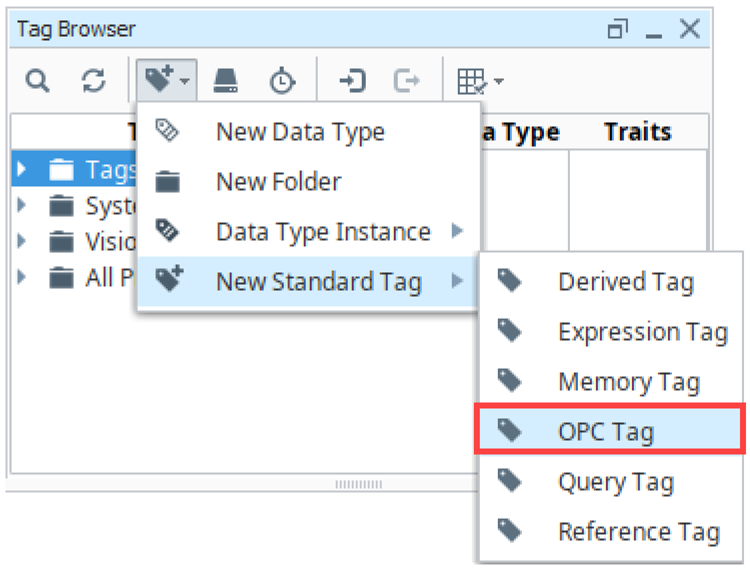
About Modbus Addressing

Watch the Video

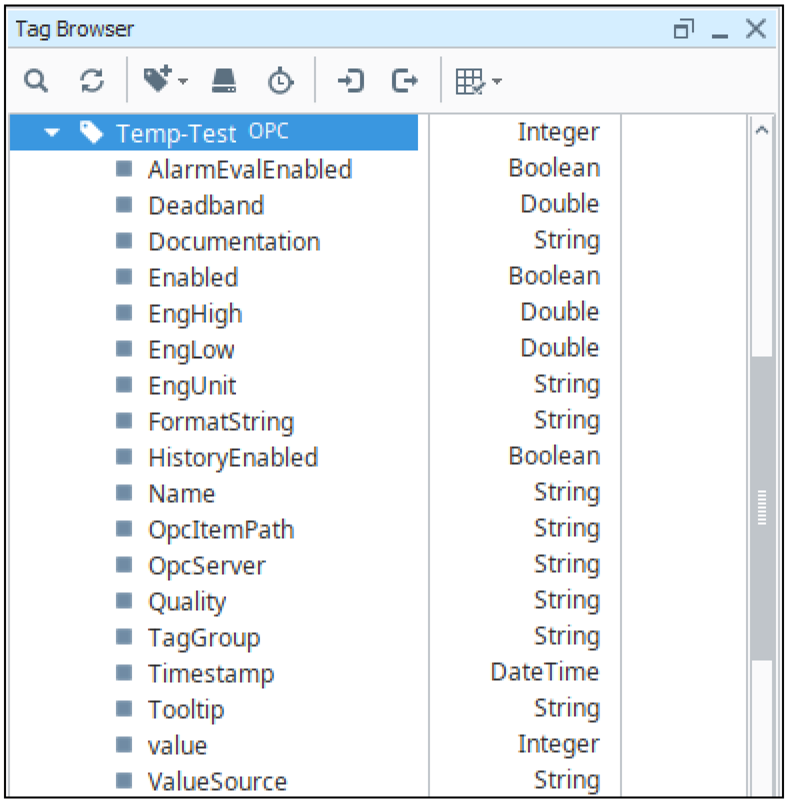
To Manually Specify Each Address

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

1. In the **Tag Browser**, right-click on the **New Tag** icon, and then go to **New Standard Tag > OPC Tag**.



2. 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 PLC which in this case is the **HR** designator plus **1** as the Modbus address. The Modbus Specific Addressing section below, explains how your can construct these addresses.


Tag Browser

Temp-Test OPC	Integer
AlarmEvalEnabled	Boolean
Deadband	Double
Documentation	String
Enabled	Boolean
EngHigh	Double
EngLow	Double
EngUnit	String
FormatString	String
HistoryEnabled	Boolean
Name	String
OpcItemPath	String
OpcServer	String
Quality	String
TagGroup	String
Timestamp	DateTime
Tooltip	String
value	Integer
ValueSource	String

3. 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)

To 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:

⚠ 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)

For example, to use these designators with the Modbus address you would 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:

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

Related Topics ...

- Modbus Address Mapping

Modbus Address Mapping

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

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. This is not very straight forward.

On this page ...

INDUCTIVE UNIVERSITY

About Modbus Address Mapping

Watch the Video

Address Mapping Properties

Name	Description
Prefix	A prefix applied to each mapped address as they appear in the OPC browser. Must compose of letters, numbers, and underscore characters. The following values are reserved, and may not be used: **HR**, **IR**, **C**, or **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.
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.
Modbus	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.

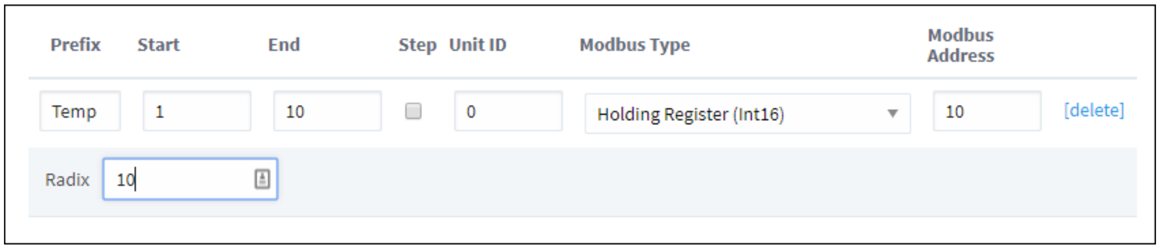
Address	Thus, when attempting to start a mapping at the 100th Holding Register, the value on this property would be 100 (assuming one-based addressing), not 40100; the Modbus Type property determines what the leading number is.

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 OPC Browser 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

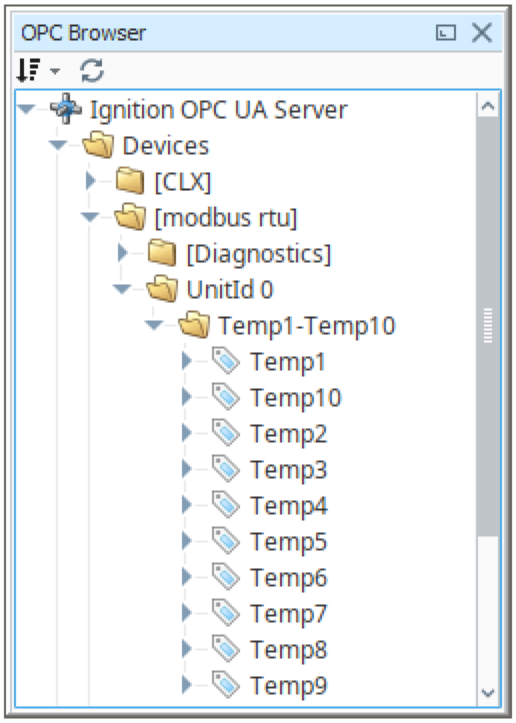
Note that 40,010 (HR10) uses Modbus Address 10 as a starting point and not 40,010. The leading 4 is automatically entered for you when you select Holding Register from the dropdown. The same is true for all other types of Tags: Inputs are 30,000, Discrete are 10,000, etc.



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

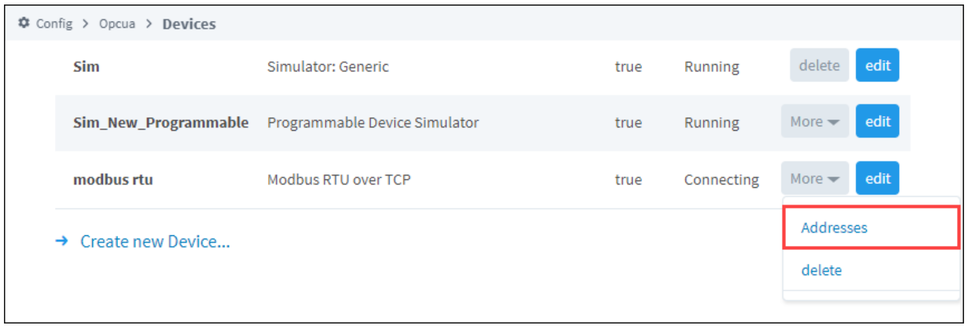
Radix: 10

The above configuration would result in the following items appearing in the OPC Browser:



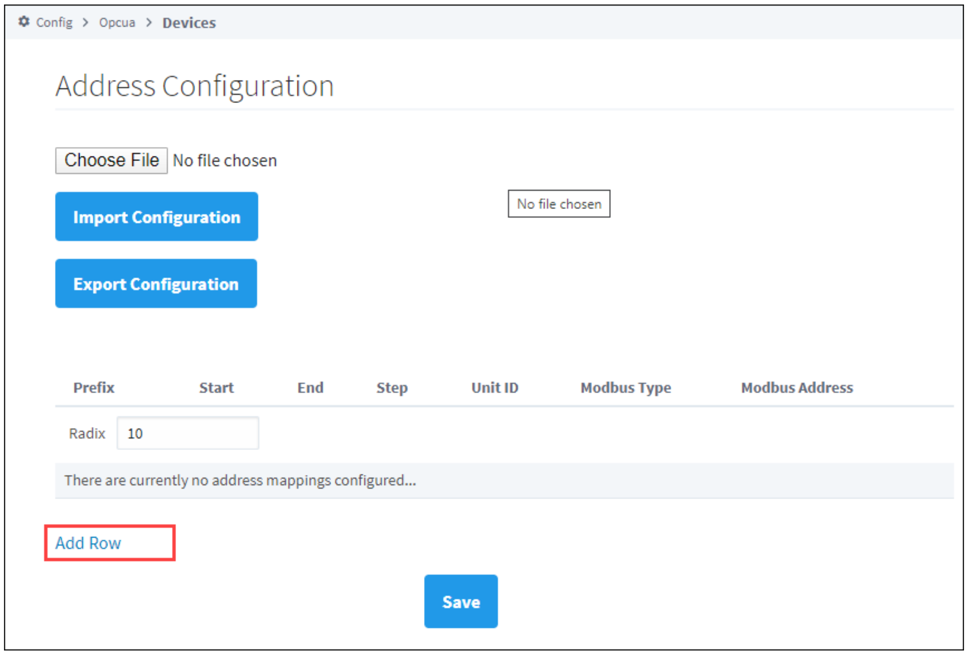
Specify the Address Mapping

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



Sim	Simulator: Generic	true	Running	delete edit
Sim_New_Programmable	Programmable Device Simulator	true	Running	More edit
modbus rtu	Modbus RTU over TCP	true	Connecting	More edit

4. Click on **Add Row**.



Address Configuration

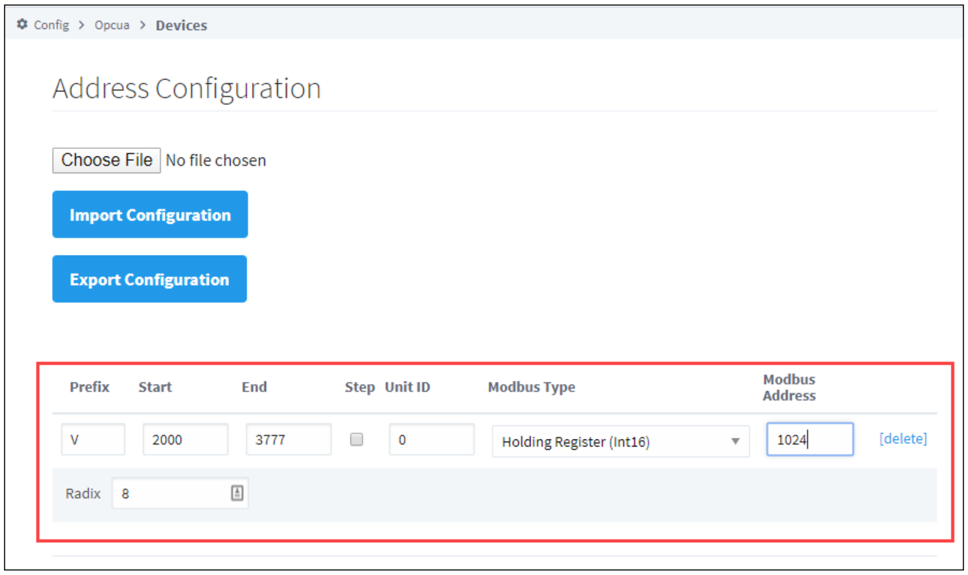
Prefix	Start	End	Step	Unit ID	Modbus Type	Modbus Address

Radix 10

There are currently no address mappings configured...

