

Ignition 7.9 User Manual — OPC UA and Device Connections

Generated 2026-10-12 from
<https://www.docs.inductiveautomation.com/docs/7.9/>

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

OPC-UA and Device Connections

What is OPC?

OPC is a specification for the transport and use of industrial data. It 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.

History

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

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 instead of DCOM, was platform independent, and combined the various separate specifications into one: 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 in Ignition, makes Ignition act as an OPC-UA server, serving data collected by its built in drivers to other Ignition modules, as well as to third-party OPC-UA clients.

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

data is then used to drive all aspects of the system. Creating connections to OPC-UA servers is described below in Connecting with OPC-UA.

Steve Hechtman's Blog

Click the link below to see the blog:

OPC-UA goes mainstream

Distributed Systems with OPC-UA

OPC-UA breaks down boundaries and enables free data flow. Using standard TCP/IP instead of legacy DCOM, OPC-UA makes it easy to securely transfer data between networks and through firewalls. All OPC-UA connections are based on the same 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 much more connected enterprises.

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

Servers and Clients

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

The Ignition platform inherently offers OPC-UA client functionality. That is, 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, and are not as efficient.

Connecting to Devices with OPC-UA

Connecting to a Device

Here at Inductive Automation, we created our own OPC-UA server and device drivers to connect with. To quickly get connected to one of your devices, go to the **Configure** section of the Gateway webpage and scroll down to **OPC-UA Server > Devices**. The Device page will appear showing all of your installed devices. To add a new device, click on the on the blue link, **Create new Device**. Scroll through the list of devices, and click the radio button next to the type of device connection that you are interested in:

- [Allen Bradley Ethernet](#)
- [Modbus](#)
- [Siemens](#)
- [UDP and TCP Driver](#)
- [DNP3](#)
- [Omron](#)
- [Simulators](#)

Add Device Step 1: Choose Type

☒ **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.

☐ **Legacy Allen-Bradley CompactLogix**

Connect to CompactLogix firmware v20 and prior processors.

☐ **Legacy Allen-Bradley ControlLogix**

Connect to ControlLogix firmware v20 and prior processors.

☐ **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 NJ Driver**

Connect to Omron NJ series PLCs.

☐ **Siemens S7-1200**

Connect to Siemens S7-1200 PLCs over Ethernet.

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** section. You can browse by expanding the tree nodes and read from or write to 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).

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

OPC-UA Connections and Settings

An OPC-UA Connection is used to communicate with an OPC-UA compliant server, such as the one the OPC-UA module provides. The following steps walk-through connecting Ignition (as an OPC-UA client) to a OPC-UA server for versions 7.9.13 and prior. For Ignition version 7.9.14 and greater, see the 7.9.14 section below.

Connecting to a OPC-UA Server - 7.9.13 and Prior

1. On the **Configure** page of the Gateway, go to **OPC Connections > Servers**.

The **OPC Server Connections** page is displayed.

2. Click on the **Create new OPC Server Connection** link.

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

The **Discover OPC-UA Endpoints** page will appear.

4. Enter an OPC-UA endpoint for the OPC-UA Server Ignition should connect to. The format should be as follows:

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

Instead of an IP addresss, a hostname can be used:

```
opc.tcp://myhostname:1234
```

5. Once an endpoint has been entered, click the **Discover** button. A list of available Security Policies and Message Security options will appear.

6. 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 you're connecting to.
7. 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. By default, Ignition's OPC-UA server requires the following credentials:
 - **Username:** opcuauser
 - **Password:** password
8. Once credentials have been entered, click the **Create New OPC Server Connection** button. Ignition is now connected to the OPC UA server.

Connecting to a OPC Server - 7.9.14+

New in 7.9.14 The connection workflow was updated in 7.9.14. The steps below detail UA client connections for more modern versions of Ignition.

1. On the **Configure** page of the Gateway, go to **OPC Connections > Servers**.
The **OPC Server Connections** page is displayed.

2. Click on the **Create new OPC Server Connection** link.

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

The **Server Discovery** page will appear.

4. Type in the endpoint for the server, and click **Next** if you want to use the discovery wizard.

Alternatively, you can click the **Skip to Advanced Configuration** link that allows you to manually enter the Discovery URL, Endpoint URL, and a Endpoint Host Override. You can continue to follow the basic wizard workflow, but users that opted into the Advanced Configuration can skip to Step 8.

Controller

USER MANUAL SUPPORT

admin Sign Out

Launch Designer

Ignition! by inductive automation

HOME STATUS CONFIGURE

Search...

SYSTEM

- Overview
- Backup/Restore
- Licensing
- Modules
- Projects
- Redundancy
- Gateway Settings

NETWORKING

- Gateway Network
- Email Settings

SECURITY

- Auditing
- Users, Roles
- Service Security
- Security Zones

[Skip to Advanced Configuration](#)

Server Discovery

Endpoint URL

Example: opc.tcp://localhost:4096/discovery

Next: Choose Server

5. From here, you'll be shown a list of servers at the endpoint you entered. Select your server and press **Next**.

The screenshot displays the Ignition Controller web interface. The top navigation bar includes 'Controller', 'USER MANUAL', 'SUPPORT', 'admin', 'Sign Out', and 'Launch Designer'. The main navigation menu on the left lists categories: SYSTEM (Overview, Backup/Restore, Licensing, Modules, Projects, Redundancy, Gateway Settings), NETWORKING (Gateway Network, Email Settings), and SECURITY (Auditing, Users, Roles, Service Security, Security Zones). The 'CONFIGURE' tab is active. The main content area shows a link to 'Skip to Advanced Configuration' and a section titled 'Choose a Server'. A light blue box contains the text 'Ignition OPC UA Server' and a radio button selected next to the endpoint 'opc.tcp://1.2.3.4:112233'. Below this are two buttons: 'Previous: Endpoint Discovery' and 'Next: Select Endpoint'.

6. You will then be presented with a list of available security policies and security modes for the server. Select the policy and mode you wish the connection to use, and click **Finish**.
7. You'll arrive at the **Confirm Connection Settings** page, which allows you to review your configuration thus far. If everything is accurate, press the **Finish** button, which exits the wizard, and takes you to the server configuration page.
8. On the **Server Configuration** 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. By default, Ignition's OPC-UA server requires the following credentials:
 - **Username:** opcuauser
 - **Password:** password
9. Once credentials have been entered, click the **Create New OPC Server Connection** button. Ignition is now connected to the OPC-UA server.

OPC-UA Server Properties

Main

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

Authentication

Property	Description
Username and Password	If a username and password are specified, they are used as a user identity token when connecting to the specified OPC-UA server. The internal OPC-UA server provided by the OPC-UA module uses an Ignition security profile to govern who can connect to it. This can be configured in the OPC-UA Server > Settings section.
Change Password?	Enable this property to change the configured password.
Password	Enter password
Password	Re-type password for verification.

Advanced

Property	Description
Host Override	When specified, if the endpoint address returned by the OPC server has a different IP address or hostname than the discovered endpoint, the overridden value will be used. Expects just an IP address or hostname. Example: 192.168.1.10
Connect Timeout	New in 7.9.12 The timeout, in milliseconds, when opening a socket connection to a remote host.
Acknowledge Timeout	New in 7.9.12 The timeout, in milliseconds, to wait for an Acknowledge message in response to the client's Hello message.
Max Per Operation	Specify the maximum number of nodes to read, write, subscribe, or unsubscribe to in any given UA server request.
Max References Per Node	New in 7.9.6 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.
Request Timeout	Maximum amount of time, in milliseconds, to wait for the response to a request. (default: 120,000)
Secure Channel Re-authentication Enabled	(default: true)

Redundancy Settings

Property	Description
Backup Discovery URL	The discovery URL for the backup server's OPC-UA server. Expects the following format: <code>opc.tcp://hostname:port</code>
Backup Endpoint URL	The full endpoint URL for the backup Gateway, e.g. <code>opc.tcp://10.20.1.100:4096/ia/opcua</code> or <code>opc.tcp://192.168.1.10:49320</code>
Backup Host Override	When specified, if the endpoint address returned by the Backup OPC server has a different IP address or hostname than the discovered endpoint, the overridden value will be used. Expects just an IP address or hostname. Example: <code>192.168.1.10</code>
Failover Enabled	When enabled, if this OPC-UA server connection is faulted, requests will be sent to a failover OPC-UA server.
Failover Endpoint	The endpoint of the failover server. <i>Example:</i> <code>opc.tcp://192.168.1.0:4096</code>
Failover Host Override	When specified, if the endpoint address returned by the failover OPC server has a different IP address or hostname than the discovered endpoint, the overridden value will be used. Expects just an IP address or hostname. Example: <code>192.168.1.10</code>
Failover Threshold	The number of retry attempts before the failover connection is used. The default is 5.

❗ HOW DO I DETERMINE THE ENDPOINT FOR MY BACKUP?

See step 4 in the [Connecting to a OPC-UA Server](#) guide above. Clicking the Discover button will show a list of available endpoints to connect to.

Failover Versus Backup Properties

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

NOTE

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

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

OPC Quick Client

You can access the OPC Quick Client from under the **OPC Connections** section of the Ignition Gateway **Configure** section. It allows for quick, simple testing of any devices connected to the server. You can browse by expanding tree nodes and read/write to Tags by clicking on the [r] and [w] buttons next to those Tags.

Subscriptions can be made by clicking on the [s] button. For each individual subscription, you can click on the **SET** button and it will automatically refresh subscriptions and show live value changes (if there are any).

OPC Quick Client

TYPE	ACTION	TITLE
Server	refresh	[-] Ignition OPC-UA Server
Object		[-] Devices
Object		[+] [CLX]
Object		[+] [Dairy]
Object		[-] [Generic]
Object		[-] Ramp
Tag	[s][r][w]	[+] Ramp0
Tag	[s][r][w]	[+] Ramp1
Tag	[s][r][w]	[+] Ramp2
Tag	[s][r][w]	[+] Ramp3
Tag	[s][r][w]	[+] Ramp4
Tag	[s][r][w]	[+] Ramp5
Tag	[s][r][w]	[+] Ramp6
Tag	[s][r][w]	[+] Ramp7
Tag	[s][r][w]	[+] Ramp8

Subscription 1 [x]

Subscription 2 [x]

Subscription 3 [x]

[Add]

Server	Address	Value	Quality	Timestamp
Ignition OPC-UA Server	ns=1;s=[Generic]_Meta:Ramp/Ramp7	110.8	[Good] Good; unspecified.	2/23/17 4:46:53 PM -08:00 [x]

Subscription name :

Rate (ms) :

Set

Ignition OPC-UA Server

You can see the following OPC-UA Server Settings by going to the **Configure** section of the Gateway, and then choosing **OPC-UA Server > Settings**.

Authentication

Setting	Description
Authentication Profile	The User Source that the OPC-UA module will use to authenticate incoming connections against. By default, this is set to the opcua-module User Source. This profile is included in the default installation and has the following as its default settings: user: opcuauser password: password
Allowed Roles	Roles within the given User Source that are allowed to connect to the server. Multiple roles should be separated by a comma, for example, <code>Administrator,user,manager</code> .
Allow Anonymous Access	Allows users to connect whether or not they possess authentication credentials. Not checked by default.

Server

Setting	Description
Server Port	The port on the local machine of the OPC-UA server runs on. Requires a module restart to take effect.
Endpoint Address	This is the local address that the Ignition UA server will bind to. It is also the address that will be used in a GetEndpointResponse, so it is important that this be an address reachable by any clients that wish to connect. Requires a module restart to take effect. This is useful if the server machine has a VPN connection or multiple adapters and is returning the wrong address. Localhost by default
Backup Endpoint Address	This is the local address that the Ignition UA server will bind to <i>on the redundancy backup</i> . Leave blank if not using redundancy.
Minimum Sampling Interval	The fastest rate (in milliseconds) that the server will use to sample its underlying data sources. Requires a module restart to take effect. The default is 100.

Expose Configured Tags

Setting	Description
Expose Tag Providers	If enabled, Ignition Tag providers will be exposed through the OPC-UA server, allowing third party clients to access the Tags configured in the system. By default, the check box is not selected or False.
Auditing Enabled	Enables an Audit profile for OPC-UA server. (default: false)
Audit Profile	If enabled, writes to exposed Tags will be audited to the selected profile.

Other

Setting	Description
Stale Threshold	The multiplier by which the server determines that updates from a driver have become stale. This period will be calculated as the fastest sampling rate for that node multiplied by this settings value. Default is 5.
Auto-cast Writes	Attempt to auto-cast incoming writes to the correct DataType before rejecting them with a <code>Bad_InvalidType StatusCode</code> . By default, the check box is selected or True.
Read Only When Inactive	Operate in read-only mode when this node is the inactive node of a redundant pair.

A Frequently Asked Question

How do I get data from my PLC?

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

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

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

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

Brief summary of device connection in Ignition

- Ignition can only connect directly to devices over Ethernet.
- Ignition can only connect directly to devices for which there is an Ignition device driver. Included drivers are:
 - Allen Bradley - ControlLogix 5500, CompactLogix, MicroLogix 11/1400, PLC-5, SLC 505
 - Siemens - S7-300, S7-400, S7-1200
 - Modbus - The Modbus driver connects to any ethernet enabled device that uses the Modbus protocol.
- 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 Configuration section under the OPC-UA -> Devices page.

1. Click "Add a device..."
2. Select the driver for the device you wish to add.
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 here: [Adding a New Device](#)
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 3rd Party OPC Server via OPC-UA

If your device does not have an Ignition driver, you can use a 3rd party OPC server to connect to your device and then have Ignition connect to the server as a client. If the OPC server talks OPC-UA, you can add a new OPC-UA server connection in the Ignition Gateway. Configuration will be different depending on what OPC server you are using, but the following is an example of a popular solution, connecting to KEPServer via OPC-UA, see [Connecting to Kepware OPC-UA](#).

Adding Connection to 3rd Party OPC-Server via OPC-COM

The following section provides a detailed walk-through on how to connect to an OPC server using the OPC-COM module. If Ignition doesn't have a driver for your device and you don't have an OPC server that talks OPC-UA, you have to connect using the OPC-COM module, see [Connecting to OPC Classic \(COM\)](#).

Adding Tags for Allen Bradley Devices

Tags are how Ignition represents your PLC Tags. You create Tags in the Ignition Designer and then you can use these Tags to store history or display PLC data in your projects. For the most part, Allen Bradley devices support browsing of Tags in the PLC. There are a few exceptions, like the MicroLogix 1200/1500 for which you have to manually address your Tags. For now, we will focus on creating Tags from devices that support browsing.

1. Open the Ignition Designer.
2. Drag desired Tags from the OPC Browser to the Tag Browser as described here: [Creating Tags](#) (if you don't know what **Tag Provider** means don't worry, merely drag them into the **Tags** folder). You should now see some Tags in the Tag Browser that show the current values of the respective Tags in your PLC. Don't stop here. You should read through the related links below so you can learn more about Tags and how they work.

Why am I getting Endpoint errors?

If you are getting an "UaException: status=Bad_TcpEndpointUrlInvalid" error, it is because an OPC UA connection Endpoint can only contain certain characters. Most likely, you have an underscore in the Endpoint Address. See the [oracle docs](#) for more information on what exactly can be used in the URI.

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Allen Bradley Ethernet

Connecting to AB 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 and prior
- **CompactLogix** firmware v20 and prior
- **PLC-5**
- **MicroLogix** 1100, 1200, 1400, 1500
- **SLC 5/05**

CAUTION

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.

DANGER

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

Verifying Device Connectivity

Device connectivity can be verified in the following places:

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

- In the **Status** page of the Gateway under **Connections > OPC Connections**

Allen Bradley Connection Paths Explained

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

Here is a basic outline for figuring out your connection path, but 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.

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 is slightly different depending on which module type you are using. You can only move in two directions once you are "in" a module: out to the back plane, 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.

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Connecting to ControlLogix v21



INDUCTIVE
UNIVERSITY

Connecting to ControlLogix
v21

[Watch the video](#)

Connect to a Device

1. Go to the **Configure** section of the Gateway webpage.
2. Scroll down and select **OPC-UA** then **Devices**.
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 **Allen-Bradley Logix Driver**, and click **Next**.
5. On the **New Device** page, leave all the default values and type in the following fields:
 - Name: **CLX**
 - Hostname: type the **IP address** for the PLC
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.
7. To see all the tags, navigate to the **Configure** section of the gateway, and click on **Quick Client** under **OPC Connections**. On the **OPC Quick Client** page, expand the folder matching the name

used in step 5, and expand the **Global** folder to see available tags.

New Device

General	
Name	CLX ⓘ
Description	
Enabled	☒ (default: true)

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

Device Connection Settings

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

General Settings

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

Connectivity Settings

Name	Description
Hostname	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.
Port	New in 7.9.14 The port number for the Logic driver device connection.
Max Concurrent Requests	The number of requests that Ignition will try to send to the device at the same time. Increasing this number can sometimes help with your request throughput, however increasing this too much can overwhelm the device and hurt your communications with the device.
Slot Number	The Slot Number value is the zero based ControlLogix chassis slot number of the ControlLogix processor to connect to.
Connection Path	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 .

Advanced Settings

Name	Description
Automatic Rebrowse	Monitor for tag additions and UDT changes and automatically initiate a re-browse when detected. If this is disabled tags will only be browsed when connecting and reconnecting. (default: true)
CIP Connection Size	The CIP connection size to use during Forward Open requests. (default: 500)

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Connecting to ControlLogix



INDUCTIVE
UNIVERSITY

Connecting to ControlLogix

[Watch the video](#)

CAUTION

This driver is intended for devices using ControlLogix firmware version 20 and older.

Connect to a Device

1. Go to the **Configure** section of the Gateway webpage.
2. Scroll down under **OPC-UA Server**, and click **Devices**.
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 **Legacy Allen-Bradley ControlLogix**, and click **Next**.
5. On the New Device page, leave all the default values and type in the following fields:
 - Name: **CLX**
 - Hostname: type the **IP address** for the PLC
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.
7. To see all the tags, navigate to the **Configure** section of the gateway, and click on **Quick Client** under **OPC Connections**. On the **OPC Quick Client** page, expand the folder matching the name used in step 5, and expand the **Global** folder to see available tags.

New Device

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)

Device Connection Settings

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

General Settings

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

Connectivity Settings

Name	Description
Hostname	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 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 your communications with the device.

Supported Connection Methods

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

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Connecting to CompactLogix



INDUCTIVE
UNIVERSITY

Connecting to CompactLogix

[Watch the video](#)

CAUTION

This driver is intended for devices using CompactLogix firmware version 20 and older.

Connect to a Device

1. Go to the **Configure** section of the Gateway webpage.
2. Scroll down and select **OPC-UA Server > Devices**.
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 **Legacy Allen-Bradley CompactLogix**, and click **Next**.
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
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.
7. To see all the tags, navigate to the **Configure** section of the gateway, and click on **Quick Client** under **OPC Connections**. On the **OPC Quick Client** page, expand the folder matching the name used in step 5, and expand the **Global** folder to see available tags.

New Device

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)

Device Connection Settings

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

General Settings

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

Connectivity Settings

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

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Connecting to MicroLogix



INDUCTIVE
UNIVERSITY

Connecting to MicroLogix

[Watch the video](#)

IMPORTANT

MicroLogix 1200 series and 1500 series 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.

Connect to a Device

1. Go to the **Configure** section of the Gateway webpage.
2. Scroll down and select **OPC-UA Server > Devices**.
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 **Allen-Bradley MicroLogix**, and click **Next**.
5. On the **New Device** page, leave all the default values and type in the following fields:
 - Name: **MLX**
 - Hostname: type the **IP address** for the PLC
6. Click **Create New Device**. The **Devices** page is displayed showing the **MicroLogix** device is added to Ignition. The **Status** will show as Disconnected and then Connected.
7. To see all the tags, navigate to the **Configure** section of the gateway, and click on **Quick Client** under **OPC Connections**. On the **OPC Quick Client** page, expand the folder matching the name used in step 5, and expand the **Global** folder to see available tags.

New Device

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	

Device Connection Settings

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

General Settings

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

Connectivity Settings

Name	Description
Hostname	The Hostname value is the IP Address of the MicroLogix 1100 processor, MicroLogix 1400 processor or 1761-NET-ENI Ethernet interface. EthernetIP protocol on TCP port 44818 (0xAF12) is used to communicate to the listed devices.
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.
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 .

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

INFO

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.

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Connecting to PLC-5



INDUCTIVE
UNIVERSITY

Connecting to PLC-5

[Watch the video](#)

! INFO

ASCII data types from a PLC-5 are not supported by the Allen-Bradley PLC-5 driver

Connect to a Device

1. Go to the **Configure** section of the Gateway webpage.
2. Scroll down and select **OPC-UA Server > Devices**.
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 **Allen-Bradley PLC-5**, and click **Next**.
5. On the **New Device** page, leave all the default values and type in the following fields:
 - Name: **PLC-5**
 - Hostname: type the **IP address** for the PLC
6. Click **Create New Device**. The **Devices** page is displayed showing the **PLC-5** device is added to Ignition. The **Status** will show as Disconnected and then Connected.
7. To see all the tags, navigate to the **Configure** section of the gateway, and click on **Quick Client** under **OPC Connections**. On the **OPC Quick Client** page, expand the folder matching the name used in step 5, and expand the **Global** folder to see available tags.

New Device

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	

Device Connection Settings

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

General Settings

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

Connectivity Settings

Name	Description
Hostname	The Hostname value is the IP Address of the PLC-5 processor. The protocol that the PLC-5 processor supports is automatically detected. It will use either CSP protocol on port 2222 (0x8AE) or EthernetIP protocol on port 44818 (0xAF12).
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. For a more in depth explanation of connection paths, see Allen Bradley Connection Paths Explained section .

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.

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Connecting to SLC



INDUCTIVE
UNIVERSITY

Connecting to SLC

[Watch the video](#)

Connect to a Device

1. Go to the **Configure** section of the Gateway webpage.
2. Scroll down and select **OPC-UA Server > Devices**.
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 **Allen-Bradley SLC**, and click **Next**.
5. On the **New Device** page, leave all the default values and type in the following fields:
 - Name: **SLC**
 - Hostname: type the **IP address** for the PLC
6. Click **Create New Device**. The **Devices** page is displayed showing the **SLC** device is added to Ignition. The **Status** will show as Disconnected and then Connected.
7. To see all the tags, navigate to the **Configure** section of the gateway, and click on **Quick Client** under **OPC Connections**. On the **OPC Quick Client** page, expand the folder matching the name used in step 5, and expand the **Global** folder to see available tags.

New Device

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	

☒ Show advanced properties

Advanced	
Browse Cache Enabled	☒ (default: true)
Disable Processor Browse	☐ (default: false)
Zero TNS Connection	☐ (default: false)

Device Connection Settings

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

General Settings

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

Connectivity Settings

Name	Description
Hostname	The Hostname value is the IP Address of the 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. For a more in depth explanation of connection paths, see Allen Bradley Connection Paths Explained section .

Advanced Settings

Name	Description
Browse Cache Enabled	New in 7.9.13 Determines if browse results for the device should be cached. Default is enabled.
Disabled Processor Browse	If true, browsing the device is disabled.
Zero TNS Connection	If true, use a Transparent Network Substrate (request key) of zero for connection requests.

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

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Allen Bradley Connection Paths

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

Follow the Path

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

Setting Up the Device Connection

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

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

- **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.

CAUTION

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

Connecting to a PLC Through ControlNet



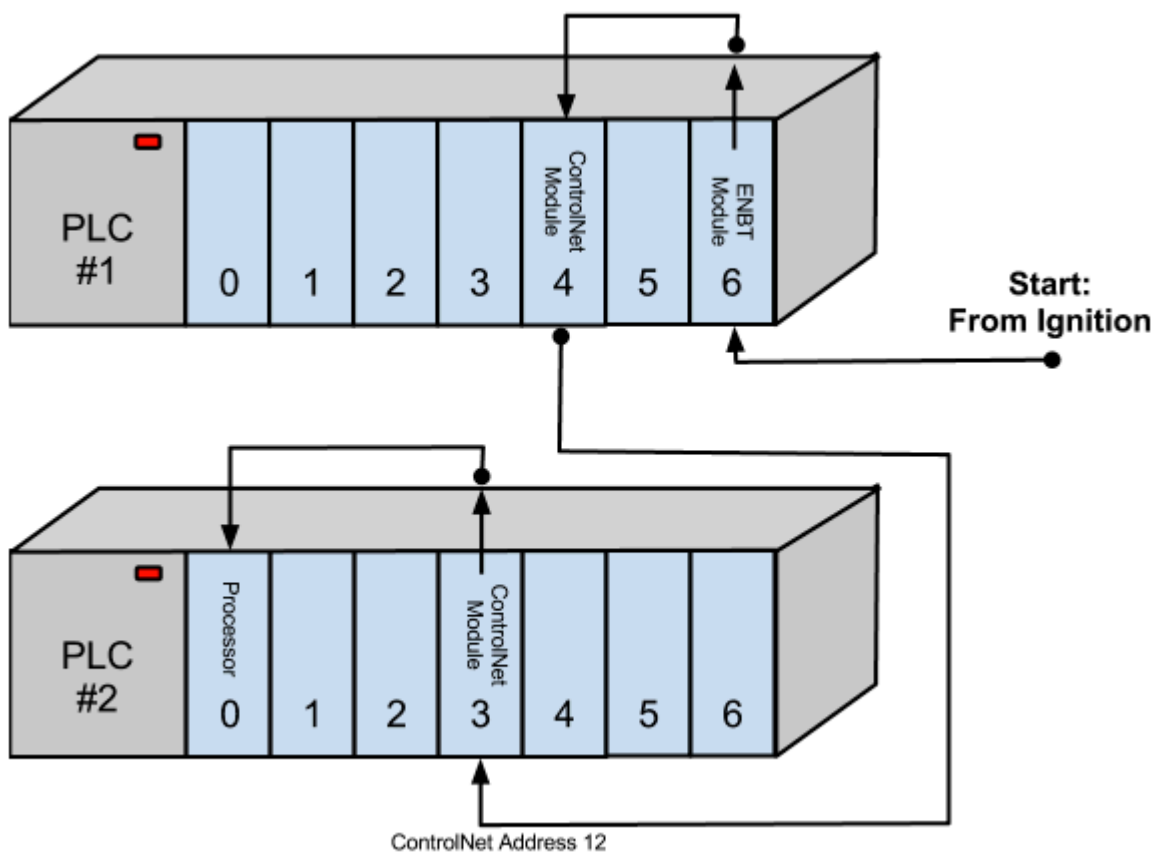
INDUCTIVE
UNIVERSITY

ControlNet Example

[Watch the video](#)

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

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



Connecting to a PLC Using ENBT



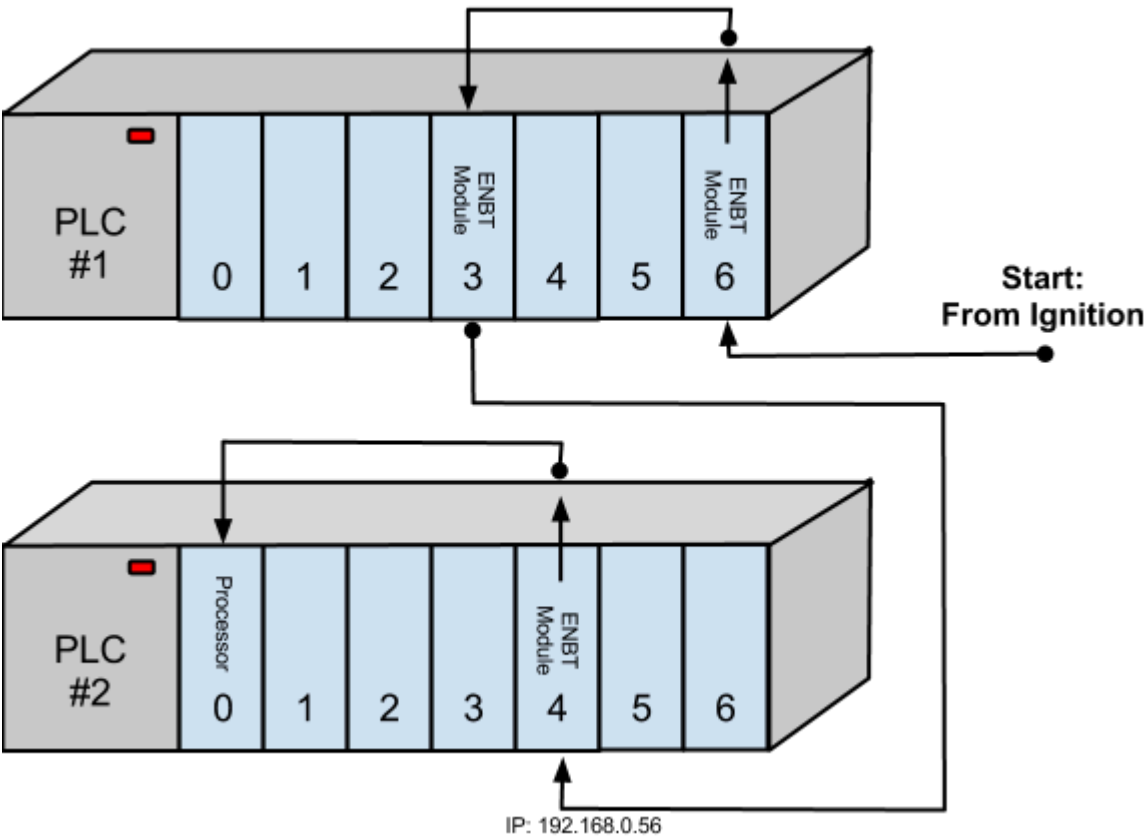
INDUCTIVE
UNIVERSITY

An ENBT Example

[Watch the video](#)