5. Enter the mapping as follows:

Prefix: **V**
Start: **2000**
End: **3777**
Modbus Type: **Holding Resister (int16)**
Modbus Address: **1024**
Radix: **8** (8 causes the addresses to be in octal, also known as base 8)


Config > Opcua > Devices

Address Configuration

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

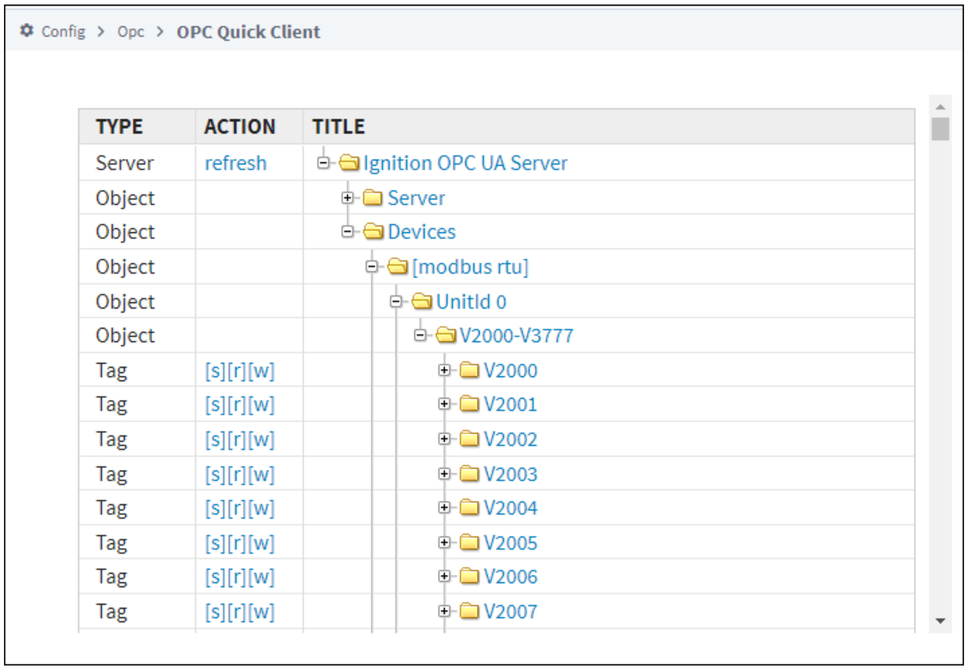
Radix 8

These settings map the **Modbus address range V2000 to V3777** in octal to **Modbus Holding Register addresses 1024 to 2047**.

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

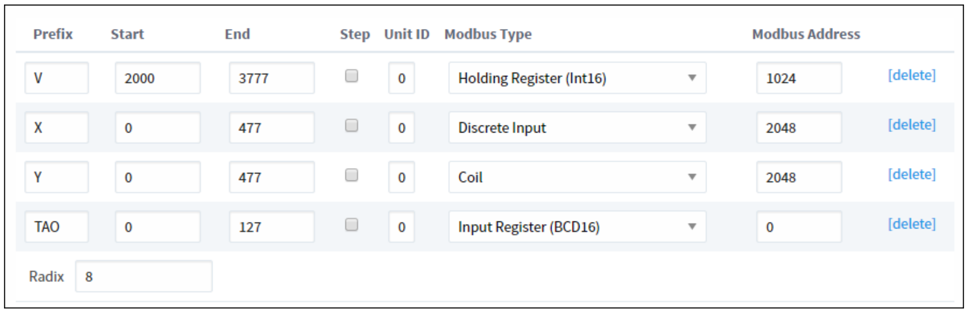
6. Click **Save**.
7. Go to the **OPC Browser** in Designer or the **OPC Connections > Quick Client** (on Gateway), open the **Modbus** folder.

   You can now see all the Modbus addresses from V2000 to V3777 in octal.


Config > Opc > OPC Quick Client

TYPE	ACTION	TITLE
Server	refresh	Ignition OPC UA Server
Object		Server
Object		Devices
Object		[modbus rtu]
Object		UnitId 0
Object		V2000-V3777
Tag	[s][r][w]	V2000
Tag	[s][r][w]	V2001
Tag	[s][r][w]	V2002
Tag	[s][r][w]	V2003
Tag	[s][r][w]	V2004
Tag	[s][r][w]	V2005
Tag	[s][r][w]	V2006
Tag	[s][r][w]	V2007

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



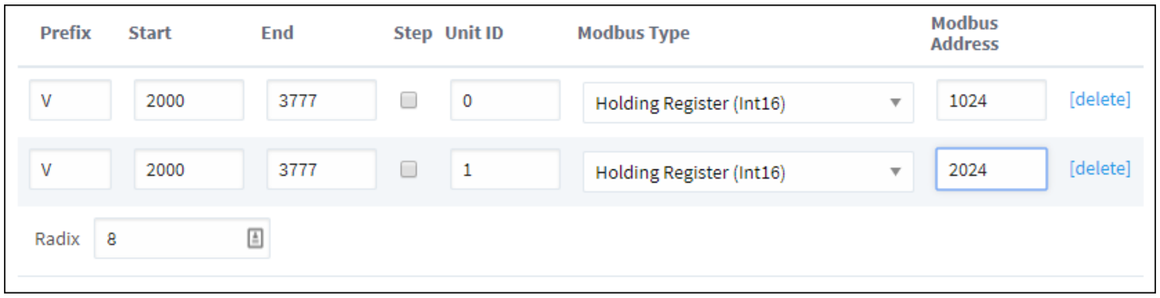
Prefix	Start	End	Step	Unit ID	Modbus Type	Modbus Address	
V	2000	3777	☐	0	Holding Register (Int16)	1024	[delete]
X	0	477	☐	0	Discrete Input	2048	[delete]
Y	0	477	☐	0	Coil	2048	[delete]
TAO	0	127	☐	0	Input Register (BCD16)	0	[delete]

Radix 8

Address Mapping Multiple Devices

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

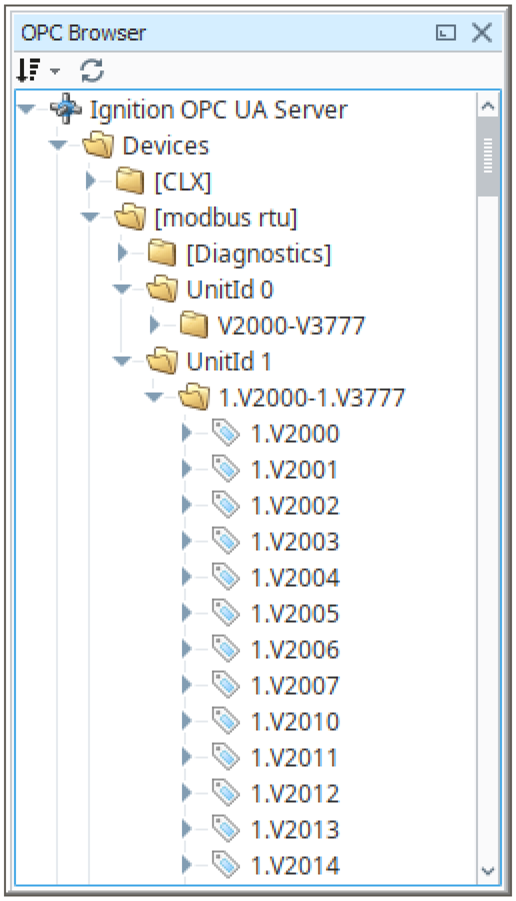
Only add one Modbus device to the Ignition OPC-UA Server device list for Gateway and specify the different unit IDs in the address mapping. The unit ID is specified for each entry in the address mapping for the Modbus device. Notice in the example below, the Prefix, Start, End, Modbus Type and Modbus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



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

Radix 8

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


OPC Browser

- Ignition OPC UA Server
  - Devices
    - [CLX]
    - [modbus rtu]
      - [Diagnostics]
      - UnitId 0
        - V2000-V3777
      - UnitId 1
        - 1.V2000-1.V3777
          - 1.V2000
          - 1.V2001
          - 1.V2002
          - 1.V2003
          - 1.V2004
          - 1.V2005
          - 1.V2006
          - 1.V2007
          - 1.V2010
          - 1.V2011
          - 1.V2012
          - 1.V2013
          - 1.V2014

Floating Point or 32-bit Address Mapping

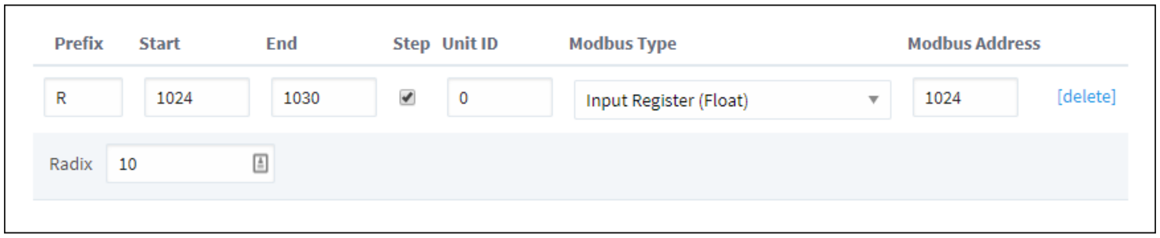
Modbus only supports reading and writing to memory types of bits and 16-bit words. This is not very useful when reading from or writing to float point or 32-bit integers.

To workaround this problem, the Modbus driver is designed to read 2 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** c olumn checked, the addresses on the Ignition side will index by 2. In this case, R1024, R1026, R1028 and R1030 will be created.