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

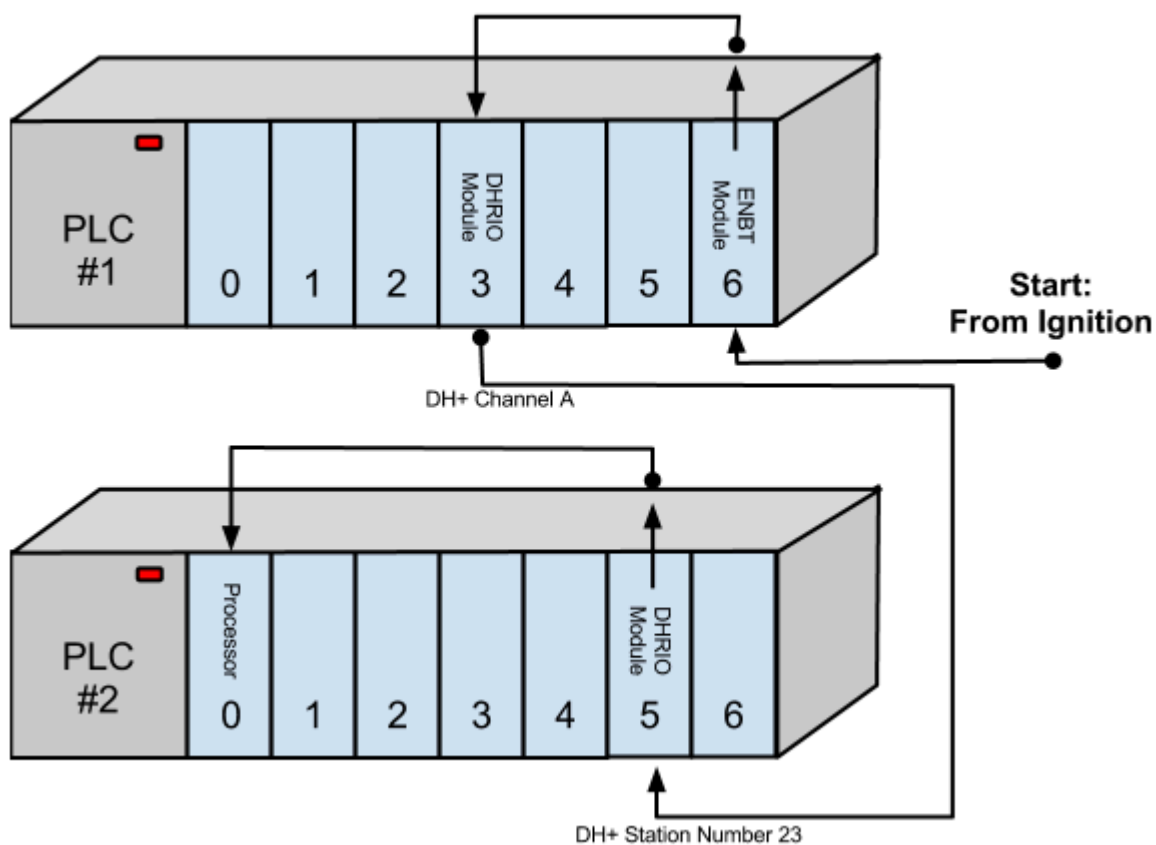
Number/Entry	Description
1	Move out of ENBT Module to the backplane
3	Move to ENBT Module in Slot 3
2	Move out the Ethernet Port
192.168.0.56	Move in EBNT Module at IP Address 192.168.0.56
1	Move out of EBNT to the backplane
0	Move to the Processor in Slot 0



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

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

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



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



INDUCTIVE
UNIVERSITY

A DH+ Example

[Watch the video](#)

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

The connection paths are:

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

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

SLC X

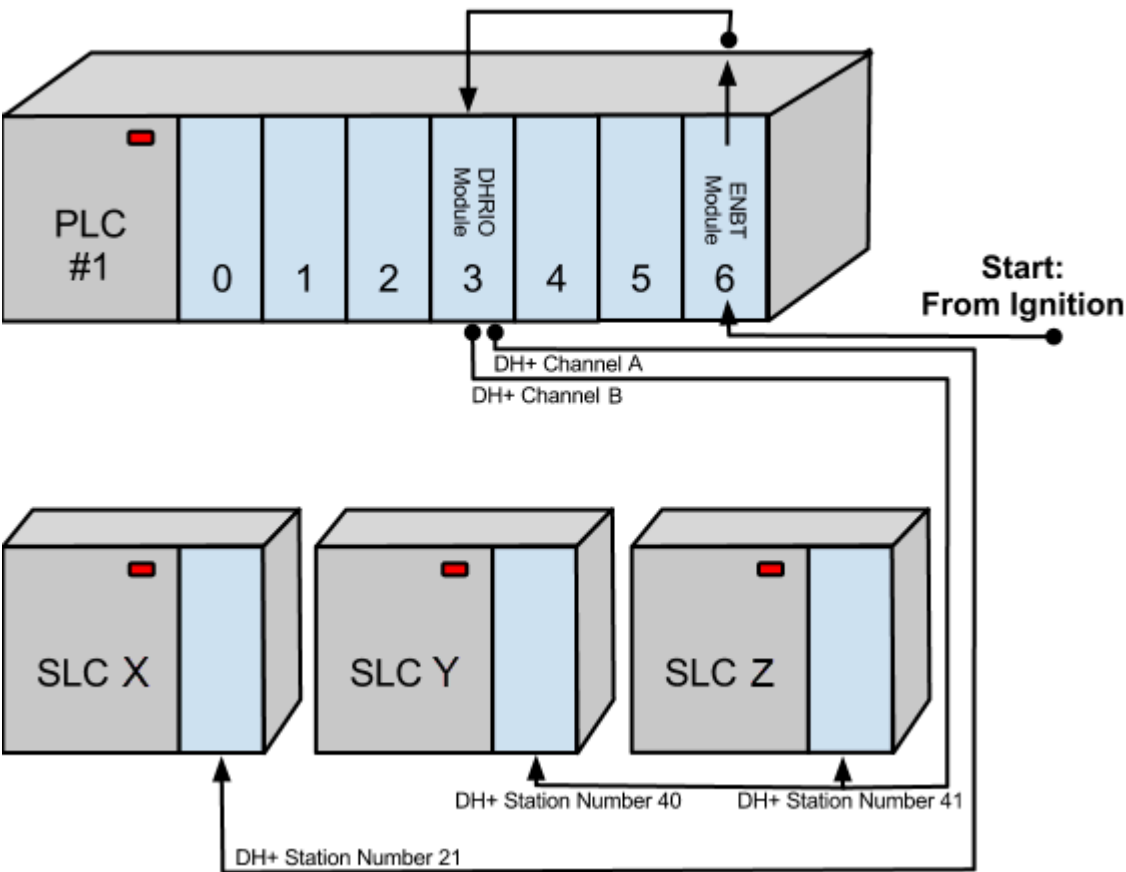
Number/Entry	Description
1	Move out of ENBT Module to the backplane
3	Move to DHRIO Module in Slot 3
2	Move out DH+ Channel A
21	Move in DH+ Station Number 23

SLC Y

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

SLC Z

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



This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

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. The great part is that you can 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 is often the more time consuming part. Luckily, it isn't that bad. Even with poor (or missing) device documentation, it's just a little bit of trial and error to get your addressing set.

Modbus Addresses

Getting access to your Modbus tags can be confusing because of all the different options available in the device connection. For example, you could have 0/1 based addressing or reversed word order

that isn't clearly documented in the device instructions. It's helpful to manually create a few Tags in Ignition and change your connection settings until your values are correct. One important setting to note is the **Max Holding Registers Per Request**. This setting can cause all of your test tags to go to bad quality when only one is bad. Change this setting (under the Advanced properties) to 1 while you are testing, and set it back when you are done. If you leave it at 1, this will cause a huge strain on your system after you have added hundreds or thousands of tags from your device.

Manually Addressing

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

Address Mapping

Creating an **Address Map** in Ignition for your device allows you to drag and drop tags just like any of the browseable device connections (like an AB 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.

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Connecting to Modbus Device



INDUCTIVE
UNIVERSITY

Connecting to Modbus Device

[Watch the video](#)

To connect Ignition to a Modbus device

1. Go to the **Configure** section of the Gateway webpage.
2. Scroll down and select **OPC-UA Server > Devices**.
3. On the **Devices** page, find the orange arrow and click on **Create new Device**.
4. On the **Add Device Step 1: Choose Type** page, select **Modbus TCP**, and click **Next**.

There are two modules 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.

Typically, you connect using the Modbus TCP driver as we have in step 4.

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

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 **Modbus** device is successfully created and added to Ignition. The **Status** will show as Disconnected and then Connected.

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

Driver Properties

Connectivity

Property	Description
Hostname	Is the IP Address of the Modbus device.
Port	Is the port to use when connecting to a Modbus device. The Modbus TCP port specified in the Modbus specification is 502, but it can be changed to a different port.
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.

Advanced

Property	Description
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 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	When this option is checked, the address range for each area starts at 1. If unchecked, the range starts at 0. 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. This will cause 1024 to be sent to the device to read Modbus address 1024.
Span Gaps	When this option is checked, it spans address gaps when optimizing requests, reducing the number of requests but increasing the amount of

Property	Description
	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.

StringHandling

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

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Modbus Addressing



INDUCTIVE
UNIVERSITY

About Modbus Addressing

[Watch the video](#)

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 **OPC 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.

To manually specify each address

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

1. In the **Tag Browser**, right-click on the **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: Temp

Data Type: Int4

OPC Server: choose **Ignition OPC-UA Server** from the dropdown

OPC Item Path: **[Modbus]HR1**, the **Modbus** device name goes in the square brackets then 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.

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:

i NOTE

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

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

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

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

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

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

Float/Double

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

Integer

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

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

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

 NOTE

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:

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

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, and 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.

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Modbus Address Mapping



INDUCTIVE
UNIVERSITY

**About Modbus Address
Mapping**

[Watch the video](#)

Because it can be very tedious manually entering 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.

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

Simple Mapping Demonstration

Temperature readings are being stored in 10 16-bit addresses: 40,010 - 40,019. There is a single Modbus device at the IP address (unit ID 0), and the addresses are decimal. The mapped addresses should appear in the 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

 INFO

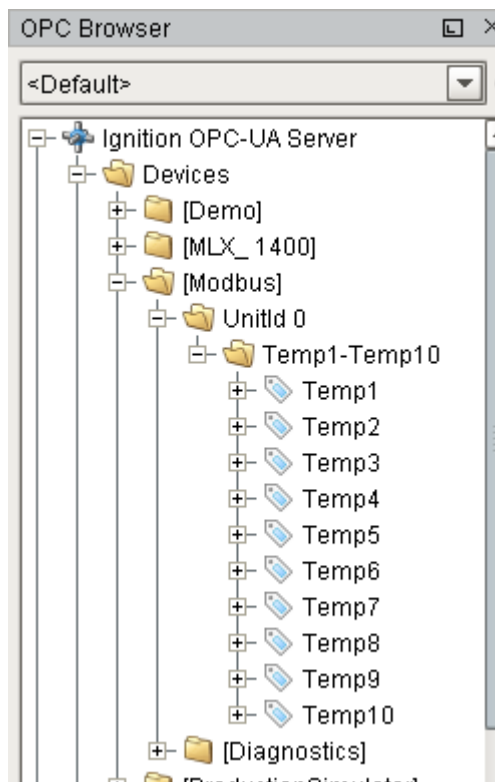
Note that 40,010 (HR10) uses Modbus Address 10 as a starting point and not 40,010. The leading 4 is automatically entered for you when you select Holding Register from the dropdown. The same is true for all other types of tags: Inputs are 30,000, Discrete are 10,000, etc.

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:



To specify the address mapping

1. Go to the **Configure** section of the Gateway webpage and under **OPC-UA Server > Devices**, click on the **More** button and select **Addresses** to the right of your Modbus device.

License Incomplete 0:00:00 [Reset Trial](#) [View Modules](#)

Devices

Name	Type	Description	Enabled	Status		
CLX	Legacy Allen-Bradley ControlLogix		true	Connected: Protocol: EIP - Run Mode	More ▾	edit
Dairy	Simulators Dairy Demo Simulator		true	Connected	delete	edit
Generic	Simulators Generic Simulator		true	Connected	delete	edit
MLX	Allen-Bradley MicroLogix		false	Disabled	More ▾	edit
Modbus	Modbus TCP		true	Connecting	Addresses	More ▾ edit
SLC	Simulators SLC Simulator		true	Connected	delete	delete edit
Tanks	Production Simulator		true	Connected	delete	edit

[→ Create new Device...](#)

2. Click on **Add Row**.

Choose File No file chosen

[Import Configuration](#)

[Export Configuration](#)

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

There are currently no address mappings configured...

[Add Row](#)

[Save](#)

3. 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)

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

Radix

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

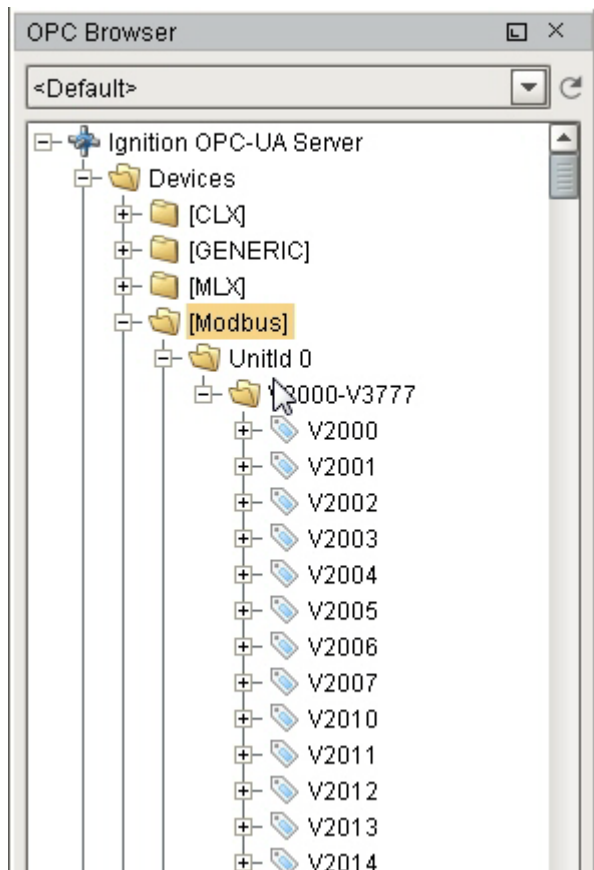
NOTE

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

4. Click **Save**.

5. Go to the **OPC Browser** in the Designer or the **OPC Connections > Quick Client** (on Gateway) and open the **Modbus** folder.

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



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

Prefix	Start	End	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

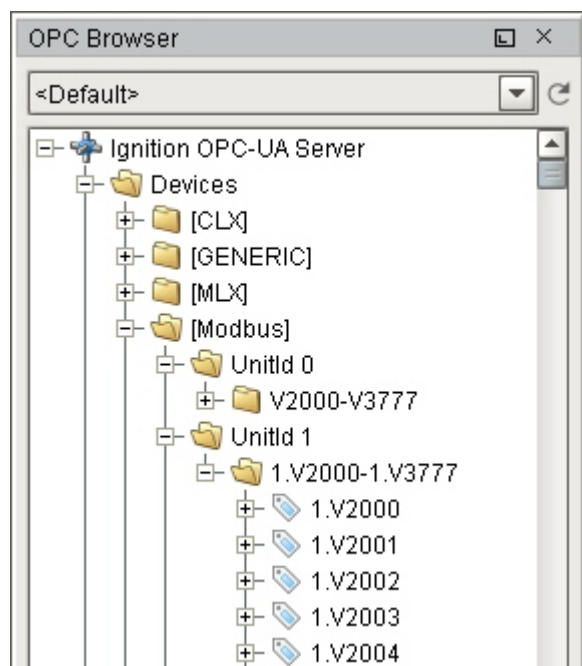
It is not recommended to communicate to multiple Modbus devices through a Modbus Gateway where Gateway has the same address. Therefore, do not add multiple Modbus devices with the same IP address.

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

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.



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.

To map float point addresses

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

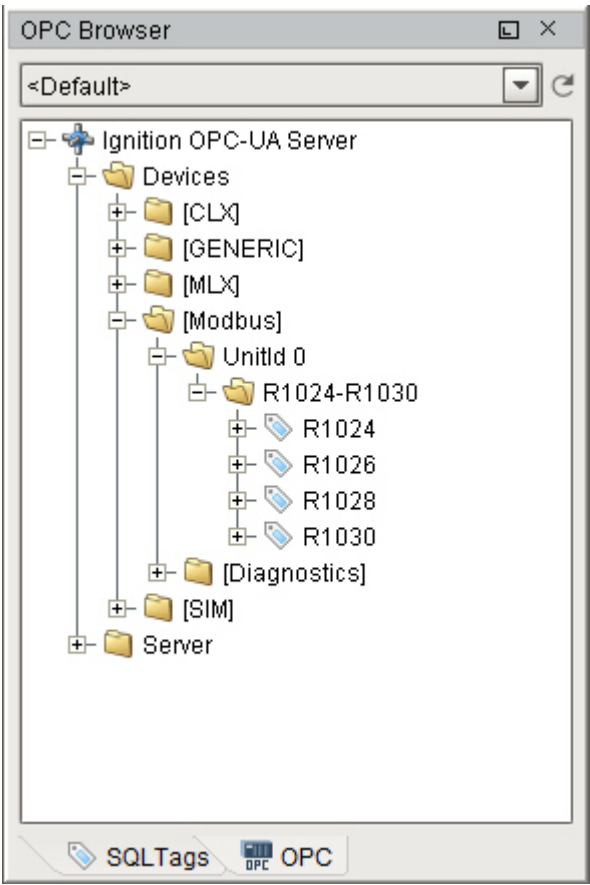
Because **Modbus Type of Holding Register (Float)** is selected, the driver will read two consecutive 16-bit words and convert it to a floating point value. It also indexes the Modbus Address by 2 for each

entry. In this case, R1024 reads from Modbus addresses 1024 and 1025 and converts them into a floating point value. When writing, the reverse of converting a floating point value into two 16-bits words is done before sending them to the device.