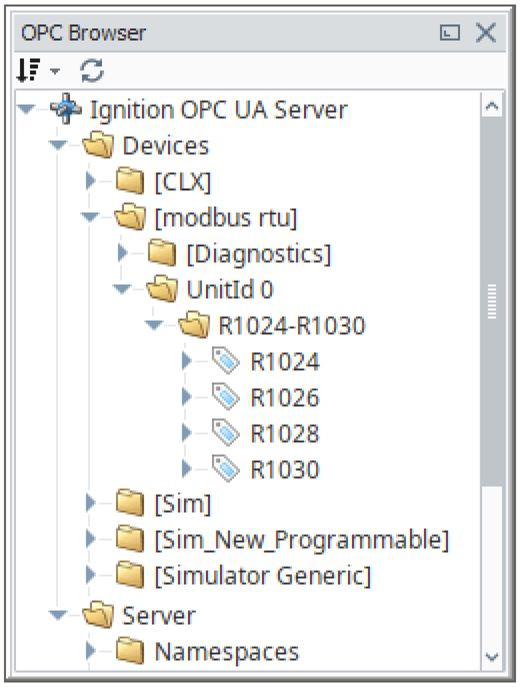
Because **Modbus Type of Holding Register (Float)** is selected, the driver will read two consecutive 16-bit words and convert it to a floating point value. It also indexs the Modbus Address by 2 for each entry. In this case, R1024 reads from Modbus addresses 1024 and 1025 and converts them into a floating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



Prefix	Start	End	Step	Unit ID	Modbus Type	Modbus Address	
R	1024	1030	✓	0	Input Register (Float)	1024	[delete]

Radix 10

The following window shows what is displayed in the OPC Browser. 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 OPC Browser to match the addresses in the device.



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

Overview

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

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

Considerations for 1200 and 1500 Devices

The following considerations and configurations changes must be made when using the S7-1200 and S7-1500 drivers:

1. Only global DBs can be accessed.
2. The optimized block access must be turned off.
3. The access level must be "full" and the "connection mechanism" must allow GET/PUT.
4. Reads and Writes can not be used in TM/CT areas.

Connect to a Siemens Device

Connecting to S7 Devices

Watch the Video

1. Go to the **Config** section of the Gateway Webpage.
2. Scroll down and select **OPC UA > Device Connections**.
3. On the **Devices** page, click on **Create new Device**.
4. On the **Add Device Step 1: Choose Type** page, select **Siemens S7-1200**, and click **Next**.
5. There are four modules to choose from
    - **Siemens S7-1500**
      Which connects to Siemens S7-1500 PLCs over Ethernet.
    - **Siemens S7-1200**
      Which connects to Siemens S7-1200 PLCs over Ethernet.
    - **Siemens S7-300**
      Which connects to Siemens S7-300 PLCs over Ethernet.
    - **Siemens S7-400**
      Which connects to Siemens S7-400 PLCs over Ethernet.
6. On the **New Device** page, leave all the default values and type in the following fields:
   Name: **S71200**
   Hostname: type the **IP address**, for example 10.20.4.71
7. You can check the box for **Show advanced properties?** to see the additional settings, but you can keep all the defaults.
8. Click **Create New Device**.
   The **Devices** page is displayed showing the **Siemens** device is successfully created and added to Ignition. The **Status** will show as Disconnected and then Connected.

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

Configuring Siemens Addressing

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

To manually specify each address

1. From the Designer, in the **Tag Browser**, right-click on **Tags** folder, and then go to **New Tag > OPC Tag**.
2. 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 Configuring Siemens Addressing section, explains how your can construct these addresses.
3. Click **OK**.
   Now you can see the **Temp** tag in the **Tag Browser**.

INDUCTIVE UNIVERSITY

About Siemens Addressing

Watch the Video

Address Syntax

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

Tag addresses are made up of three different components: **Area**, **DataType**, and **Offset**.

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

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

Examples	
Area+Data Type+Offset	

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

Offsets

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

Bits

Bits are addressed by using the Bit DataType (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 DataType 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 DataType is not specified when accessing counters.

Device Settings

General	
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	
Hostname	The hostname or IP address of the device.
Timeout	The request timeout, specified in milliseconds. The default is 2,000.
Advanced	
Port	The port to use when connecting to the device. The default is 102.
PDU Size	Number of bytes to fit into PDU block of a single packet. Increasing this number can improve request throughput only if the processor supports a higher PDU Size. Varies from 240 and up to 960 depending on the device. The default is

	240.
Rack Number	The number of the rack that the device is positioned in. The default is 0.
CPU Slot Number	The slot number assigned to the CPU. The default is 2.
Reconnect After Consecutive Timeouts	After several consecutive timeouts, the Device Connection will attempt to reconnect to the device. This setting determines how many consecutive timeouts must occur before reconnecting.

UDP and TCP Driver

What Is a TCP Connection?

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

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

Structure in the Address Space

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

Connecting to a Device

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

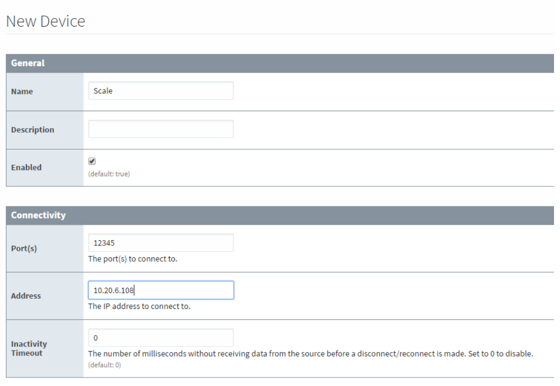
Connecting to TCP Device

Watch the Video

Connect to a Barcode Scanner or Scale

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

1. Go to the **Config** section of the Gateway Webpage.
2. Scroll down and select **OPC UA > Device Connections**.
3. On the **Devices** page, click on **Create new Device**.
4. On the **Add Device Step 1: Choose Type** page, scroll all the way down and select **TCP Driver**, and click **Next**.
5. On the **New Device** page, leave all the default values and type in the following fields:
   Name: the name you specify here will appear under **Devices** folder on the **Quick Client** p age in the **Gateway.**
   Port(s): **12345**, as an example
   Hostname: type the **IP address**, for example 10.20.6.108



New Device

General	
Name	Scale
Description	
Enabled	✓ (default: true)

Connectivity	
Port(s)	12345 The port(s) to connect to.
Address	10.20.6.108 The IP address to connect to.
Inactivity Timeout	0 The number of milliseconds without receiving data from the source before a disconnect/reconnect is made. Set to 0 to disable. (default: 0)

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

7. Click **Create New Device**.
   The **Devices** page is displayed showing the **Scale** device is successfully created and

added to Ignition. The **Status** will show as **1/1 Connected**.

8. Go to the **OPC Connections > Quick Client** in the **Configure** section of the Gateway, under the **Devices>[Scale]>12345** folder you will see the **Last Receive Time** and the **Message** folders.
   Next to each tag, under the **Action** column, you will see `[s][r][w].`

9. Click on `[s]` which means **Subscription**. You will be able to see the **Value** of the tag displayed on this **OPC Quick Client** page.

Device Properties

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

General	
Name	Name of the device using this driver. This name will appear in the Devices folder when browsing the OPC-UA server.
Enables	When selected, the device is enabled. When not selected, disabled devices will not make a connection attempt.
Connectivity	
Port(s)	On the UDP driver, this is the port(s) to listen on. On the TCP driver, this is the port(s) to connect to. Separate multiple ports with a comma.
Address	On the UDP driver, this is the IP address to listen to. On the TCP driver, this is the IP address to connect to.
Inactivity Timeout	The number of milliseconds without receiving data from the source before a disconnect/reconnect is made. Set to 0 to disable.
Message	
Message Delimiter Type	Sets the method used to determine how much or what data length constitutes a full **message**. **Packet Based:** Assumes that whatever arrives in one packet, regardless if length or content, is the message. **Character Based:** Content is appended to a message buffer until the given character or set of characters arrives, at which point the contents of the buffer are considered the message. **Fixed Size** Content is appended to a message buffer until some fixed number of bytes is received, at which point the contents of the buffer are considered the message.
Message Delimiter	If the message delimiter type is **Character Based**, this will be the character or set of characters used to identify a message. If the type is **Fixed Size**, this will be the size used to identify a message.
Field Count	The number of fields within a message must be fixed. This property dictates how many fields will be present in each message. When the number of fields received does not match the designated count, all nodes will receive quality BAD_CONFIG_ERROR.
Field Delimiter	This is the character(s) that are used as field delimiters. For example, the message **a|b|c|d** with a field delimiter of "**|"** would be split into four fields: **a**, **b**, **c**, and **d**. The field count would have to be set at 4.

Both drivers have unique Advanced Properties

TCP Driver	
Writeback Enabled	Enable writeback capabilities for the device.
Writeback Message Delimiter	The delimiter expected by the device signaling the end of an incoming message.
UDP Driver	
Message Buffer Size	The size of the message buffer in bytes.

Multicast	If the connection should be enabled for multicast.

Tag Polling Speed

Tag values update no faster than the scan class a Tag is assigned to. If additional messages are received over TCP or UDP at a faster rate, the Tag will not show these additional values. Generally it is recommended to use these drivers with updates that come through at a 1 second rate or slower. The absolute maximum rate for the TCP driver is 25ms, although most systems cannot reliably execute this quickly. The UDP driver does not have a maximum rate, but a practical maximum rate will depend on hardware, and likely will be significantly slower than 25ms as well. Most "fast" scan rates fall in the 100ms-200ms range.

DNP3

What Is DNP3?

DNP3 is a protocol used commonly in utilities like water and electric companies. It is commonly used to connect to one master station (device) that is then connected to several other devices. This creates a web of devices without taxing the network too much. DNP3 is similar to the Modbus protocol in that is it more device agnostic, but it's newer, more robust, and because of that, more complex. You can use it to connect to many modern devices, check your documentation to see whether your device supports DNP3.

On this page ...

INDUCTIVE UNIVERSITY

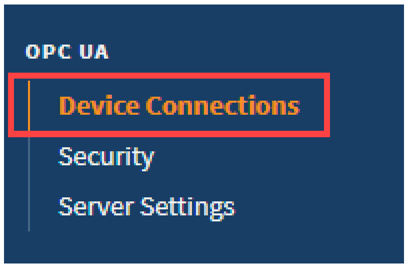
Connecting to DNP3 Devices

Watch the Video

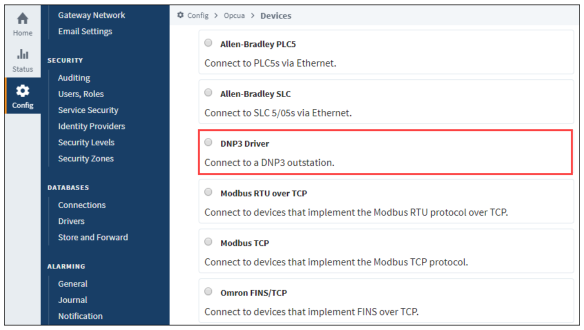
Connecting to a Device

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



3. On the **Devices** page, find the blue arrow and click on **Create new Device**.
4. On the **Add Device Step 1: Choose Type** page, select **DNP3 Driver**, and click **Next**.



Allen-Bradley PLC5
Connect to PLC5s via Ethernet.

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

DNP3 Driver
Connect to a DNP3 outstation.

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

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

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

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

   Name: **DNP3**
   Hostname: Enter IP address, for example 10.20.8.51

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

7. Click **Create New Device**.
   The **Devices** page is displayed showing the **DNP3** device is successfully created and added to Ignition. The **Status** will show as Disconnected and then Connected or Idle, depending on the status of the device.