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

Radix 10

The following 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.

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Siemens

Siemens Devices

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

Connection to a Device



INDUCTIVE
UNIVERSITY

Connecting to S7 Devices

[Watch the video](#)

1. Go to the **Configure** section of the Gateway webpage.

2. Scroll down under **OPC-UA Server** and click **Devices**.
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 the type of device. After making a selection, click **Next**. There are four types 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.
5. On the **New Device** page, leave all the default values and type in the following fields:
 - Name: **S7Device**
 - Hostname: type the **IP address** for the PLC
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 **Siemens** device is successfully created and added to Ignition. The **Status** will show as Disconnected and then Connected.

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

Configuring Siemens Addressing



INDUCTIVE
UNIVERSITY

About Siemens Addressing

[Watch the video](#)

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

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** from the dropdown
- OPC Item Path: `[S71200]IW0`, the `S71200` device name goes in the square brackets then you give the address to PLC which in this example is **IW0** (Word at Offset 0 in the Inputs area). The **Address Syntax** section below, explains how you can construct these addresses.

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

Address Syntax

You need a device name plus a tag address to create a tag. The device name is a known, but the tag address needs to be configured. Once you have both the device name and tag address you enter them in the **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**.

Areas

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

Data Types

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

Examples

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

Address	Description
<code>IB0</code>	Byte at Offset 0 in the Inputs area
<code>IW0</code>	Word at Offset 0 in the Inputs area
<code>DB500,DI8</code>	DInt at Offset 8 in DataBlock 500
<code>ISTRING24.50</code>	A String of length 50 starting at offset 24 in the Inputs area
<code>IX20.3</code>	Bit 3 of the Byte at Offset 20 in the Inputs area
<code>T0</code>	Timer at offset 0 (No DataType is specified for Timers)
<code>C0</code>	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 2 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 Settings

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

Connectivity Settings

Setting	Description
Hostname	The hostname or IP address of the device.
Timeout	The request timeout, specified in milliseconds. The default is 2,000.

Advanced Settings

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

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

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 to 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 1 variable node containing the entire message received as well as an additional variable node per field configured. A device configured with a field count of four would have five nodes total: one for the message and four for the fields.

Connecting to a Device



INDUCTIVE
UNIVERSITY

Connection to TCP Device

[Watch the video](#)

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.

To 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 **Configure** section of the Gateway webpage.
2. Scroll down and select **OPC-UA Server > Devices**.
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, 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** page 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 Properties

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

Connectivity Properties

Property	Description
Port(s)	On the UDP driver, this is the port(s) to listen on. On the TCP driver, this is the port(s) to connect to. Separate multiple ports with a comma.
Address	On the UDP driver, this is the IP address to listen to. On the TCP driver, this is the IP address to connect to.
Inactivity Timeout	The number of milliseconds without receiving data from the source before a disconnect/reconnect is made. Set to 0 to disable.

Message Properties

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

Advanced

Both drivers have unique Advanced Properties

TCP Driver

Property	Description
Writeback Enabled	Enable writeback capabilities for the device.
Writeback Message Delimiter	The delimiter expected by the device signaling the end of the incoming message.

UDP Driver

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

TAG POLLING SPEED

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

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Omron

Connecting Ignition to an Omron Device

1. Go to the **Configure** section of the **Gateway webpage**.
2. Scroll down and select **OPC-UA Server > Devices**.
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 the device
 - 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 Settings

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

Main Settings

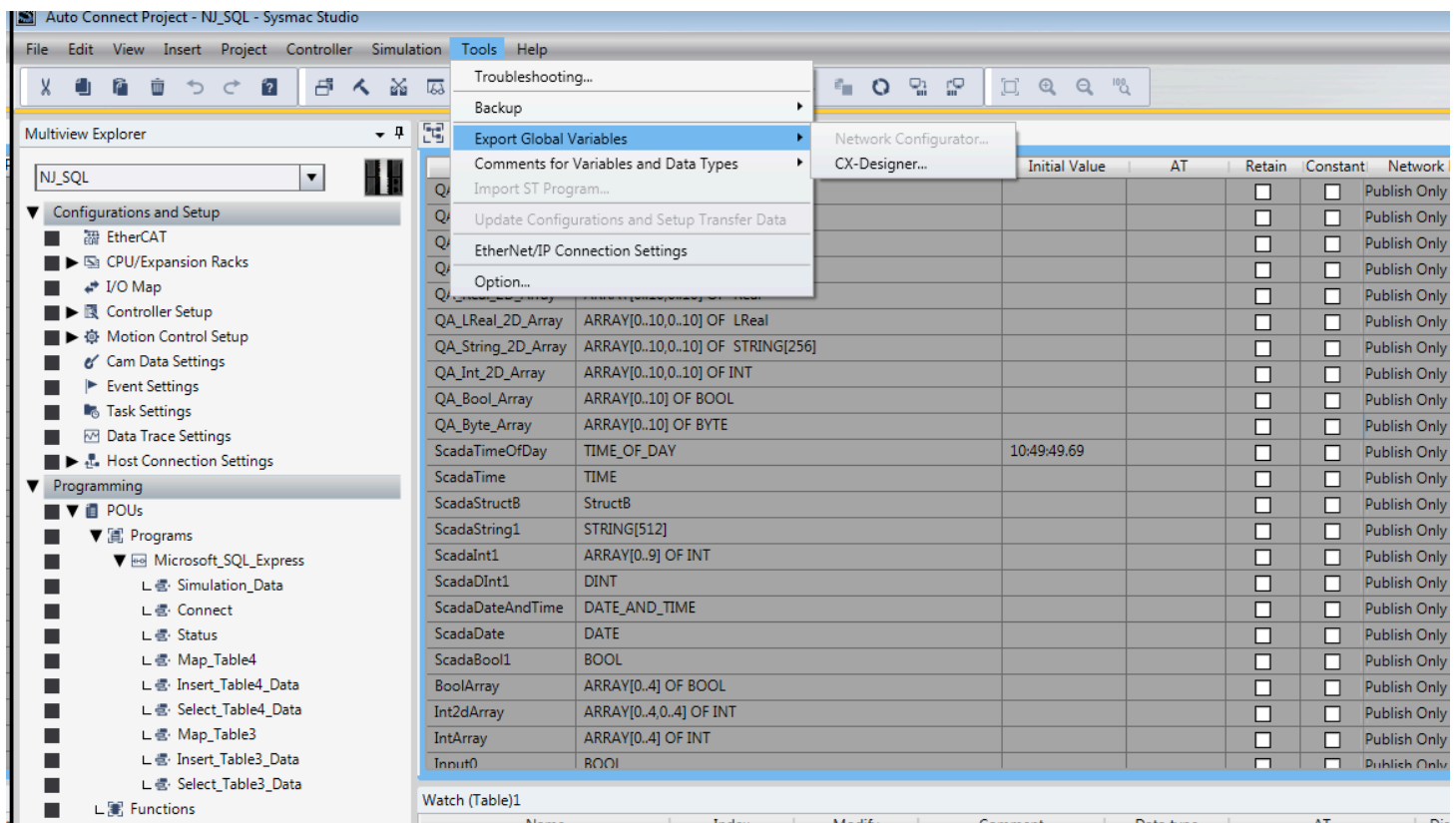
Name	Description
Hostname	The hostname or IP address of the device.
Timeout	The request timeout, specified in milliseconds. The default is 2,000.

Advanced Settings

Name	Description
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

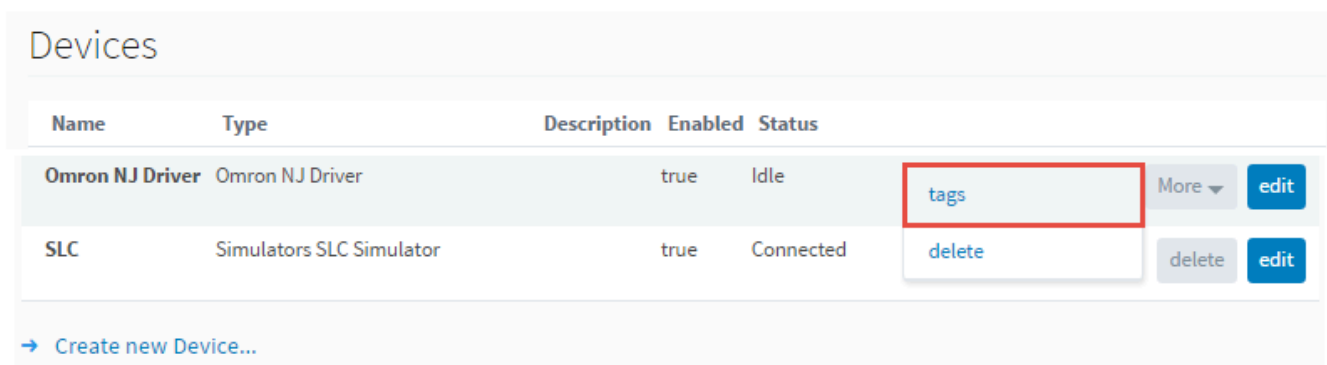
To export variables from Sysmac Studio, navigate to the global variables and select **Tools > Export Global Variables > CX-Designer**.



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.



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 RowDelete Row(s)

Save ChangesCancel

Manage Tags

Import a List of Tags.

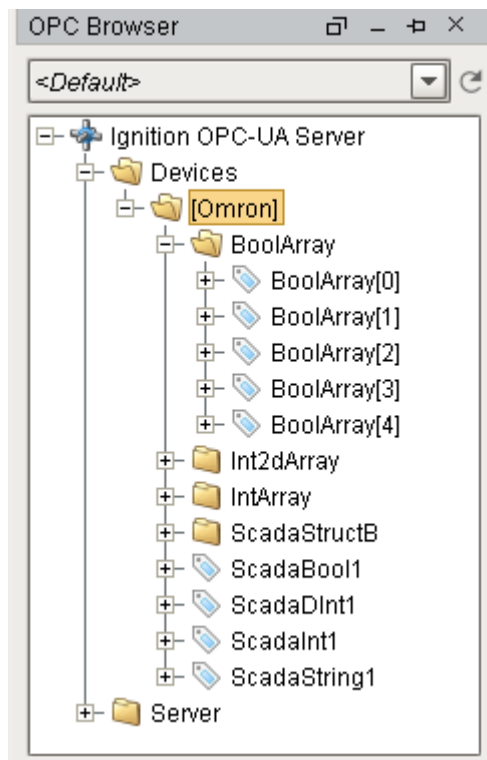
No file chosen

Tags

☐ 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 >>>

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.

Scalars

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

☐ ScadaBool1	BOOL		RW
-------------------------------------	------	--	----

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

! INFO

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
--------------------------	--	-------------------	----------------------------------	-----------------------------------	---------------

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Simulators

There are 3 simulator drivers in Ignition, and all can be used to read and write tags without any network or PLC connection.

Generic Simulator

The generic simulator 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.

There are no configurable settings for the generic simulator.

Simulator tags

Read Only

Static values that do not change. Writing is disabled on these tags.

Tag Name	Value
ReadOnlyBoolean1	False
ReadOnlyBoolean2	True
ReadOnlyShort1	1
ReadOnlyShort2	2
ReadOnlyInteger1	1
ReadOnlyInteger2	2
ReadOnlyInteger3	3
ReadOnlyInteger4	4
ReadOnlyInteger5	5
ReadOnlyLong1	1
ReadOnlyLong2	2
ReadOnlyFloat1	1.1
ReadOnlyFloat2	1.2
ReadOnlyDouble1	1.1
ReadOnlyDouble2	1.2
ReadOnlyString1	ABCDEFGH
ReadOnlyString2	"ZYXWVUT"

Writable

Tags with a persistent value that can be written to.

Tag Name	Value
WriteableBoolean1	False
WriteableBoolean2	False
WriteableShort1	0
WriteableShort2	0
WriteableInteger1	0
WriteableInteger2	0
WriteableLong1	0
WriteableLong2	0
WriteableFloat1	0
WriteableFloat2	0
WriteableDouble1	0
WriteableDouble2	0
WriteableString1	" (empty string)
WriteableString2	" (empty string)

Random

Tags with values that change randomly at a fixed rate.

Tag Name	Value
RandomBoolean1	10 sec
RandomBoolean2	10 sec
RandomShort1	5 sec
RandomShort2	5 sec
RandomInteger1	1 sec
RandomInteger2	1 sec
RandomLong1	2 sec
RandomLong2	2 sec
RandomDouble1	10 sec
RandomDouble2	10 sec

Sine

Tags that follow a sine wave over a set period. The value transitions from the low value, to the high value, and finally back to the low value over a single period.

Tag Name	Low Value	High Value	Period
Sine0	-100	100	60 seconds
Sine1	-10	10	10 seconds
Sine2	0	50	15 seconds
Sine3	-40	60	20 seconds
Sine4	-100	100	40 seconds
Sine5	-100	100	60 seconds
Sine6	-10	10	10 seconds
Sine7	.-20	30	15 seconds
Sine8	-40	60	20 seconds
Sine9	-85	115	40 seconds

Ramp

Tags that start at a low value, and increase to a high value over a set rate. Once the high value is reached, they reset back to the low value.

Tag Name	Low Value	High Value	Period
Ramp0	0	1000	75 seconds
Ramp1	0	100	10 seconds
Ramp2	-25	175	15 seconds
Ramp3	10	310	20 seconds
Ramp4	0	400	40 seconds
Ramp5	0	500	60 seconds
Ramp6	0	600	12.5 seconds
Ramp7	-5	695	17.5 seconds
Ramp8	10	810	30 seconds
Ramp9	-10	890	50 seconds

Realistic

Tags that change value by having a number added to them at a fixed rate. The number added is determined by randomly selecting between the low and high values on the tables listed below.

Tag Name	Low Value	High Value	Rate of Change
Realistic0	-50	50	5000 milliseconds
Realistic1	-50	50	500 milliseconds
Realistic2	-50	50	1000 milliseconds
Realistic3	-50	50	1500 milliseconds
Realistic4	-50	50	2000 milliseconds
Realistic5	-50	50	2500 milliseconds
Realistic6	-50	50	3000 milliseconds
Realistic7	-50	50	3500 milliseconds
Realistic8	-50	50	4000 milliseconds
Realistic9	-50	50	4500 milliseconds

Allen Bradley SLC Simulator

The SLC simulator driver creates a simple device whose address structure mimics a basic SLC structure. There are currently no configurable parameters.

Simulators Dairy Demo Simulator

A simulator for use with the legacy IA Dairy Demo project. It has a ControlLogix like structure with Compressor, Tank, Motor tags and more.

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Third Party OPC Servers

Other 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.

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Connecting to Kepware OPC-UA



INDUCTIVE
UNIVERSITY

Connecting to Kepware OPC-UA

[Watch the video](#)

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

To Connect to KEPServerEX from Ignition using OPC-UA

1. In the **Configure** section of the Gateway, go to **OPC Connections > Servers**. The OPC Server Connection page is displayed showing the OPC-UA servers your Ignition is connected to.
2. Find the blue arrow, and click on **Create new OPC Server Connection....** The **Add OPC Server Connection Step 1: Choose Type** page is displayed.
3. Choose **OPC-UA** as the connection type, and click **Next**.
4. On the **Discover OPC-UA Endpoints** page, type in the endpoint of the OPC-UA server Ignition should connect to. The format should be as follows:

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

#Examples

```
#If KepServer is installed on the same server as Ignition and uses the default port,
```

```
#the following endpoint could be used:
```

```
opc.tcp://localhost:49320
```

```
#If KepServer is installed on a different server from Ignition, then the IP address would be used:
```

```
opc.tcp://10.1.1.10:49320
```

Once an endpoint has been entered, click the **Discover** button. A list of available Security Policies and Message Security options will appear.

5. Select a Security Policy and Message Security configuration to use when connecting to the endpoint. Both **Basic256** and **SignAndEncrypt** offer a higher level of security.

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

6. On the **New OPC UA Connection Settings** page, give the connection a name, and click the **Create New OPC Server Connection** button. A Username and Password do not need to be given, and should be left blank.
7. 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.
8. 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.
9. On the **OPC UA Configuration Manager** window, go to the **Trusted Clients** tab, 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**.

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.

10. Go to the **OPC Connections > Quick Client** in the Configure section of the Gateway, under the **Kepware > Channel1 > Device1** folder, you will see the **Tag** folders.

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 OPCUA 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 Connections and 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, take a look at KepServer's 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.

To Download a Client Certificate Manually

1. Go to **Configure** section in the Gateway.
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.

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

OPC COM

Connecting to OPC Classic (COM)

CAUTION

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.

To 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.

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.

To Install OPC-DA server connection

1. Go to the Ignition Gateway **Configure** 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 your server connection up, give our Inductive Automation tech support line a call and one of our representatives will be happy to assist you.

Troubleshooting OPC-COM Connections

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

Common Problems

OPC server is not listed in Choose OPC-DA Server list when first creating a connection

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

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

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

Connection status is Connected but data quality is bad or the connection goes Faulted after trying to read tag data

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

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

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

Ignition launches second instance of an already running OPC server and is unable to see any data

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

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

To 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 Ignition gateway configuration section, as outlined above.
2. Under Tags, History, select "Create new Historical Tag Provider..."
3. Select "OPC-HDA Provider"
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.

Property	Description
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.

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Replacing the Default OPC-UA Certificate

Root of Trust Secure Connection

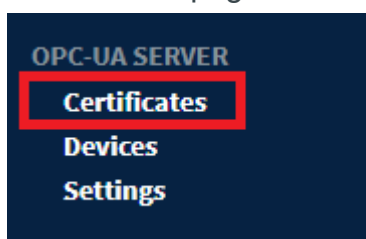
Some OPC-UA servers or PLCs (specifically, [Bedrock Automation](#)) may support the use of signed security certificates. This prevents unauthorized access to the data hosted on the UA server, and in some cases allows the UA client to connect without using a username or password. You can replace the default security certificate that is used by Ignition when connecting as an OPC-UA client, allowing Ignition to connect to these UA servers.

This process involved is very similar to applying an SSL Certificate to Ignition's web server: download a file (a Certificate Signing Request) from the Ignition Gateway, submit the file to an entity that will sign it, and then upload a signed certificate back into the Ignition Gateway. This extra layer makes connecting to the OPC Server very secure.

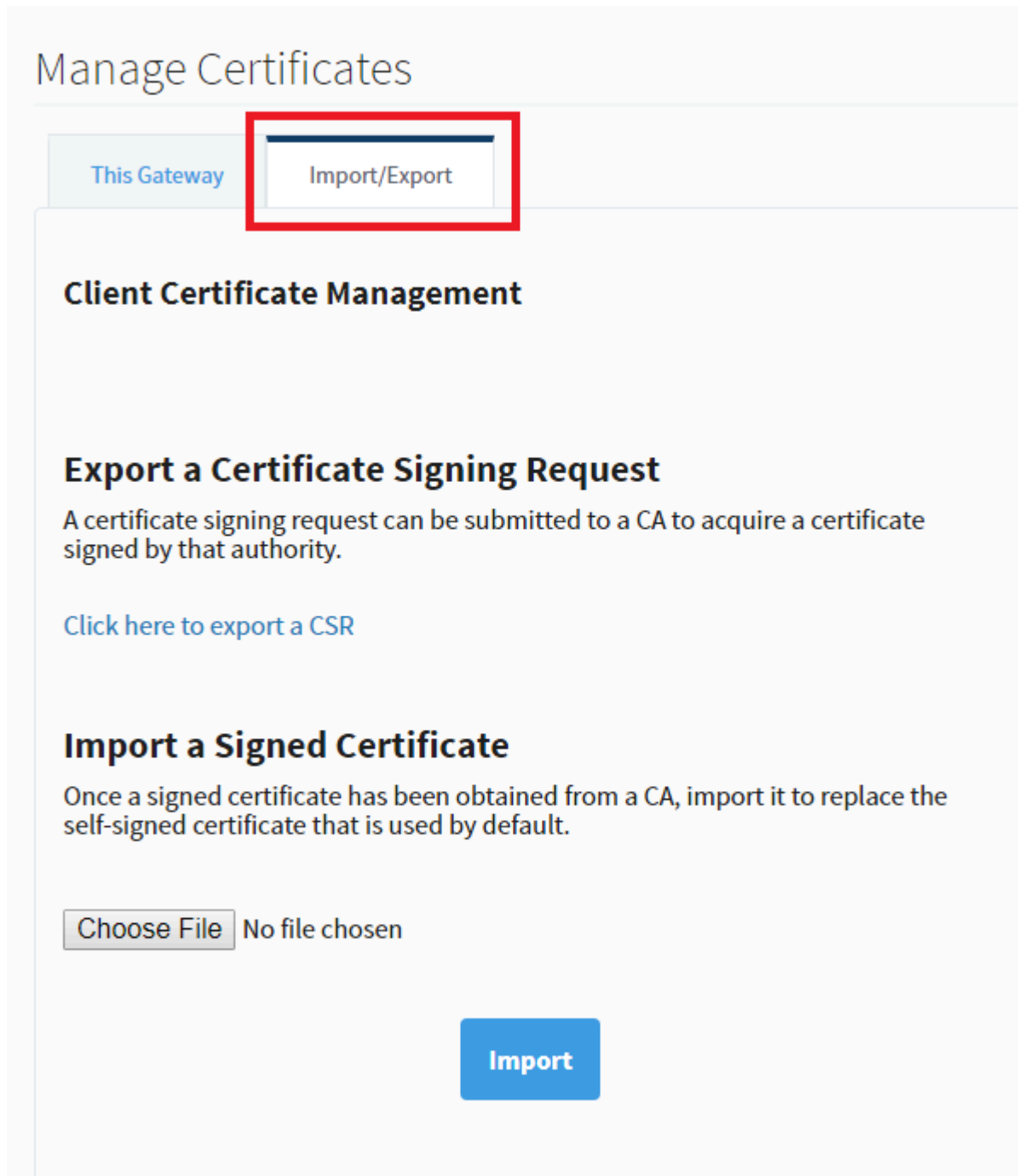
This process involved is very similar to applying [an SSL Certificate to Ignition's web server](#): download a file (a **Certificate Signing Request**) from the Ignition Gateway, submit the file to an entity that will sign it, and then upload a signed certificate back into the Ignition Gateway. This extra layer makes connecting to the OPC Server very secure.

How to Replace Ignition's Default UA Client Certificate

1. Go to the Configure section Gateway Webpage and click on the link for the **OPC-UA Server > Certificates** page.



2. Click on the **Import/Export** tab.



3. Click on the '**Click here to export a CSR**' link to generate the **Certificate Signing Request (CSR)** file.
4. Now that you have the **CSR**, you will need to send the file off to be signed. Who you send the file to differs depending on the UA server you are trying to have Ignition connect to. For example, if using the built-in OPC-UA server on a **Bedrock Automation** PLC, then you can submit the CSR to Bedrock. Always consult the UA server's documentation for recommendations on who to send the CSR to.

This step may take some time, as the entity you sent your CSR to will need to process the request, and generate a unique Signed Certificate.

5. Once processed, the entity that processed your CSR will respond with a Signed Certificate.
Again, on the **OPC-UA Server > Certificates** page, import the Signed Certificate by using the **Browse** button.
6. Click the **Import** button. After uploaded, you will see a confirmation at the top of the page stating that the file was uploaded successfully.
7. With the new certificate, you can now create an **OPC-UA connection** to the UA server.

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

DNP3



INDUCTIVE
UNIVERSITY

Connecting to DNP3 Devices

[Watch the video](#)

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 it is 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.

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 Configure section of the Gateway webpage.
2. Scroll down under **OPC-UA Server** and click **Devices**.
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**.
5. On the **New Device** page, leave all the default values and type in the following fields:
 - Name: **DNP3**
 - Hostname: type the IP address for the device.

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 Settings

Setting	Description
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 Settings

Setting	Description
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 milliseconds, 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. 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

Advanced Settings

Setting	Description
Message Fragment Size	The maximum size of a message fragment in the application layer, default is 249.
Message Timeout	The amount of time to wait for a message response from the outstation, default is 5,000.
Retries	The number of retries on a message timeout, default is 0.
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

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

A Note on Source 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.

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:** Universal 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.

Browsing DNP3 Points



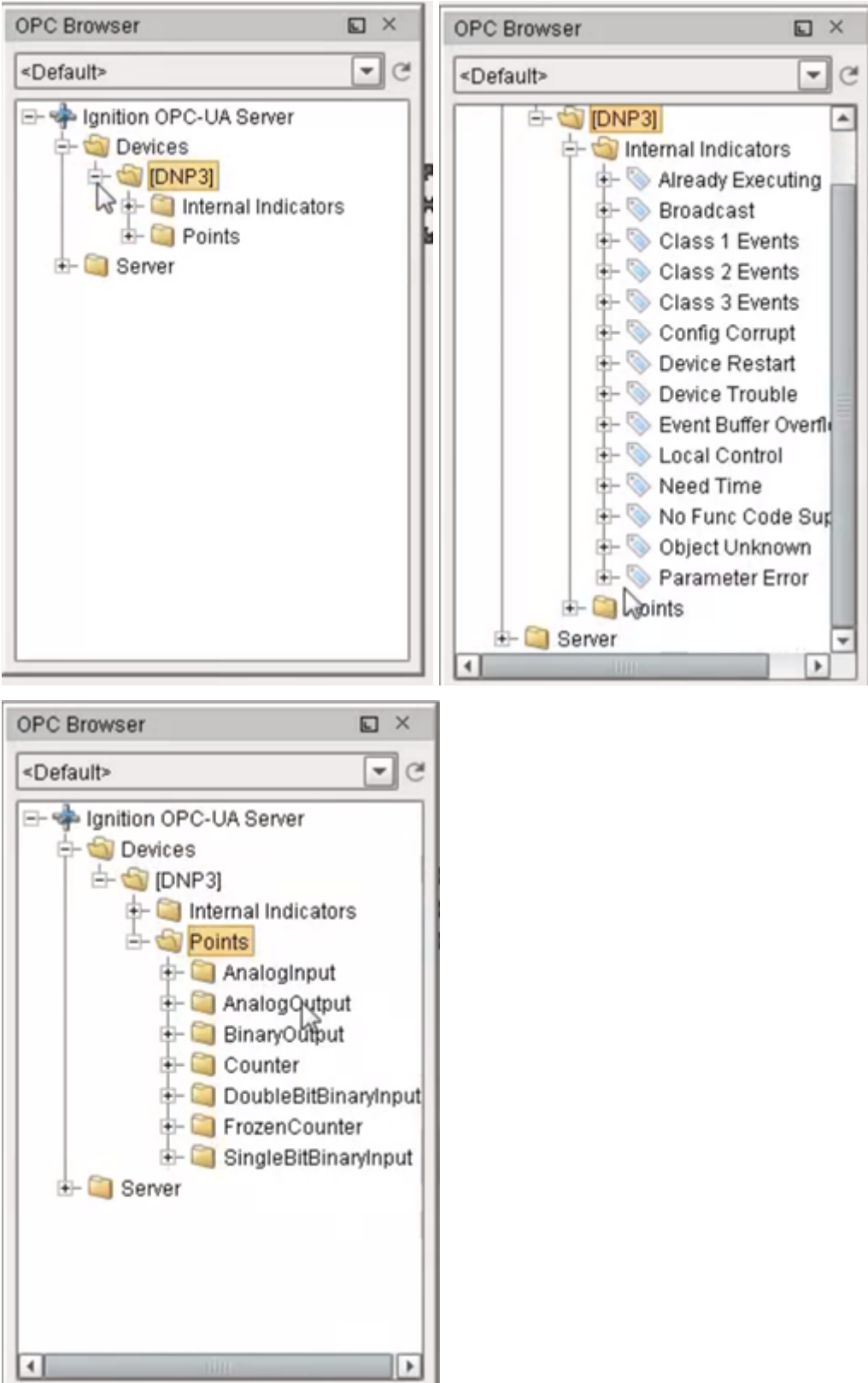
INDUCTIVE
UNIVERSITY

About DNP3 Addressing

[Watch the video](#)

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.



Point Types

Type Name	Group	Supported Variations
SingleBitBinaryInput	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

Type Name	Group	Supported Variations
		• 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

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 Attribute	Description
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.

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

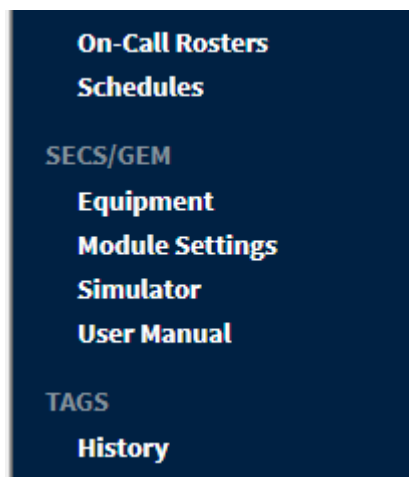
For documentation on the latest stable release, see the [latest version](#) (8.3).

SECS/GEM

SECS/GEM Overview

The SECS/GEM Module enables Ignition projects and third-party applications to communicate with semiconductor fab equipment. The module is an implementation of the GEM standard (SEMI E30-0307), the SECS-II standard (SEMI E5-0712) and the HSMS standard (SEMI E37-0308). The SECS/GEM Module is implemented as a host that can talk to one or more tools via equipment connections configured within the Gateway. Communication with equipment is in the form of messages defined by the SECS-II standard and referred to as SECS messages or just messages. By use of a simple definition file, any SECS message (including custom ones) can be supported.

The module will add a new header to the Configure section of the Gateway Webpage that allow you to configure connections to the equipment, or setup simulators that are used for testing.



Equipment Connections

The equipment connections are what allow Ignition to communicate with the specialized semiconductor fab equipment. Equipment connections are setup within the Configure section of the Gateway Webpage. Connections can be made over Ethernet using TCP/IP (HSMS protocol) or over a direct serial connection (SECS-I protocol) using a RS-232 cable. Note, that the direct serial connection requires that the equipment be connected directly to the Ignition Gateway server, and that the Serial Support Gateway Module be installed. The equipment connections also require the use of a

database, which will store sent and received messages for audit purposes, and to hold some configuration data. Each equipment connection connects to one tool and one database and transfers SECS messages between the Gateway to the tool. Any number of equipment connections can be configured in the Gateway as long as your license allows it, and different equipment connections can be configured to use the same database connection or different database connections.

ignition

by inductive automation

HOME

STATUS

CONFIGURE

Search...