Connection Settings

General	
Name	The name of this DNP3 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	
Hostname	The IP Address of the device.
Port	The port to use when connecting to a DNP3 device. The default port is 20000.
Source Address	The address of the master station, default is 3.
Destination Address	The address of the outstation, default is 4.
Integrity Poll Interval	The interval at which to perform an integrity poll, in millis, default is 3,600,000.
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.
Advanced	
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.
Link Layer Confirmation	When true, a link layer confirmation will be required from the outstation when sending messages, default is false
Default Outstation Conformance level	The default DNP3 Application Layer level subset to use when communicating with the outstation.
Default Value Types	
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. defaultis 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

Notes

- **Source Address and Destination Address**: These addresses are assigned to the computers and should be the same across all settings, because of this the settings in Ignition and the settings in the device are the opposite of each other. For example, if the device is configured with an address of 4 and Ignition has an address of 3:
    - the settings in Ignition should have the Source Address set to 3 and the Destination Address set to 4.
    - the settings in the Device should have the Source Address set to 4 and the Destination Address set to 3.

- **Unsolicited messages enabled** property: setting this property to True (checking the box) allows the outstation to send unsolicited messages to Ignition. This means that 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.

Internal Indicators

Each response received from a connected outstation will contain an Internal Indication (IIN) bit field. This field indicates certain states or error conditions in the outstation. IINs are mapped to read-only points, indicating the following:

- Broadcast message received (**Broadcast**)
- Additional Class 1, 2, or 3 event data is available (**Class 1 Events, Class 2 Events, Class 3 Events**)
- Time synchronization required in the outstation (**Need Time**)
- Some output points are in local mode (**Local Control**)
- An abnormal condition exists (**Device Trouble**)
- The outstation device has restarted (**Device Restart**)
- Function code not implemented (**No Func Code Support**)
- Object Unknown (**Object Unknown**)
- Request parameter error (**Parameter Error**)
- Outstation event buffer overflow (**Event Buffer Overflow**)
- An operation is already executing (**Already Executing**)
- Configuration corrupt (**Config Corrupt**)

Terminology

- **unsolicited response**: An Application Layer message from an outstation to a master for which no explicit request was received. The request is implied by the act of a master enabling unsolicited reporting of various points within an outstation.
- **integrity poll**: Requests all event data, followed by the static data of all points assigned to one of the four classes (static Class 0 or event Class 1, 2, or 3).
- **DNP3TIME**: Univeral Coordinated Time (UTC) time expressed as the number of milliseconds since the start of January 1, 1970. The effective date for using the UTC time base is January 1, 2008. Prior to this, DNP3 did not require a specific time reference.

Aliased Points

Aliased points allow the user to assign meaningful names and descriptions to DNP3 points. They are also useful for addressing any points that were not returned by the initial integrity-poll on connection.

Point Address	The group, variation, and index that fully describe a point. A full address consists of all three parts: • Group – An integer prefixed with **`g`**. For example, **`g40`** • Variation – An integer prefixed with **`v`**. For example, **`v2`** • Index – An integer prefixed with **`i`**. For example, **i5** Example: **`g30v1i20`**
Path	A “/” separated folder hierarchy in which to create the aliased point. Example: **`Facility1/Voltage`**
Description	A user-defined description of the point mapping.

The following feature is new in Ignition version **8.0.13**
Click here to check out the other new features

Buffered Events

Buffered events can be enabled by setting the Queue Size property on the corresponding Tag Group to a number greater than 1. (See Queue Size on the Tag Groups page.) The number represents the number of buffer events that will be handled by the driver. The queue always keeps the most recent events, dropping older events. For example, if Queue Size were set to 20, and 30 value changes occur on the device while disconnected, then the 10 oldest events would be ignored by the driver.

After recovering from a disconnect, the driver will playback missed events from the driver, and update the value on the corresponding Ignition Tags by cycling through each event quickly, from oldest event in the buffer to the most recent. This value change will trigger value changes in certain systems. Specifically:

- alarms events (recent evens as well as alarm journal records)
- Tag History records (when the History **Sample Mode** is set to "On Change")
- Tag Events Scripts

Once finished cycling through the buffered values, the Tag will resume showing the live value.

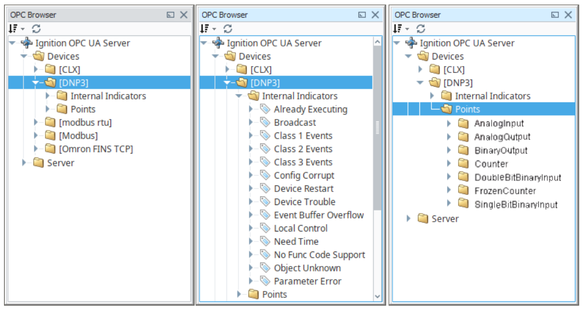
Support for this Feature

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

Browsing DNP3 Points

When the driver (master) connects to a device (outstation), an integrity poll is performed. Any DNP3 objects returned in the response that fall under the Point Type categories listed in the table on the right are mapped to the OPC server with the appropriate index. (For example, g40v1i2 corresponds to an AnalogOutput point, variation 1, index 2.)

To see the points mapped, you can go to the Designer, open the OPC Browser, and drill down to the DNP3 connection node.



INDUCTIVE UNIVERSITY

About DNP3 Addressing

Watch the Video

Point Types

Type Name	Group	Supported Variations	
SinglBitBinaryInput	1	1 - Packed format	2 - With flags
DoubleBitBinaryInput	3	1 - Packed format	2 - With flags
BinaryOutput	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 9 - 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
AnalogOutput	40	1 - 32-bit with flags 2 - 16-bit with flags	3 - Float with flags 4 - Double with flags
OctetString	110	0 - 255	

Omron NJ Driver

Connect Ignition to an Omron NJ Device

1. Go to the **Config** section of the **Gateway** Webpage.
2. Scroll down and select **OPC UA > Device Connections**.
3. On the **Devices** page, find the blue arrow and click on **Create new Device**.
4. On the **Add Device Step 1: Choose Type** page, select **Omron Driver**, and click **Next**.
5. On the **New Device** page, leave all the default values and type in the following fields:

   Name: **Omron**
   Hostname: type the IP address, for example 74.125.224.72
   Check the box for **Show advanced properties?** to see the additional settings, but you can keep all the defaults.

6. Click **Create New Device**.
   The **Devices** page is displayed showing the **Omron** device is successfully created and added to Ignition.

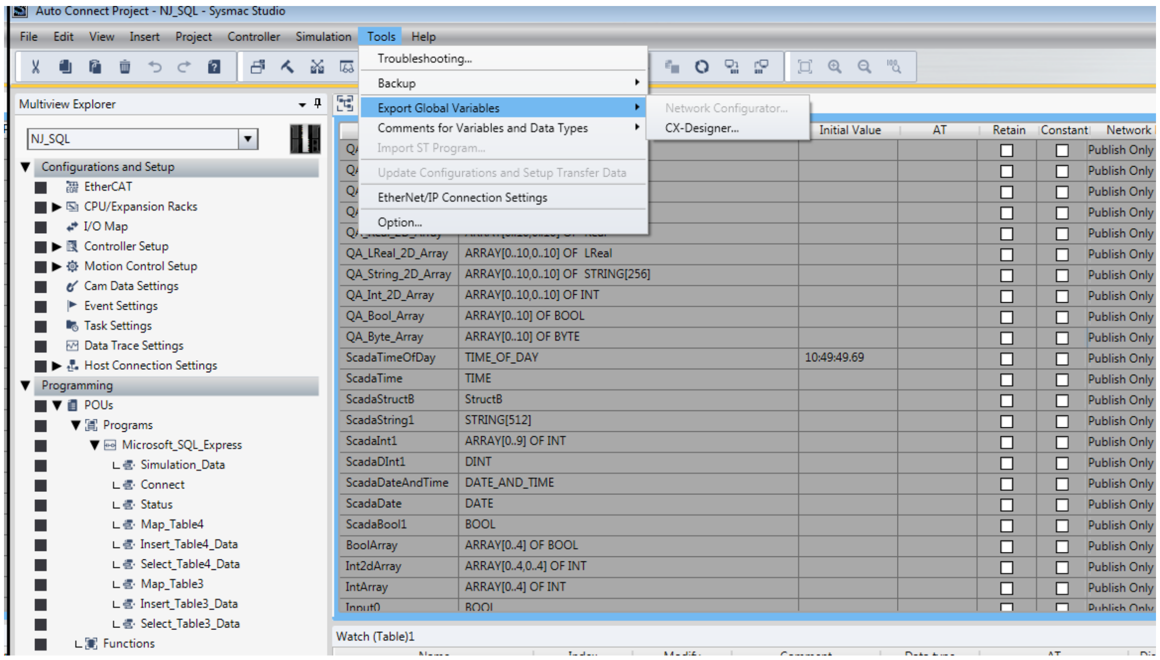
7. On the **Devices** page, click the Tags link next to the newly created device.
   The **Manage Tags** page is displayed, allowing you to configure which variables in the device will show up as Tags in Ignition.

Device Settings

General	
Name	Device name.
Description	Decription of the device connection.
Enabled	Default is set to true.
Main	
Hostname	The hostname or IP address of the device.
Timeout	The request timeout, specified in milliseconds. The default is 2,000.
Concurrency	The number of concurrently issued requests allowed. There is a 1:1 correlation between concurrency and the number of CIP connections used.
Advanced	
Connection Size	The CIP connection size to use. The default (and maximum) is 1,994 bytes.
Slot Number	The slot number in the backplane in which the CPU is located.

Exporting from the Device

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

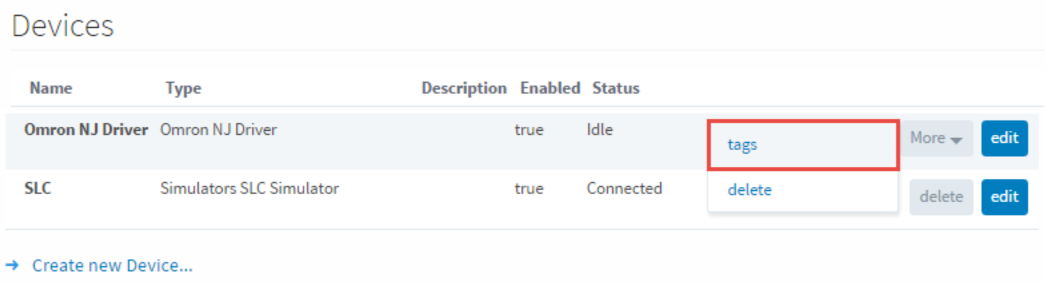


		Initial Value	AT	Retain	Constant	Network
QA_LReal_2D_Array	ARRAY[0..10,0..10] OF LReal			☐	☐	Publish Only
QA_String_2D_Array	ARRAY[0..10,0..10] OF STRING[256]			☐	☐	Publish Only
QA_Int_2D_Array	ARRAY[0..10,0..10] OF INT			☐	☐	Publish Only
QA_Bool_Array	ARRAY[0..10] OF BOOL			☐	☐	Publish Only
QA_Byte_Array	ARRAY[0..10] OF BYTE			☐	☐	Publish Only
ScadaTimeOfDay	TIME_OF_DAY	10:49:49.69		☐	☐	Publish Only
ScadaTime	TIME			☐	☐	Publish Only
ScadaStructB	StructB			☐	☐	Publish Only
ScadaString1	STRING[512]			☐	☐	Publish Only
ScadaInt1	ARRAY[0..9] OF INT			☐	☐	Publish Only
ScadaDInt1	DINT			☐	☐	Publish Only
ScadaDateAndTime	DATE_AND_TIME			☐	☐	Publish Only
ScadaDate	DATE			☐	☐	Publish Only
ScadaBool1	BOOL			☐	☐	Publish Only
BoolArray	ARRAY[0..4] OF BOOL			☐	☐	Publish Only
Int2dArray	ARRAY[0..4,0..4] OF INT			☐	☐	Publish Only
IntArray	ARRAY[0..4] OF INT			☐	☐	Publish Only
Input0	BOOL			☐	☐	Publish Only