SYSTEM

Overview

Backup/Restore

Licensing

Modules

Projects

Redundancy

Gateway Settings

NETWORKING

Gateway Network

Email Settings

SECURITY

Auditing

Users, Roles

Service Security

Security Zones

DATABASES

Connections

Drivers

Store and Forward

ALARMING

General

Journal

Notification

On-Call Rosters

Schedules

SECS/GEM

Equipment

Module Settings

Simulator

User Manual

Equipment Connections (5 used out of 5)

Equipment Name	Equipment Description	SECS Definition Language (SDL) file	Connection Type	Status		
EquipmentFour		messages.sdl	Ethernet/HSMS Protocol	Communicating	More	edit
EquipmentOne		messages.sdl	Ethernet/HSMS Protocol	Communicating	More	edit
EquipmentThree		messages.sdl	Ethernet/HSMS Protocol	Communicating	More	edit
EquipmentTwo		messages.sdl	Ethernet/HSMS Protocol	Communicating	More	edit
hitachi		messages.sdl	Ethernet/HSMS Protocol	Not Connected	More	edit

Create new Equipment Connection...

SECS Definition Language (SDL) File

The **SDL file** is where SECS messages are defined. Any SECS message defined in this file can be sent and received. All SECS messages that are sent and received are validated against this file. If a message is not defined or doesn't match a definition then it is not sent, and it is instead inserted into the Errors table as a validation error along with information about why it didn't validate.

Each Equipment Connection has one SDL file and a default SDL file will be used if an alternate is not specified. The SDL file definitions are based on the GEM and SECS standards.

Simulators

The SECS/GEM Module includes the capability to create and configure **equipment simulators**. The simulators are not a full implementation of the GEM standard. Equipment Connections can connect to

simulators and they can exchange various messages. A range of SECS messages are supported by the simulators. This capability exists for getting started with the SECS/GEM Module and for testing applications.

SECS/GEM Message Basics

Once a connection has been properly configured to a tool, messages can be sent between Ignition and the equipment. These messages are often complex, which is why SECS messages are often in the form of a JSON string, which can contain any number of lists and name/value pairs. The messages are typically sent and received using specialised scripting functions within Ignition, such as `system.secsgem.sendRequest` and `system.secsgem.getResponse`, that the SECS/GEM module adds. To send a message, a user calls the `system.secsgem.sendMessage` function in a Python script within an Ignition Gateway or client, and passes the SECS message as JSON text to the function. A transaction id is returned, which can then be passed to the `system.secsgem.getResponse` function. This function returns the response from the tool as JSON text. However, some SECS messages can be handled differently, such as those that are sent repeatedly from the equipment like status variables. Those messages can be configured to fire Message Handler scripts in the Gateway or Client when the message is received, which can handle the constant flow of messages in a consistent manner. Alternately, the messages can be configured to create Tags with their values, updating the Tags anytime a new message comes in.

Definitions

The SECS/GEM Module uses some unique terms that are not used in the rest of Ignition:

Term	Description
SECS/GEM	The SECS/GEM Module is named after two standards the SECS-II standard (SEMI E5-0712) and the GEM standard (SEMI E30-0307). The SECS-II standard defines messages that can be exchanged between a host and an equipment. SECS stands for Semi Equipment Communications Standard. The GEM standard defines ways for equipment to implement functionality using SECS messages. GEM is short for Generic Model for Communications and Control of Manufacturing Equipment.
HSMS Protocol	A TCP/IP based message transfer protocol for sending SECS messages. The SECS/GEM Module uses HSMS or SECS-I for sending messages.
Equipment	A semiconductor device that sends and receives SECS messages. This is also called a tool.
Equipment Connection	An Equipment Connection enables a database interface to an equipment. An Equipment Connection has a connection to an equipment and a connection to a database and transfers SECS messages between them. Ignition projects or third-party applications can insert SECS messages into a database table to send them to an equipment, and query database tables to retrieve SECS messages from an equipment.
Stream	A category of SECS messages.
Function	A specific SECS message in a stream.
Request Message	Also primary message. A SECS message having an odd numbered function that originates a communication and may require a response message.
Response Message	Also reply message. Also secondary message. A SECS message with an even numbered function that is a response to a request message.
SDL File	Also SECS Definition Language file. A file consisting of definitions of SECS messages and definitions of items used in SECS messages. This file is used to validate SECS messages that are sent and received and to add various functionality having to do with them.

Term	Description
SECS-1	A serial-based message transfer protocol used for sending SECS messages. The SECS-I standard is SEMI E4.
SECS Message	Also just message. A communication message in the format specified by the SECS-II standard that is used to communicate with an equipment.
Transaction	A request message and its response message if required.
TransactionID	Also TxID. Every request message has a unique 32-bit integer identifier that identifies the transaction. Every response message has the same TransactionID as the request message it responds to. In the HSMS and SECS-1 protocols, this is called System Bytes.

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Using the SECS/GEM Module

Getting Started Tutorial

This tutorial will get you up and running with a demo project that uses the SECS/GEM Module.

Download and Installing the the Module

Go to the [Downloads](#) page and under the Device Connectivity section, download the SECS/GEM Module. Once the module is downloaded to your computer, install the module. This is done in the normal way that Ignition modules are installed by going to the Gateway webpage and selecting **Configure > System > Modules**. See the [Installing or Upgrading a Module](#) page for more information.

NAMING CONVENTIONS

Before getting started, it's a good idea to establish a naming convention when setting up your SECS/GEM module so you know at a glance what elements are related to each other. This is helpful when you have multiple pieces of equipment, simulators, databases, and equipment connections with keeping these elements organized.

Configuring the Database Connection

To setup a database connection in Ignition that you want to use with the SECS/GEM Module, go to the **Configure > Database > Connections** section on the Gateway webpage. The Database Connection name that is configured for this example is called **SecsGem**.

New in 7.9.12

The SECS/GEM Module has been tested with MySQL, SQL, and Oracle Servers. The SECS/GEM Demo project that you will upload later as part of this tutorial currently only supports MySQL. More information about configuring a database connection can be found in [Connecting to Databases](#).

Configuring the Simulator

The SECS/GEM Module can directly interact with a piece of equipment, but we can also use the built-in simulator. This tutorial will assume you are using a simulator. To configure a simulator, refer to the example below.

Example - Setting Up the Simulator

The instructions below will walk you through configuring the simulator. Most of the values can be left as defaults to get you up and running quickly.

1. Go to the Gateway webpage and select **Configure > SECS/GEM > Simulator**.
2. On the Simulator page of the Gateway, click on the **Create new Simulator** link.
3. **Simulator Name** - enter the simulator name (ie., **SimEquip**).
4. **Active Port** - use the default port number of **5000**. The port number needs to be unique if you're using multiple simulators.
5. **Passive Port** - use the default port number of **5000**. The port number needs to be unique if you're using multiple simulators.
6. **Device ID** - use the default ID of **0**. Each piece of equipment must have a unique device ID, and must be an integer.
7. **Connection Mode** - use the default **Passive mode**. The Connection Mode is a property that is configured in both the Simulator and Equipment Connection settings, and cannot be set to the same method. Alternating mode can be used, in which case, it will switch between Active mode and Passive mode until a connection is made.

Simulator

Main	
Simulator Name	SimEquip Choose a name for this simulator.
Simulator Description	
Enabled	☒ Disabling a simulator will prevent communication with an Equipment Connection. (default: true)
Active IP Address	localhost IP Address of Equipment Connection to connect to. Used when simulator is in Active mode. (default: localhost)
Active Port	5000 Port number of Equipment Connection to connect to. Used when simulator is in Active mode. (default: 5,000)
Passive IP Address	localhost IP Address of Ignition that an Equipment Connection will connect to if simulator is in passive mode. (default: localhost)
Passive Port	5000 Port number that an Equipment Connection will connect to if simulator is in passive mode. (default: 5,000)
Device ID	0 Unique identifier of equipment. Must be an integer. (default: 0)
Connection Mode	PASSIVE Method used to connect. (default: PASSIVE)

8. Click on the **Create New Simulator** button once you entered all your configuration settings. The window will refresh telling you the simulator was created successfully, and the status will toggle back and forth between "Connecting" and "Not Connected" until an equipment connection is created.

Simulators

✓ Successfully created new Simulator "SimEquip"

Simulator Name	Simulator Description	Status	
SimEquip		Not Connected	More ▾ edit
SimFour		Communicating	More ▾ edit

For more information on configuring a simulator, refer to the [SECS/GEM Simulator](#) page.

TRIAL MODE RESET

If you do not have a SECS/GEM Module license installed, the simulators and equipment connections will cause communication to be stopped when the Gateway Trial Mode expires in 2 hours. Reset the trial mode on the Gateway webpage to resume equipment and simulator communications. You can find more information about Trial Mode on the [About the Trial Period](#) page.

Creating an Equipment Connection

Once the simulator is configured, (or if there is an actual piece of equipment you wish to connect to), a connection to the Equipment must be made on the **Configure > SECS/GEM > Equipment** section of the Gateway webpage. To configure an equipment connection, refer to the example below.

Example - Creating the Equipment Connection

When creating the equipment connection, most of the settings can be left as the default values for this tutorial. Use the following instructions to configure the equipment connection.

1. On the **Equipment Connections** page, click on **Create new Equipment Connection...**
2. On the **Add Equipment Connection Step 1: Choose Type** page, select **Ethernet/HSMS Connection**, and click **Next**.
3. On the Equipment Connection page:
 - i. **Equipment Name** - enter the equipment name (i.e., **Equip_MySim**).
 - ii. **Enabled** - default set to **'true'**. Enables communication to the target equipment.
 - iii. **Active IP Address** - use the default of **localhost**.
 - iv. **Active Port** - use the default port number of **5000**. The Active Port number must match the port number used in the simulator configuration setup (i.e., 5000).
 - v. **Passive IP Address** - use the default of **localhost**.
 - vi. **Passive Port** - use the default port number of **5000**. The Passive Port number must match the port number used in the simulator configuration setup (i.e., 5000).
 - vii. **Connection Mode** - use the default **Active mode**.
 - viii. **Device ID** - the default ID is **0**. Each equipment requires a unique identifier, and must be an integer.

- ix. **Database Connection** - enter the database connection name (i.e., **SecsGem**). This is the database name that was configured in **Configuring the Database Connection** step.
- x. **Database Table Prefix** - enter a table prefix (i.e., **Sim_**). This is the prefix that will be prepended to each of the database table names that are automatically setup when the connection is made. If no prefix is specified, the table will still get created. The advantage of using a prefix is it helps keep database tables organized, and it is used to differentiate one equipment's tables from another equipment's tables. It also allows you to filter a project based on the prefix.
- xi. **SECS Definition Language (SDL) File** - use the default file **messages.sdl**. Validates sent and recieved messages.

Equipment Connection

Main	
Equipment Name	Equip_MySim Choose a name for equipment connection settings.
Equipment Description	
Enabled	☒ Disabling a connection will prevent communication to the target equipment. (default: true)
Active IP Address	localhost IP Address of equipment to connect to. Used when SECS/GEM module is in Active mode. (default: localhost)
Active Port	5000 Port number of equipment to connect to. Used when SECS/GEM module is in Active mode. (default: 5,000)
Passive IP Address	localhost IP Address of Ignition that an Equipment will connect to if SECS/GEM module is in passive mode. (default: localhost)
Passive Port	5000 Port number that an Equipment will connect to if SECS/GEM module is in passive mode. (default: 5,000)
Connection Mode	ACTIVE Method used to connect to equipment. (default: ACTIVE)
Device ID	0 Unique identifier of equipment. Must be an integer. (default: 0)
Database Connection	SecsGem This database connection will be used to send and receive data.
Database Table Prefix	Sim_ SECS/GEM database table names will use the specified prefix for this equipment connection. If no prefix is specified then no prefix will be used. (default:)
SECS Definition Language (SDL) file	Choose File No file chosen This file validates SECS messages sent and received. By default messages.sdl is used. (default: messages.sdl)

- xii. Once all the configuration settings are entered, click on the **Create new Equipment Connection** button. The Equipment Connection will try to connect to the simulator that you setup in **Configuring the Simulator** section. The window will refresh showing a status of "Connecting" until the equipment connection is created.

More information connecting to a tool can be found on the SECS/GEM Equipment Connections page.

Equipment Connection Status

The status of the Equipment Connection is shown in the Status column on the Equipment Connections webpage. When the status changes to "Communicating", the Equipment Connection successfully connected to the simulator.

Equipment Connections

Equipment Name	Equipment Description	SECS Definition Language (SDL) file	Connection Type	Status	
Equip_MySim		messages.sdl	Ethernet/HSMS Protocol	Communicating	More ▼ edit
EquipmentFour		messages.sdl	Ethernet/HSMS Protocol	Communicating	More ▼ edit

Download and Restore the Demo Project

At this point, the bare minimum requirements to start utilizing the SECS/GEM module have been met. However, there is a free SECS/GEM Demo Project available from the Downloads page in the Extra's tab on the Inductive Automation website: <https://inductiveautomation.com/downloads/extra-material>. This demo project contains resources to allow you to start interacting with the SECS/GEM module. If you wish to install the demo project, first download the project, and then restore the "secsgemdemoproject.proj" by going **Configure > System > Project** section of the Gateway webpage. More information can be found on the **Project Restore** page for how to restore a project.

Using the SECS/GEM Demo Project

Once you restored the demo project, launch a client from the Gateway webpage under the **Home** section to begin using the module. For more information, refer to the page on **Launching Clients**.

When the client is open, there will be three dropdown menus at the top left of the application. From the dropdowns, make the appropriate selections. If you followed the tutorial, the selections are shown below.

- **Datasource:** select the database connection (i.e, **SecsGem**)

Display Tabs

The Display Tabs allow you to see the detailed SECS messages associated with each request and reponse. There are five Display Tabs: Simple, Tree, Table, JSON, and Python. Select any request or response from the Search and Display table and toggle the Display Tabs to view the SECs messages in their respective formats.

Example - Display Tabs

The Display Tabs below show the detailed response message (S1F14) returned back from the Establish Comm request (S1F13).

Simple Tab

SimpleTreeTableJSONPython

S1F14, Establish Communications Request Acknowledge
Accept or deny Establish Communications Request (S1F13).

Reply expected: False
Direction: Response from Equipment Connection sent to Equipment

Body:
Establish Communications
Acknowledge Code

Accepted

Equipment Model Type

Software Revision Code:

Tree Tab

SimpleTreeTableJSONPython

header

doc

Establish Communications Request Acknowledge

stream

1

function

14

reply

False

body

1) COMMACK

Establish Communications Acknowledge Code

Accepted

B

0

Table Tab

Simple	Tree	Table	JSON	Python	
path		doc	cod...	format	value
header/doc		Establish Communications Request Acknowl...			
header/stream					1
header/function					14
header/reply					False
body/1) COMMACK		Establish Communications Acknowledge Code	Accept...	B	0

JSON Tab

SimpleTreeTableJSONPython

```
{
  "body": [
    {
      "codeDesc": "Accepted",
      "doc": "COMMACK, Establish Communications Acknowledge Code",
      "format": "B",
      "value": 0
    },
    []
  ],
  "header": {
    "doc": "Establish Communications Request Acknowledge",
    "function": 14,
    "reply": false,
    "stream": 1
  }
}
```

Python Tab

Simple Tree Table JSON Python

```
{u'body': [{u'codeDesc': 'Accepted',
            u'doc': 'COMMACK, Establish Communications Acknowledge Code',
            u'format': 'B',
            u'value': 0},
          []],
u'header': {u'doc': 'Establish Communications Request Acknowledge',
            u'function': 14,
            u'reply': False,
            u'stream': 1}}
```

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

SECS/GEM Equipment Connections

Connection Types

Equipment connections can be created using one of two different connection types: Ethernet through a TCP/IP network, or Serial using an RS-232 serial cable. Each uses a different protocol to communicate.

Ethernet Connections

Ethernet connections use the HSMS protocol for communication. The name of the standard for HSMS is SEMI E37, and are much faster than serial connections. The SECS/GEM module enables ethernet connections automatically and are the preferred method of connection.

Serial Connections

Serial connections use the SECS-I protocol for communication. The name of the standard for SECS-I is SEMI E4. To use a serial connection, the Serial Support Gateway Module must be installed in addition to the SECS/GEM module. The equipment must also have a direct serial connection to the Ignition Gateway server.

Equipment Connection Properties

When setting up an equipment connection through either ethernet or serial, there are various properties that can be configured.

Common Properties

Property Name	Description	Default
Equipment Name	The name to give this equipment connection.	
Equipment Description	A description of this equipment connection.	
Enabled	Whether the equipment connection is enabled or disabled. A disabled equipment connection will prevent communication between the Gateway and the equipment.	TRUE
Active IP Address	IP Address of the equipment to connect to when in Active mode.	localhost
Active Port	The Port number of the equipment to connect to when in Active mode, which must match the port number used to create the Simulaor.	5000
Passive IP Address	IP Address of the equipment to connect to when in Passive mode.	localhost
Passive Port	The Port number of the equipment to connect to when in Passive mode, which must match the port number used to create the Simulaor.	5000
Connection Mode	The Method used to connect to Equipment. • ACTIVE- Will attempt to connect to the equipment at the given Active IP Address and Active Port. • PASSIVE - Will listen for a connection from the equipment at the given Passive IP Address and Passive Port. • ALTERNATING - Will switch between Active mode and Passive mode until a connection is made.	ACTIVE
Device ID	A unique identifier of the equipment, which is defined by the equipment. Must be an integer.	0

Property Name	Description	Default
Database Connection	The Database Connection that will be used to store messages in for audit purposes as well as some configuration data.	
Database Table Prefix	The prefix that will be prepended to each of the database table names that get automatically setup when the connection is made. Used to help differentiate one equipment's tables from another equipment's tables. If no prefix is specified, then no prefix will be used.	
SECS Definition Language (SDL) File	The file that validates SECS messages sent and received between the Gateway and the equipment. If none is specified, a default messages.sdl file is used. See SECS Definition Language (SDL) File for the default file.	messages.sdl
T3 Reply Timeout	The number of seconds to wait for an expected SECS message reply.	45

Properties for Equipment Connections using the Ethernet/HSMS Connection Type

Connects to equipment using HSMS over TCP.

Property Name	Description	Default
Active IP Address	IP Address of the equipment to connect to when in Active mode.	localhost
Active Port	The Port number of the equipment to connect to when in Active mode, which must match the port number used to create the Simulaor.	5000
Passive IP Address	IP Address of the equipment to connect to when in Passive mode.	localhost
Passive Port	The Port number of the equipment to connect to when in Passive mode, which must match the port number used to create the Simulaor.	5000
Connection Mode	The Method used to connect to Equipment. • ACTIVE- Will attempt to connect to the equipment at the given Active IP Address and Active Port. • PASSIVE - Will listen for a connection from the equipment at the given Passive IP Address and Passive Port. • ALTERNATING - Will switch between Active mode and Passive mode until a connection is made.	ACTIVE
T5 Connect Separation Timeout	The number of seconds which must elapse between successive attempts to connect to the equipment after disconnection.	10
T6 Control Transaction Timeout	The number of seconds which a control transaction (such as LinkTest or Select) may remain open before it is considered a communications failure.	5
T7 Not Selected Timeout	The number of seconds which a TCP/IP connection can remain in NOT SELECTED state (i.e. no HSMS activity) before it is considered a communications failure. This timeout is only used for Passive Connection Mode.	10
T8 Network Intercharacter	The maximum number of seconds between successive bytes of a single HSMS message before it is considered a	5

Property Name	Description	Default
Timeout	communications failure. This applies to HSMS messages received from the equipment.	
Keep Alive Timeout	The number of seconds interval for sending LinkTest control messages for testing a connection. Automatically reconnects if the test fails. A value of 0 will disable it.	300

Properties for Equipment Connections using the Serial/SECS-I Connection Type

Connects to equipment serially using SECS-I.

Property Name	Description	Default
Serial Port Name	The name of the serial port, e.g. "COM1" or "/dev/ttyS0".	COM1
Data Bit Rate	The bit rate of data that is being sent and received. Possible values are: - BR_9600 - BR_4800 - BR_2400 - BR_1200 - BR_19200 - BR_150	BR_9600
T1 Inter-Character Timeout	The maximum number of seconds allowed for interruptions between characters being sent.	0.5
T2 Protocol Timeout	The maximum number of seconds for a lack of protocol response.	10
T4 Inter-Block Timeout	The maximum number of seconds allowed for interruptions in multi-block messages.	45
Retry Limit	The maximum number of send retries allowed.	3

Database Tables

Regardless of the type of equipment connection, database tables are created for each connection based on the prefix specified in the connection if they do not already exist.

CAUTION

No automatic maintenance (such as pruning or partitioning) is performed on these tables. Some of the tables can grow quite large, so it is up to the user to properly manage each set of tables from each equipment connection.

SECSGEM_EquipmentInfo Table

The SECSGEM_EquipmentInfo table is unique in that only one will get created for all databases that use the same database connection and will not use a prefix in its name. Each row of the SECSGEM_EquipmentInfo table contains information about a specific Equipment Connection.

SECSGEM_EquipmentInfo

Column Name	Datatype	Description
Equipment	String	The name of the equipment
Prefix	String	The prefix that the equipment connection uses for its unique tables.
Status	String	The status of the equipment connection.
SDL File	String	The SDL File that was uploaded for this equipment connection. If no file was uploaded, it will use the default messages.sdl.

Messages Table

The Messages table is where SECS messages are stored after they are sent or received. It will include a prefix in the name if one was specified in the equipment connection. Multiple equipment connections can use the same prefix which will have them write to the same messages table. Because this table records every message between the Gateway and equipment, it can grow quite large.

prefixMessages

Column Name	Datatype	Description
ID	Auto incrementing PrimaryID, Integer	The ID of the row.
Equipment	String	The name of the equipment.
StreamFunction	String	The Stream and Function such as "S1F1".
Direction	String	Whether the message was received or sent.
RequestResponse	String	Whether the message was a request or a response.
CommonID	String	A special identifier that will link together multiple related messages.
TxID	Integer	The transaction ID of the message.
Reply	Integer	Whether the message expects a reply or not.
Message	String	The JSON string message.
TimeSentReceived	DateTime	A datetime value of when the message was sent or received.

Errors Table

Any errors that occur are logged to the Errors table. It will include a prefix in the name if one was specified in the equipment connection. Multiple equipment connections can use the same prefix which will have them write to the same errors table.

prefixErrors

Column Name	Datatype	Description
ID	Auto incrementing PrimaryID, Integer	The ID of the row.
Equipment	String	The name of the equipment.
StreamFunction	String	The Stream and Function such as "S1F1". Only used if the error is related to a SECS Message
ErrorType	String	The type of error that occurred, such as Connection Error, Timeout, or JSON Syntax Error.
Error	String	The error message.
Time	DateTime	The date and time when the error occurred.

Creating an Equipment Connection

In order to communicate with an equipment or the built-in simulator, an Equipment Connection must be configured in the Gateway. This example walks you through configuring an equipment connection for a piece of equipment.

1. Go to the **Configure** section of the Gateway webpage and select **SECS/GEM > Equipment**.
2. On the **Equipment Connections** page, click on Create new **Equipment Connection...**
3. On the **Add Equipment Connection Step 1: Choose Type** page, select **Ethernet/HSMS Connection**, and click **Next**.
4. On the Equipment Connection page:
 - i. **Equipment Name** - enter equipment name, (i.e., **EquipOne**).
 - ii. **Enabled** - set to '**true**'. Enables communication to the target equipment.
 - iii. **Active IP Address** - use the default of **localhost**, or enter the IP address of equipment to connect to when in Active mode.
 - iv. **Active Port** - set the port number (i.e., **5007**). The Active and Passive port numbers must be unique for each piece of equipment. Note, the Active Port must match the port number used in the equipment or simulator configuration setup.

- v. **Passive IP Address** - use the default of **localhost**, or enter the IP address of the equipment to connect to when in Passive mode.
 - vi. **Passive Port** - set the port number (i.e., **5007**). The Active and Passive port numbers must be unique for each piece of equipment. Note, the Passive Port must match the port number used in the equipment or simulator configuration setup.
 - vii. **Connection Mode** - the default is set to **Active**. The Connection Mode is a property that is configured in both the Simulator and Equipment Connection settings, and cannot be set to the same method. Alternating mode can be used, in which case, it will switch between Active and Passive modes until a connection is made.
 - viii. **Device ID** - enter an equipment ID (i.e. **1**). Each piece of equipment must have a unique ID.
 - ix. **Database Connection** - enter the database that will be used to send and receive data (i.e., **SecsGem**).
 - x. **Database Table Prefix** - enter a prefix to prepend to the database table names. (i.e. **EquipOne_**). A prefix will be prepended to each of the database table names that are automatically setup when the connection is made. This is an optional setting, so if no prefix is specified, the table will still get created. The advantage of using a prefix is it helps keep database tables organized, and used to differentiate one equipment's tables from another equipment's tables. It also allows you to filter a project based on the prefix.
 - xi. **SECS Definition Language (SDL) File** - the default file is **messages.sdl**. Validates sent and recieved messages.
5. Once all the configuration settings are entered, click on the **Create new Equipment Connection**. The window will refresh showing a status of "Connecting" until the equipment connection is created. When the status changes to "Communicating", the Equipment Connection is successfully created.

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

SECS Definition Language (SDL) File

The SDL File

The SDL file contains definitions that all SECS messages for that equipment connection are validated against. Any messages that fail to validate against the SDL file will be inserted into the Errors table for the equipment connection instead. The SDL file uses JSON syntax to define datatypes, SECS Items, and SECS Messages.

Each equipment connection can accept one SDL file. However, if no file is specified, the default 'messages.sdl' file will be used. The SECS message definitions in the default SDL file are based on the definitions in the SECS-II standard (SEMI E5-0712) and the GEM standard (SEMI E30-0307).

The SDL files can be edited in the Gateway Webpage using the built in SDL editor, or they can be downloaded from the Equipment Connections page so that they can be edited in a text editor and re-uploaded at a later time.

SDL File Format

The SDL file should be formatted using JSON syntax. It follows a simple structure, which is demonstrated in the example below. There are four main parts to the SDL file:

- **doc** - The doc comes first and provides any overall documentation for the file. We recommend good documentation so that it is easy to make changes if needed.
- **formats** - The formats list out all of the datatype formats that the items will use. It does not need to include datatypes that are not used by the items.
- **items** - The items are used by the messages to relay information to and from the equipment. The items are sorted alphabetically, with each item under its corresponding letter. Each letter can have any number of items under it.

- **messages** - The messages handle communication with the equipment. Each message is sorted by stream, with each function listed in order under its corresponding stream. There can be multiple streams under messages, and multiple functions under each stream.

Example SDL File

Example SDL File

```
{
  "doc":[
    "The doc for the file."
  ],
  "formats":{
    "A":{
      "doc":"ASCII"
    },
    "I2":{
      "doc":"Signed Integer Two Bytes"
    }
  },
  "items":{
    "A":{
      "ALTX":{
        "doc":[
          "ALTX, Alarm Text"
        ],
        "formats":[
          "A"
        ],
        "max":120
      }
    },
    "C":{
      "CCODE":{
        "doc":[
          "CCODE, Command Code",
          "Each command code corresponds to a unique process operation the machine is capable of performing."
        ],
        "formats":[
          "A",
          "I2",
          "I4",
          "U2",
          "U4"
        ]
      }
    },
    "messages":{
      "S1":{
        "doc":[
          "Stream 1 Equipment Status",
          "This stream provides a means for exchanging information about the status of the",
          "equipment, including its current mode, depletion of various consumable items, and the status of transfer operations."
        ]
      }
    }
  }
}
```


an 'I' and an 'F ', which makes it easier to use when specifying all of those datatypes in the Item Definition.

SDL Formats Definition Example

SDL Formats Definition Example

```
{
  "A":{
    "doc":"ASCII"
  },
  "B":{
    "doc":"Binary"
  },
  "L":{
    "doc":"List"
  },
  "I1":{
    "doc":"Signed Integer One Byte"
  },
  "I2":{
    "doc":"Signed Integer Two Bytes"
  },
  "I4":{
    "doc":"Signed Integer Four Bytes"
  },
  "I8":{
    "doc":"Signed Integer Eight Bytes"
  },
  "F4":{
    "doc":"Floating Point Four Bytes"
  },
  "F8":{
    "doc":"Floating Point Eight Bytes"
  },
  "I":{
    "doc":"Signed Integer",
    "formats":[
      "I1",
      "I2",
      "I4",
      "I8"
    ]
  },
  "F":{
    "doc":"Floating Point",
    "formats":[
      "F4",
      "F8"
    ]
  }
}
```

SDL Item Definitions

The SDL Item Definitions list out all possible items that are used in SECS Message definitions. If a SECS Message attempts to use an item that isn't defined in the SDL file, it will throw an error.

SDL Item Definition Example

SDL Item Definition Example

```
{
  "A":{
    "ACKC7":{
      "doc":[
        "ACKC7, Acknowledge Code"
      ],
      "formats":[
        "B"
      ],
      "bytes":1,
      "codes":{
        "0":"Accepted",
        "1":"Permission not granted",
        "2":"Length error",
        "3":"Matrix overflow",
        "4":"PPID not found",
        "5":"Mode unsupported",
        "6":"Command will be performed with completion signaled later",
        ">":[
          6,
          "Other error"
        ]
      }
    }
  }
}
```

Each item has a list of properties that define what the item is.

Property Name	Description	Example Formatting
doc	Provides information about what the item is along with any other useful information.	<pre>{
"doc":[
"TIME, Time of day",
"12</pre>
formats	A list of the possible datatypes that the item can be. Can also be a list of items, as in the second example. The type can be a list of CATTRDAT A objects, or any of the other listed objects.	<pre>{
"formats":[
"A",
"B",
"Bool</pre> <pre>{
"formats":[
{
"format":"L",
</pre>
bytes	How many bytes the item must have. If bytes is greater than 1, then the item is an array of bytes.	<pre>{
"bytes":1
 }</pre>

Property Name	Description	Example Formatting
	Only applies to single byte datatypes such as Binary, Boolean, U1 and I1.	
codes	Lists possible values an item can have and what each means. A value of ">" means any value greater than the largest code value.	<pre>{
"0":"Accepted",
"1":"Permission not gr</pre>
max	The maximum number of characters that a value can have. Only applies to ASCII datatypes.	<pre>{
"max":40
}</pre>
pattern	A java regular expression that the ASCII value must match.	<pre>{
"pattern":"^[0-9]{6}&#124;[0-9]{8}\$"
}</pre>

Property Name	Description	Example Formatting
	Only applies to ASCII datatypes.	

SDL Message Definitions

The message definitions list out all possible SECS messages that will be used when communicating with the equipment. There can be many streams listed, each with multiple functions.