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

Managing Tags

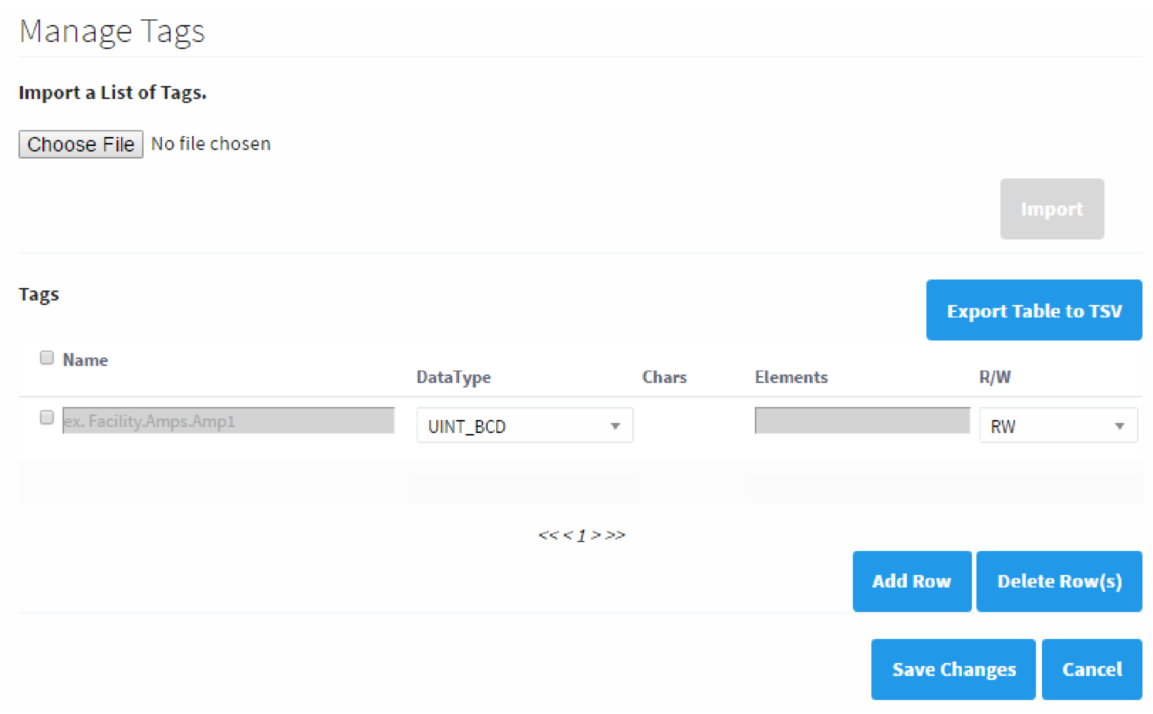
In order to browse Tags in the Designer, you must first create a mapping for the device in the Gateway. The **Manage Tags** page can be accessed by navigating to the Omron device and clicking the **Tags** link.



Name	Type	Description	Enabled	Status	
Omron NJ Driver	Omron NJ Driver		true	Idle	tags
SLC	Simulators SLC Simulator		true	Connected	delete

Importing Tags

Once on the **Manage Tags** page, you can manually enter the Tags, or import them from a tab-separated file. When importing, first choose a file, then click the **Import** button. The default option when importing is to replace the **Tags** table with Tags from the import. Select the append option to append Tags to the table.



Manage Tags

Import a List of Tags.

Choose File No file chosen

Import

Tags

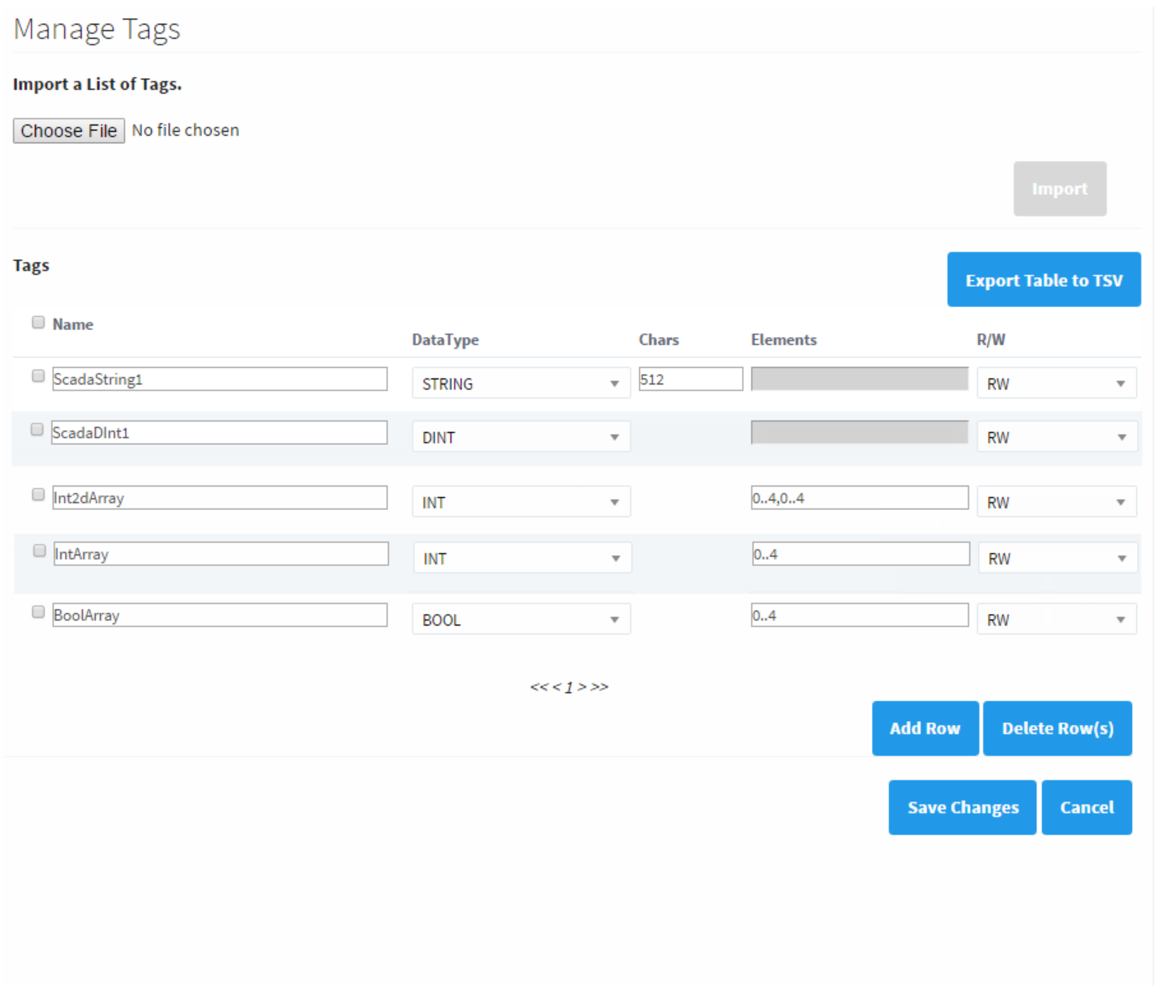
Export Table to TSV

Name	DataType	Chars	Elements	R/W
ex. Facility.Amps.Amp1	UINT_BCD			RW

<< < 1 > >>

Add Row Delete Row(s)

Save Changes Cancel



Manage Tags

Import a List of Tags.

Choose File No file chosen

Import

Tags

Export Table to TSV

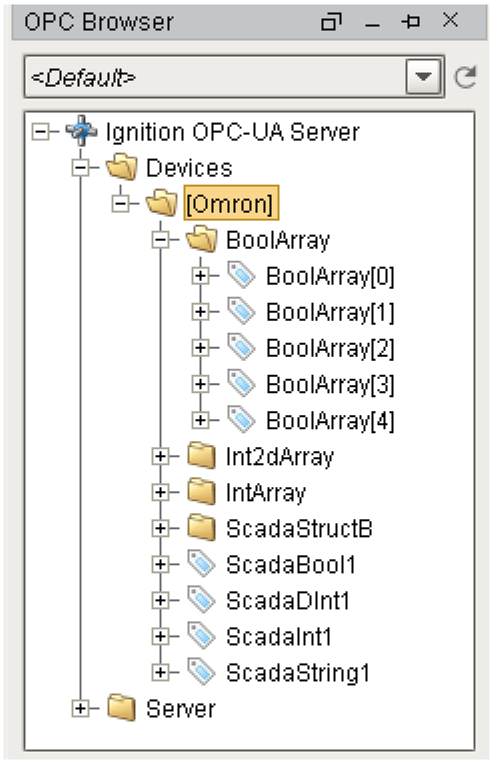
Name	DataType	Chars	Elements	R/W
ScadaString1	STRING	512		RW
ScadaDInt1	DINT			RW
Int2dArray	INT		0..4,0..4	RW
IntArray	INT		0..4	RW
BoolArray	BOOL		0..4	RW

<< < 1 > >>

Add Row Delete Row(s)

Save Changes Cancel

Once you save any changes made to the Tag mapping, you can view the Tags in the Designer OPC Browser.



Addressing

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

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

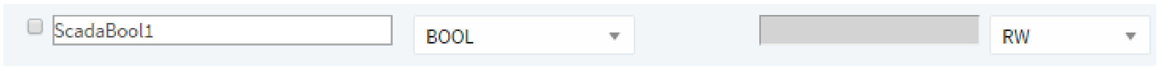
The following feature is new in Ignition version **8.0.14**
Click here to check out the other new features

In 8.0.14, support for the following data types was added:

- TIME_NSEC
- DATE_NSEC
- TIME_OF_DAY_NSEC
- DATE_AND_TIME_NSEC

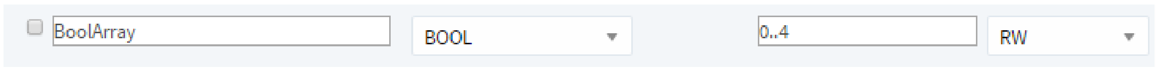
Scalars

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..4 elements is a 5 element array.


BoolArray BOOL 0..4 RW

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_

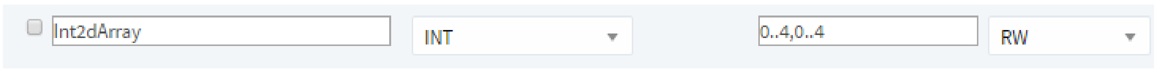
This is because the driver always assumes that the lowest configured element on the mapping page matches up with the lowest element in the PLC program. As seen above, this can cause some confusion if the mapping on the Ignition Gateway is configured with a different range.

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**.

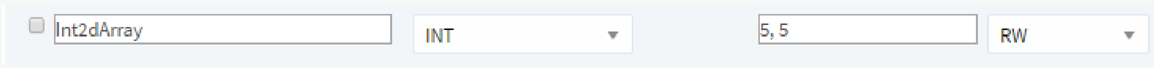
Multidimensional Arrays

Multidimensional arrays are specified in the same way as arrays with each group of indices separated by a comma.


Int2dArray INT 0..4,0..4 RW

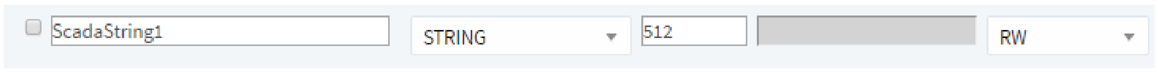
Optional Format

Array elements may also be specified with a single number equaling the total number of elements.

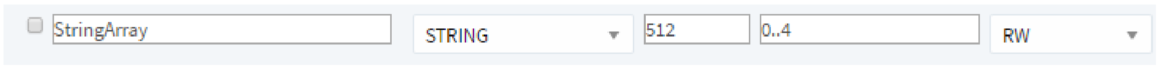

Int2dArray INT 5, 5 RW

Strings

The number of characters for String variables is specified in the **Chars** field.


ScadaString1 STRING 512 RW

String arrays are mapped using both the **Chars** and **Elements** fields.


StringArray STRING 512 0..4 RW

Omron FINS Driver

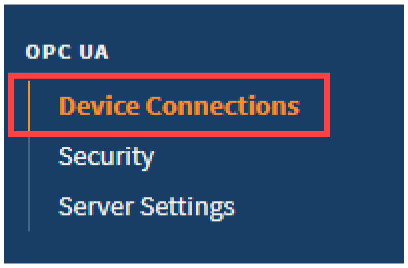
The following feature is new in Ignition version **8.0.8**
Click here to check out the other new features

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

On this page ...

Connect Ignition to an Omron FINS Device via TCP

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



3. On the **Devices** page, click on **Create new Device**.
4. Select **Omron FINS/TCP**, and click **Next**.
5. Fill in the following fields:

   Name: **FINS TCP**
   Hostname: Enter the **IP address**, for example 74.125.224.72

   Leave the default values in the remaining fields.

6. Click **Create New Device**.

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

Connect Ignition to an Omron FINS Device via UDP

1. Go to the **Config** section of the **Gateway** Webpage.
2. Scroll down and select **OPC UA > Device Connections**.
3. On the **Devices** page, click on **Create new Device**.
4. Select **Omron FINS/UDP**, and click **Next**.
5. Fill in the following fields:

   Name: **FINS UDP**
   Bind Address: Enter the **IP address** of your Gateway, for example 74.125.225.10
   Remote Address: Enter the **IP address** of your Omron FINS device, for example 74.125.224.72

6. Under FINS Settings set the following:

   FINS Source Node: The last octet of your IP address, for example 10 using the bind address above.
   FINS Destination Network: The address number of your FINS device typically configured in the routing table for your device. The valid range is 0-128.
   FINS Destination Node: The node number of your FINS device. This is typically the last octet of the IP address of the device, for example 72 in the example above.
   Fins Destination Unit: The unit number of your FINS device.

7. Click **Create New Device**.

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

Device Settings

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

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

FINS Settings		
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	
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.
	The maximum size of a request in bytes.

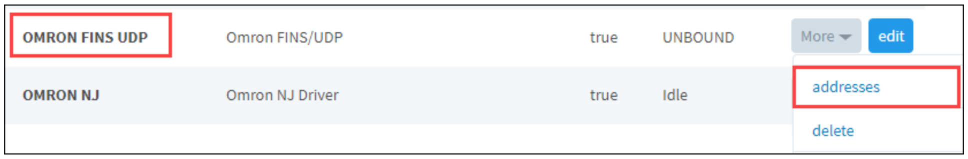
Max Request Size	
Max Gap Size	The maximum address “gap” to span when optimizing requests to contiguous address locations.
Write Priority Ratio	The number of write requests to execute for every read request, when an abundance of both are queued for execution.

The following feature is new in Ignition version **8.0.11**
Click here to check out the other new features

Importing Addresses

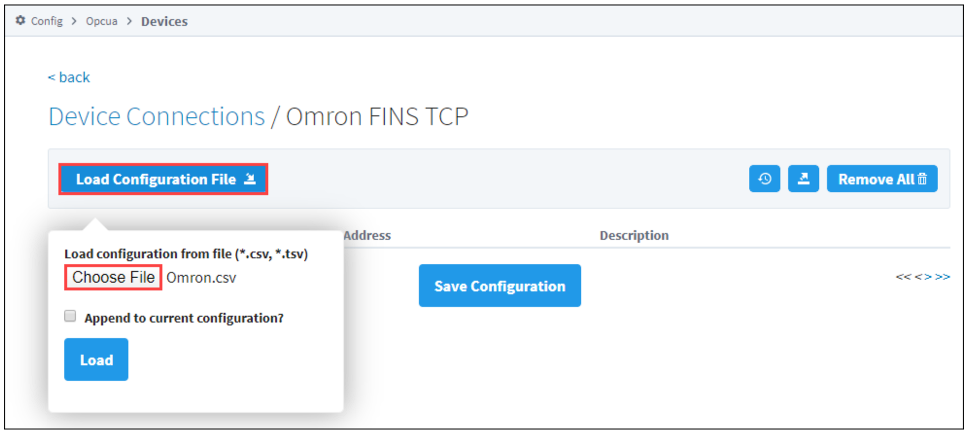
As of version 8.0.11 a user interface was added to create import addresses. After import, you'll be able to browse the Tags in the Designer.

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

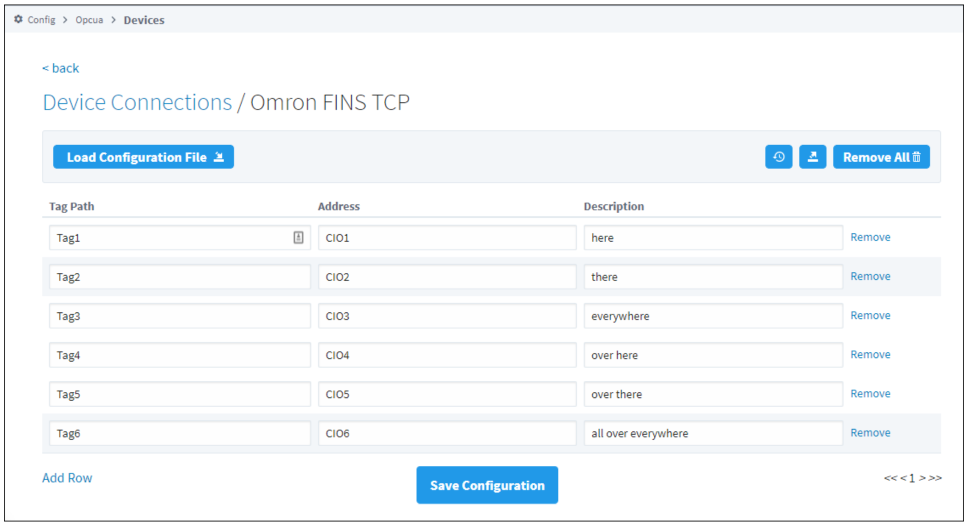


OMRON FINS UDP	Omron FINS/UDP	true	UNBOUND	More edit
OMRON NJ	Omron NJ Driver	true	Idle	addresses delete

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



You'll see the values load onto the screen.

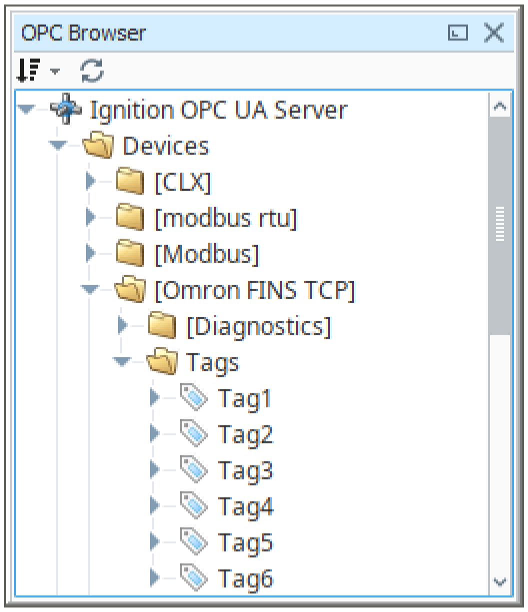

Config > Opcua > Devices

< back

Device Connections / Omron FINS TCP

Tag Path	Address	Description	
Tag1	CIO1	here	Remove
Tag2	CIO2	there	Remove
Tag3	CIO3	everywhere	Remove
Tag4	CIO4	over here	Remove
Tag5	CIO5	over there	Remove
Tag6	CIO6	all over everywhere	Remove

6. Click **Save Configuration**.
7. Once you save any changes made to the Tag mapping, you can view the Tags in the Designer OPC Browser.



Addressing

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

Data Areas

Area	Ignition Code	Omron Symbol Code	Native Size
CIO Area	CIO		Int16/UInt16
Work Area	WR or W	W	Int16/UInt16
Holding Bit Area	HR or H	H	Int16/UInt16
Auxiliary Bit Area	AR or A	A	Int16/UInt16

Timer Area	TIM or T	T	Int16/UInt16 or Bool
Counter Area	CNT or C	C	Int16/UInt16 or Bool
Index Register Area	IR	IR	Int32/UInt32
Data Register Area	DR	DR	Int16/UInt16
DM area	DM or D	D	Int16/UInt16
EM area	EM or Exx	E	Int16/UInt16

EM Area

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

Data Types

Bit	Bool	Int16
Int32	Int64	UInt16
UInt32	Float	Double
String	BCD16	BCD32

Data Type Modifiers

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

Examples

Syntax Example

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.

Ignition OPC UA Server Node Configuration

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

File Name: `tags.csv`
File Location: `$ignitionHome/data/opcua/devices/$deviceName`
File Format: Comma separated value file with the following columns:

- Browse Path
- Address
- Description

Example File

```
Tag1,           CIO0, "First tag"
Foo/Tag2,       CIO1, "Second tag, in a folder"
Foo/Bar/Tag3,   CIO2, "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 following feature is new in Ignition version **8.0.8**
Click here to check out the other new features

Overview

On this page

...

The **Programmable Device Simulator** introduced in Ignition 8.0.8 replaces the previous built-in simulators in earlier versions of Ignition, and combined the Dairy Demo, Generic and SLC simulator device connections into a single driver. It also provides new functionality that allows you to create your own simulator program with outputs you define. Users are able to create tags that can be used in their own development process, thus providing the flexibility to simulate devices with a tag structure they are building for their project. There are a number of built-in functions that will result in unique tag outputs rather than static values. We also provided the ability to set specific values at different points of the program should the functions not provide exactly what you need.

Because the Programmable Device Simulator replaces the existing simulators that were included with prior versions of Ignition, we added the ability to quickly create these same simulators with this new functionality. If you delete your existing simulator (prior to Ignition 8.0.8), adding the new Programmable Device Simulator and loading the same program, will allow addresses within the device to map existing addresses.

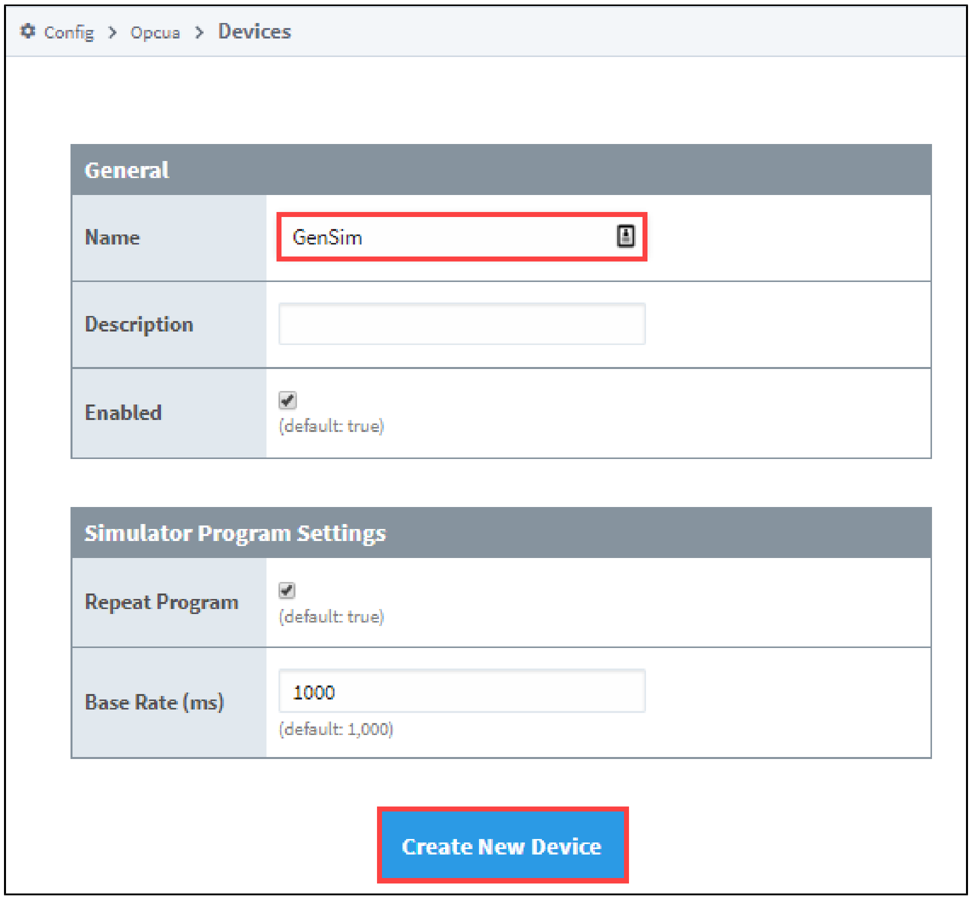
Simulator documentation prior to Ignition 8.0.8

If you are using the Simulator in Ignition version 8.0.7 or before, click here for the documentation.

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

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



Config > Opcua > Devices

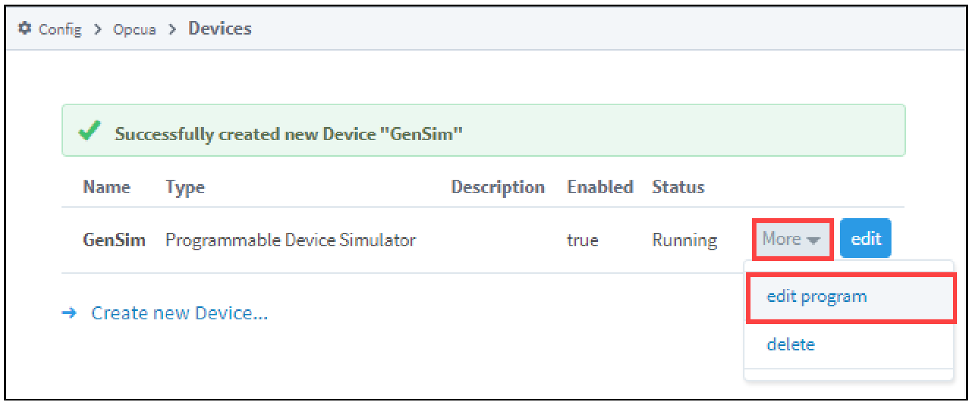
General

Name	GenSim
Description	
Enabled	✓ (default: true)

Simulator Program Settings

Repeat Program	✓ (default: true)
Base Rate (ms)	1000 (default: 1,000)

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

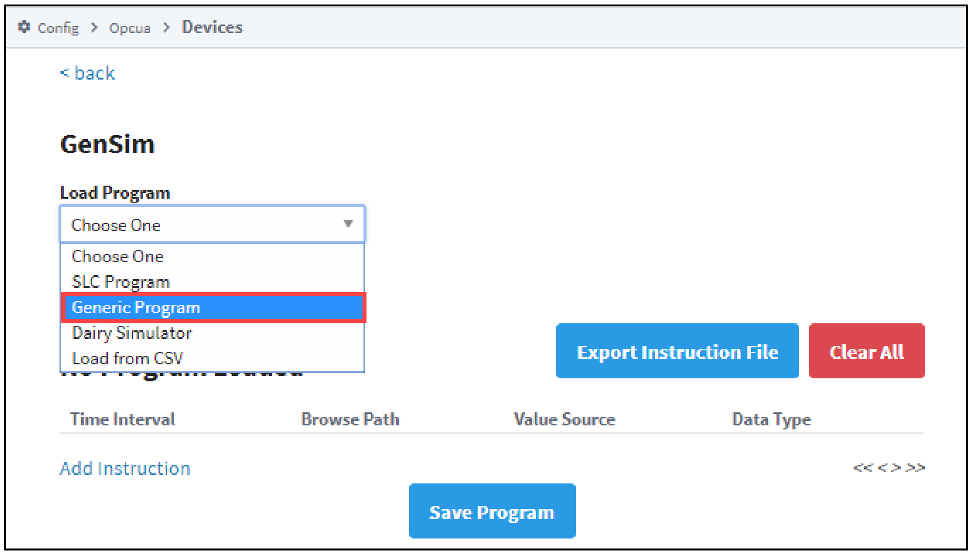


Config > Opcua > Devices

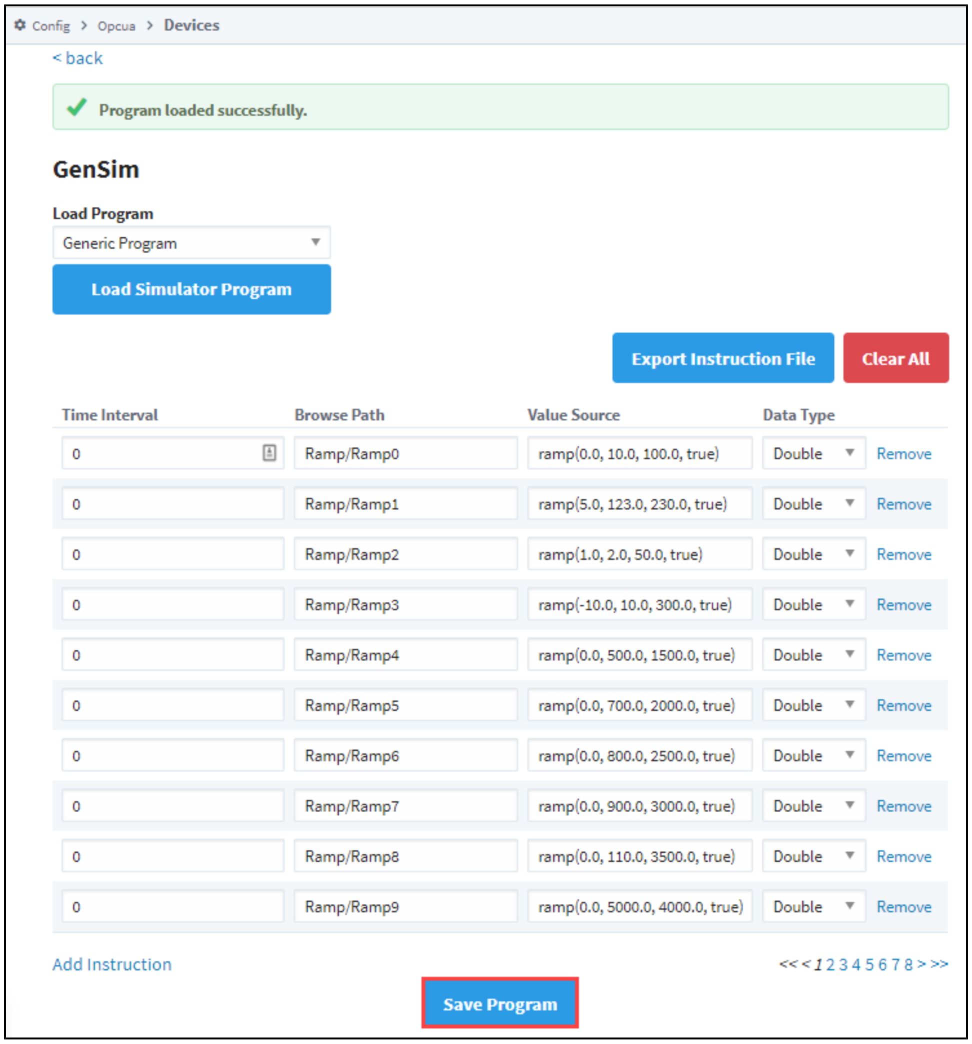
Successfully created new Device "GenSim"

Name	Type	Description	Enabled	Status
GenSim	Programmable Device Simulator		true	Running

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



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



Program loaded successfully.

GenSim

Load Program: Generic Program

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

Simulator Settings

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

Program Instructions

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

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

max, period, repeat)	
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.
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.

Each of the functions above expects 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:

TimeInteval	BrowsePath	ValueSource	DataType

0	TagA	sine()	Double

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

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

Advanced Program Example

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

The program above should yield the following results:

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

Preloaded Programs

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

Generic Simulator Program

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

Dairy Simulator Program

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

SLC Simulator

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

Load From CSV

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

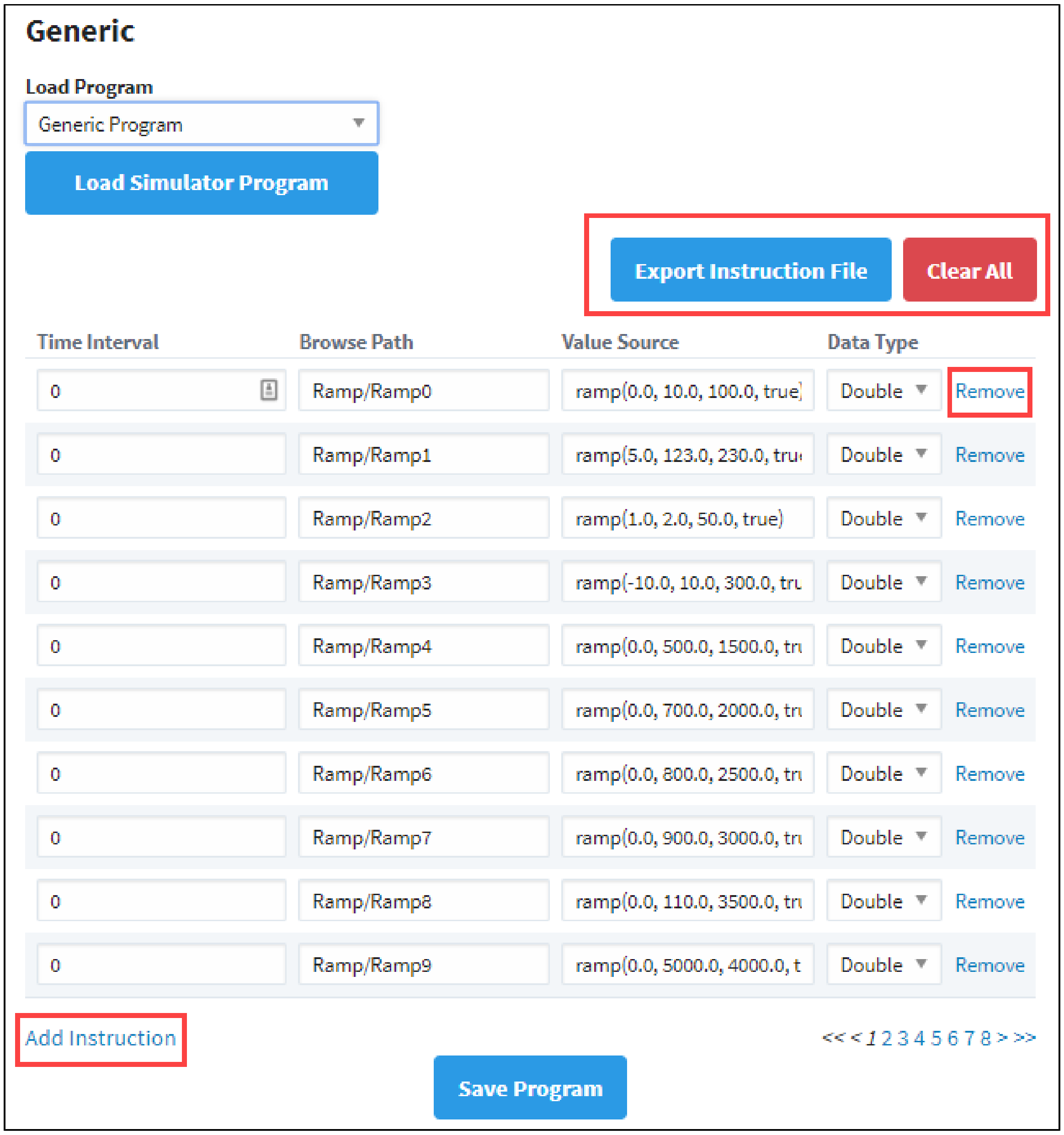
Meta Tags

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

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

Export Instruction File

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



Generic

Load Program

Generic Program

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

BACnet

The following feature is new in Ignition version **8.0.15**
Click here to check out the other new features

On this page ...

BACnet

As of release 8.0.15, 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.

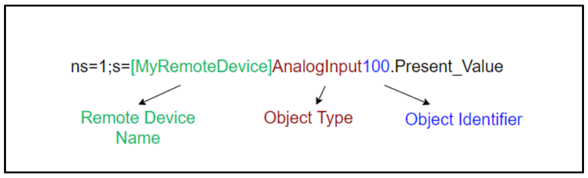
Setup Process

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

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. The image below calls out some important parts of the item path.



BACnet Addressing

Watch the Video

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:

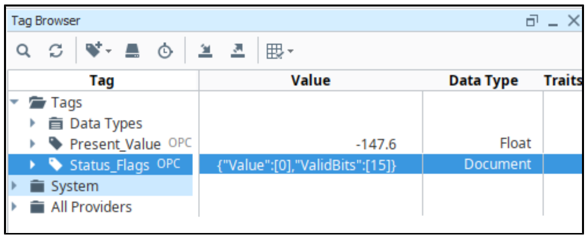
- Device
- AnalogInput
- AnalogOutput
- AnalogValue
- LargeAnalogValue
- BinaryInput
- BinaryOutput
- BinaryValue
- MultiStateInput
- MultiStateOutput
- MultiStateValue

Object Identifier

The numerical identifier of the object.

Status_Flags and Document Tags

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



Tag	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 and getBit 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)
```

In This Section ...

BACnet Local Device

The following feature is new in Ignition version **8.0.15**
Click here to check out the other new features

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.

On this page ...

Device Configuration Strategies

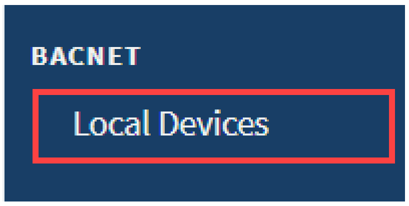
One strategy for configuring local devices is to configure one per network adapter that will be used for communication between the Ignition Gateway and remote BACnet devices.

You can also configure a local device to bind to a wildcard address such as 0.0.0.0, and in fact, this is required to receive broadcast packets from any of the remote devices you intend to communicate with. We’ve found that some devices will only respond to the BACnet Who-Is request with an I-Am that is sent to a broadcast address regardless of whether the original Who-Is was sent unicast or broadcast.

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

Configure a Local BACnet Device

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

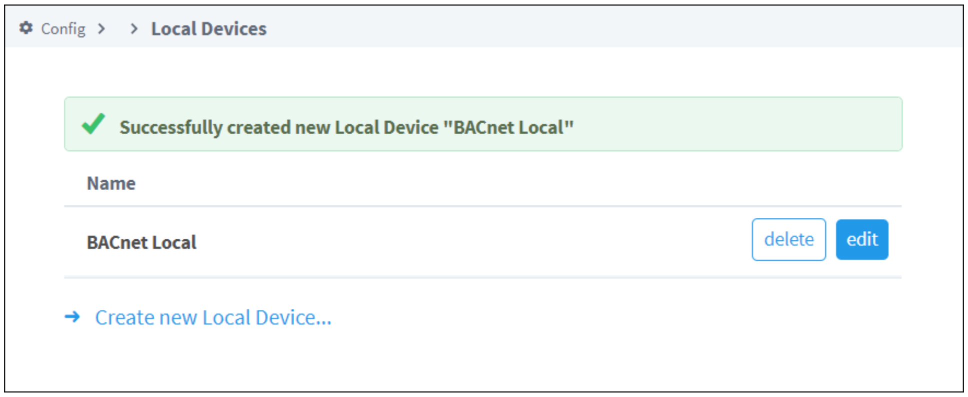


3. On the **Devices** page, click on **Create new Device**.
4. Select **BACnet/IP**, and click **Next**.
5. Fill in the following fields:

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

6. Leave the default values in the remaining fields.

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


Config > > Local Devices

Successfully created new Local Device "BACnet Local"

Name	
BACnet Local	delete edit

Create new Local Device...

Now you can configure a driver for the remote device.

Device Settings

General	
Name	Device name.
Bind Address	The local IP address to bind to. If this Local Device is going to receive broadcast packets (as opposed to unicast) from a Remote Device connection, then this must be set to a wildcard address (0.0.0.0).
Bind Port	This is the port the Device you are setting up will listen on. The local port to bind to. Default is 47,808.
Broadcast Address	A network address shared by other BACnet devices to send and receive UDP data.
Network Prefix Length	Default is 24.
Device Number	Default is 1,000.
Network Number	Default is 1.
Foreign Device Registration Enabled	If true, register as a foreign device with the configured BBMD (BACnet Broadcast Management Device). Default is false.
BBMD Address	Address of the BACnet Broadcast Management Device (BBMD) to register with. If you're planning on having this Local Device configuration work in conjunctions with a Remote Device via BBMD, then the address here should be the address of the BBMD on the same subnet as the Remote Device.
BBMD Port	Port the BACnet Broadcast Management Device to register with is listening on. Default is 47,808.

BACnet Remote Device

The following feature is new in Ignition version **8.0.15**
Click here to check out the other new features

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

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

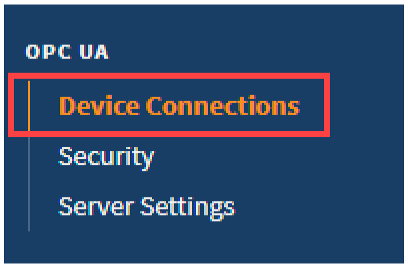
The object and properties are cached and subsequent connection attempts will only re-read them if the Database_Revision property changes. You can invalidate the browse data under the “More” menu for a given device (Config > OPC UA > Device Connections) on the devices page in the Ignition Gateway.

On this page

...

Connect to a Remote BACnet Device

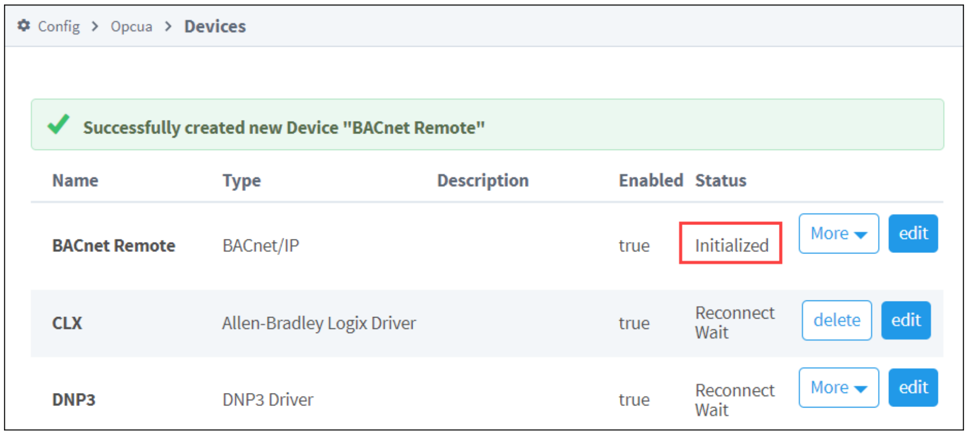
1. Go to the **Config** section of the **Gateway** Webpage.
2. Scroll down and select **OPC UA > Device Connections**.



3. On the **Devices** page, click on **Create new Device**.
4. Select **BACnet/IP**, and click **Next.**
5. Select the local device from the dropdown. If no local devices are shown, you'll need to set one up first. See BACnet Local Device.
6. Fill in the following fields:

   Name: **BACnet Remote**
   Hostname: Enter the **IP address**, for example 10.10.###.##
   Remote Device Number: The instance number of the remote device you're setting up.

7. Leave the default values in the remaining fields.
8. Click **Create New Device**.
9. The **Devices** page is displayed showing the **BACnet/IP** device is successfully created and added to Ignition.



Successfully created new Device "BACnet Remote"

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

Device Settings

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

Connectivity	
Local Device	The name of the local BACnet device instance that will be used when communicating with the remote device. If no local devices are shown, you'll need to set one up first. See BACnet Local Device.
Remote Address	The hostname or IP address of the remote device.
Remote Port	The port that the remote device is listening on. Default is 47,808.
Remote Device Number	The instance number of the remote device you're setting up. Default is 1.
Write Priority	The priority to use when writing to commandable properties. Default is 8.
COV Enabled	If true a Change of Value (COV) subscription is used for properties that support it. If false all properties are polled. Default is true. Note that 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.
Confirmed Notifications Enabled	If true COV subscriptions use confirmed notifications. If false COV subscriptions use unconfirmed notifications. Default is false.