SDL Message Definition Example

```

{
  "S6F1":{
    "doc":[
      "Trace Data Send",
      "This function sends samples to the host according to the trace setup
done by S2,F23."
    ],
    "block":"multiple",
    "direction":"h<-e",
    "or":[
      {
        "reply":true
      },
      {
        "reply":false
      }
    ],
    "CommonID":[
      "TRID"
    ],
    "realtime":true,
    "SQLTags":true,
    "autoReply":{
      "header":{
        "stream":6,
        "function":2,
        "reply":false
      },
      "body":{
        "format":"B",
        "value":0
      }
    },
    "body":[
      "TRID",
      "SMPLN",
      "STIME",
      [
        "SV",
        "..."
      ]
    ]
  }
}

```

Property Name	Description	Example
doc	Provides information about what the message is along with any other useful information.	<pre>{
"doc":[
"Are You There R</pre>
block	The size in bytes of the entire SECS message. There are two possible values: "single" - The message will not be larger than 254 bytes. "multiple" - The message will be larger than 254 bytes. This property is only important for maintaining compatibility with the SECS-I protocol through Serial connection.	<pre>{
"block":"single"
}</pre>
direction	Determines whether the message should come from the equipment, the host, or either. Possible values are: "h->e"- The message goes from the host to the equipment. "h<-e" - The message goes from the equipment to the host. "h<->e" - The message goes either direction.	<pre>{
"direction":"h<->e"
}</pre>
reply	True if the message expects a reply, false if not.	<pre>{
"reply":true
}</pre>

Property Name	Description	Example
autoReply	An auto response with a given message when a given message is received. If included in a message, the example will auto reply with a S1F2 message.	<pre>{
"autoReply":{
"header":{</pre>
body	A list of items that the stream will implement. The body list can contain lists of items, or the special "...", which signifies a variable list. In the second example, depending on the value of HCACK, any amount of a list of CNAME and CPACK can be returned.	<pre>{
"body":[
"DATAID",
</pre>
or	Will list multiple possibilities. Can also exist within a body definition.	<pre>{
"or":[
{
"direction</pre>
CommonID	Used to identify certain messages, especially when used to group multiple incoming messages together. In the example, a trace will have multiple S6F1 messages as the status variable is traced over time. The CommonID will use the trace ID to link all messages from the same trace together, allowing for easy retrieval from the database.	<pre>{
"S6F1":{
"doc":[
"T</pre>

Property Name	Description	Example
realtime	If true, will be processed by a Message Handler script. The message handler must be named onSecsGemRealtimeUpdate. Will use the CommonID for updates. See SECS/GEM Messages for more information on the Message Handler.	<pre>{
"realtime":true
}</pre>
SQLTags	If true, will create Tags in the SECSGEM provider automatically, using the CommonID as a folder name. Will create Tags in the location as follows: SECSGEM- >EquipmentName→Stream and Function- >CommonID→message value tags Can also be overridden to use a folder name other than the CommonID, like in the second example.	<pre>{
"SQLTags":true
} </code></pre>

SDL Message Defaults

The messages section typically starts with a default set of properties before the list of SECS messages. These default property values will be used in the message when the property is not present in the message definition.

SDL Message Defaults

```

{
  "defaults":{
    "doc":[
      "Default values for properties that are missing."
    ],
    "block":"single",
    "reply":false,
    "direction":"h<->e",
    "body":null
  }
}

```

SDL File Editor

In the **Equipment Connections** section, you can make edits to the equipment's SDL file right in the Gateway Webpage by clicking the **More** button and selecting the 'edit sdl file' option.

⚠ CAUTION

After editing the SDL File, the equipment connection must be restarted for the changes to take effect.

EquipmentOne SDL File

Equipment Connection must be restarted for changes to take effect.

Save SDL File



▼ object {4}	
⋮	▶ doc [1]
⋮	▶ formats {20}
⋮	▶ items {15}
⋮	▶ messages {11}

Once in the SDL file editor, there are a few things that can be done. The six dots to the left  of each row circled in blue above, allow you to change the order of the rows by clicking and dragging a row and dropping it in a new spot.

The buttons near the top circled in green apply to the document as a whole:

Button	Description
	Expands all nodes.
	Collapses all nodes.
	Undo the last edit made to the file.
	Redo the last Undo.

Finally, the buttons circled in red  bring up a context menu with a few options that allow you to add and remove rows and change the row type:

 Type

 Sort

 Insert

 Duplicate

 Remove

Option	Description
Type	Change the type of the row. Possible options are: • Auto - Automatically determines the type based on the value. • Array - Contains an ordered collection of values. • Object - Contains an unordered set of key/value pairs. • String - Determined from the value, but always returned as a string.
Sort	Sorts the rows in either ascending or descending order. • Ascending - Sort the children of this array in ascending order. • Descending - Sort the children of this array in descending order.
Insert	Inserts a new row before the selected row. Possible types are: • Auto - Automatically determines the type based on the value. • Array - Contains an ordered collection of values. • Object - Contains an unordered set of key/value pairs. • String - Determined from the value, but always returned as a string.
Duplicate	Duplicates the row, including all sub elements
Remove	Removes the row and all sub elements.

Each row can be drilled into by clicking on the expand arrow to the left of the element, which allows you to see the sub elements. Many times, these sub elements can also be drilled into to find more sub elements. Once you find the element you want to modify, you can simply edit the text to make a change.

object{4}
doc [1]
formats {20}
items {15}
A {12}
ACCESSMODE {3}
doc [1]
0 : ACCESSMODE, Load Port Access Mode
formats [1]
0 : U1
codes {2}
0 : Manual
1 : Auto
ACKA {3}
ACKC5 {4}

Once all the edits are made, clicking the '**Save SDL File**' button at the top right of the screen will save your changes. However, the changes won't take effect until the equipment connection is restarted.

Restart the Equipment

To restart the equipment, simply click the **More** button next to the Equipment on the Equipment Connections page, and select the '**restart**' option.

Default SDL File

The default '**message.sdl**' file is used when no other SDL file is specified.

Default SDL File Example

The messages.sdl Default File.

```

{
  "doc":["SECS stream/functions and items are listed and defined in this
document."],

  "formats":{
    "A":{"doc":"ASCII"},
    "B":{"doc":"Binary"},
    "B64":{"doc":"Binary encoded as Base64 string"},
    "J":{"doc":"JIS-8"},
    "L":{"doc":"List"},
    "Bool":{"doc":"Boolean"},
    "Local":{"doc":"Two Byte Character String"},
    "U1":{"doc":"Unsigned Integer One Byte"},
    "U2":{"doc":"Unsigned Integer Two Bytes"},
    "U4":{"doc":"Unsigned Integer Four Bytes"},
    "U8":{"doc":"Unsigned Integer Eight Bytes"},
    "I1":{"doc":"Signed Integer One Byte"},
    "I2":{"doc":"Signed Integer Two Bytes"},
    "I4":{"doc":"Signed Integer Four Bytes"},
    "I8":{"doc":"Signed Integer Eight Bytes"},
    "F4":{"doc":"Floating Point Four Bytes"},
    "F8":{"doc":"Floating Point Eight Bytes"},
    "I":{"doc":"Signed Integer", "formats":["I1","I2","I4","I8"]},
    "U":{"doc":"Unsigned Integer", "formats":["U1","U2","U4","U8"]},
    "F":{"doc":"Floating Point", "formats":["F4","F8"]}},

  "items":{
    "A":{
      "ACCESSMODE":{
        "doc":["ACCESSMODE, Load Port Access Mode"],
        "formats":["U1"],
        "codes":{
          "0":"Manual",
          "1":"Auto"}},
      "ACKA":{
        "doc":["ACKA, Indicates success of a request"],
        "formats":["Bool"],
        "codes":{
          "0":"False",
          "1":"True"}},
      "ACKC5":{
        "doc":["ACKC5, Acknowledge Code"],
        "formats":["B"],
        "bytes":1,
        "codes":{
          "0":"Accepted",
          ">":["0","Error, not accepted"]}},
      "ACKC6":{

```

```

    "doc":["ACKC6, Acknowledge Code"],
    "formats":["B"],
    "bytes":1,
    "codes":{
        "0":"Accepted",
        ">":["0","Error, not accepted"]}},
"ACKC7":{
    "doc":["ACKC7, Acknowledge Code"],
    "formats":["B"],
    "bytes":1,
    "codes":{
        "0":"Accepted",
        "1":"Permission not granted",
        "2":"Length error",
        "3":"Matrix overflow",
        "4":"PPID not found",
        "5":"Mode unsupported",
        "6":"Command will be performed with completion signaled
later",
        ">":["6","Other error"]}},
"ACKC10":{
    "doc":["ACKC10, Acknowledge Code"],
    "formats":["B"],
    "bytes":1,
    "codes":{
        "0":"Accepted for display",
        "1":"Message will not be displayed",
        "2":"Terminal not available"}},
"ALCD":{
    "doc":["ALCD, Alarm Code Byte",
        "bit 8 = 1 means alarm set",
        "bit 8 = 0 means alarm cleared",
        "bit 7-1 is alarm category"],
    "formats":["B"],
    "bytes":1,
    "codes":{
        "0":"Not Used",
        "1":"Alarm Cleared, Personal Safety",
        "2":"Alarm Cleared, Equipment Safety",
        "3":"Alarm Cleared, Parameter Control Warning",
        "4":"Alarm Cleared, Parameter Control Error",
        "5":"Alarm Cleared, Irrecoverable Error",
        "6":"Alarm Cleared, Equipment Status Warning",
        "7":"Alarm Cleared, Attention flags",
        "8":"Alarm Cleared, Data integrity",
        "><":["8,128","Alarm Cleared, Other categories"],
        "128":"Not Used",
        "129":"Alarm Set, Personal Safety",
        "130":"Alarm Set, Equipment Safety",

```

```

        "131": "Alarm Set, Parameter Control Warning",
        "132": "Alarm Set, Parameter Control Error",
        "133": "Alarm Set, Irrecoverable Error",
        "134": "Alarm Set, Equipment Status Warning",
        "135": "Alarm Set, Attention flags",
        "136": "Alarm Set, Data integrity",
        ">": [136, "Alarm Cleared, Other categories"]}],
    "ALED": {
        "doc": ["ALED, Alarm Enable/Disable Code"],
        "formats": ["B"],
        "bytes": 1,
        "codes": {
            ">": [127, "Enable Alarm"],
            "<": [128, "Disable Alarm"]}
    },
    "ALID": {
        "doc": ["ALID, Alarm Identification"],
        "formats": ["U"]
    },
    "ALTX": {
        "doc": ["ALTX, Alarm Text"],
        "formats": ["A"],
        "max": 120
    },
    "ATTRDATA": {
        "doc": ["ATTRDATA, Contains a specific attribute value for a
specific object."],
        "formats":
[{"format": "L", "item": "ATTRDATA"}, "B", "Bool", "A", "F", "U", "I"]
    },
    "ATTRID": {
        "doc": ["ATTRID, Identifier for an attribute for a specific type of
object."],
        "formats": ["A", "U"]
    },
    "C": {
        "CAACK": {
            "doc": ["CAACK, Carrier Action Acknowledge Code"],
            "formats": ["U1"],
            "codes": {
                "0": "Acknowledge, command has been performed",
                "1": "Invalid command",
                "2": "Can not perform now",
                "3": "Invalid data or argument",
                "4": "Acknowledge, request will be performed with completion
signaled later by an event",
                "5": "Rejected. Invalid state",
                "6": "Command performed with errors"}
        },
        "CARRIERACTION": {
            "doc": ["CARRIERACTION, Specifies the action requested for a
carrier"],
            "formats": ["A"]
        },
        "CARRIERID": {
            "doc": ["CARRIERID, The identifier of a carrier"],

```

```
    "formats":["A"]},
  "CATTRDATA":{
    "doc":["CATTRDATA, The value of a carrier attribute"],
    "formats":
[{"format":"L","item":"CATTRDATA"},"B","Bool","A","F","U","I"]},
    "CATTRID":{
      "doc":["CATTRID, The name of a carrier attribute"],
      "formats":["A"]},
    "CCODE":{
      "doc":["CCODE, Command Code",
        "Each command code corresponds to a unique process operation
the machine is capable of performing."],
      "formats":["A","I2","I4","U2","U4"]},
    "CEED":{
      "doc":["CEED, Collection event or trace enable/disable code"],
      "formats":["Bool"],
      "bytes":1,
      "codes":{
        "0":"Disable",
        "1":"Enable"}}},
    "CEID":{
      "doc":["CEID, Collected Event ID"],
      "formats":["U"]},
    "COMMACK":{
      "doc":["COMMACK, Establish Communications Acknowledge Code"],
      "formats":["B"],
      "bytes":1,
      "codes":{
        "0":"Accepted",
        "1":"Denied, Try Again"}}},
    "CPACK":{
      "doc":["CPACK, Command Parameter Acknowledge Code"],
      "formats":["B"],
      "bytes":1,
      "codes":{
        "1":"Parameter Name (CPNAME) does not exist",
        "2":"Illegal Value specified for CPVAL",
        "3":"Illegal Format specified for CPVAL",
        ">":["3","Other equipment specific error"]}},
    "CPNAME":{
      "doc":["CPNAME, Command Parameter Name"],
      "formats":["A"],
      "max":40},
    "CPVAL":{
      "doc":["CPVAL, Command Parameter Value"],
      "formats":["A","B","Bool","U","I"]}},
  "D":{
    "DATAID":{
      "doc":["DATAID, Data ID"],
```

```

        "formats":["U"]},
    "DATALENGTH":{
        "doc":["DATALENGTH, Total bytes to be sent"],
        "formats":["U"]},
    "DRACK":{
        "doc":["DRACK, Define Report Acknowledge Code",
            "If an error condition is detected the entire message is
rejected (i.e., partial changes are not allowed)."],
        "formats":["B"],
        "bytes":1,
        "codes":{
            "0":"Accepted",
            "1":"Denied. Insufficient space",
            "2":"Denied. Invalid format",
            "3":"Denied. At least one RPTID already defined",
            "4":"Denied. At least VID does not exist",
            ">":[4,"Other errors"]}},
    "DSPER":{
        "doc":["DSPER, Data sample period.",
            "DSPER has two allowable formats:",
            "Format 1: hhmmss, 6 bytes",
            "Format 2: hhmmsscc, 8 bytes",
            "Where hh is hours, mm is minutes, ss is seconds, and cc is
centiseconds.",
            "Equipment shall either (1) support only Format 1, or (2)
support both Format 1 and Format 2.",
            "Equipment shall document which formats it accepts.",
            "Equipment which supports Format 2 need not necessarily
support a minimum DSPER of 1 centisecond,",
            "nor a trace resolution of 1 centisecond, but equipment
suppliers shall document its trace performance limits."],
        "formats":["A"],
        "pattern":"^[0-9]{6}|[0-9]{8}$"}},
    "E":{
        "EAC":{
            "doc":["EAC, Equipment Acknowledge Code"],
            "formats":["B"],
            "bytes":1,
            "codes":{
                "0":"Acknowledge",
                "1":"Denied. At least one constant does not exist",
                "2":"Denied. Busy",
                "3":"Denied. At least one constant out of range",
                ">":[3,"Other equipment specific error"]}},
        "ECDEF":{
            "doc":["ECDEF, Equipment Constant Default value"],
            "formats":
[{"format":"L","item":"ECDEF"},"A","B","Bool","J","F","U","I"]},
        "ECID":{

```

```
    "doc":["ECID, Equipment Constant ID"],
    "formats":["U"]},
  "ECMAX":{
    "doc":["ECMAX, Equipment Constant Maximum value"],
    "formats":["A","B","Bool","J","F","U","I"]},
  "ECMIN":{
    "doc":["ECMIN, Equipment Constant Minimum value"],
    "formats":["A","B","Bool","J","F","U","I"]},
  "ECNAME":{
    "doc":["ECNAME, Equipment Constant Name"],
    "formats":["A"]},
  "ECV":{
    "doc":["ECV, Equipment Constant Value"],
    "formats":
[{"format":"L","item":"ECV"},"A","B","Bool","J","F","U","I"]},
  "EDID":{
    "doc":["EDID, Expected Data Identification"],
    "formats":["A","B","U","I"]},
  "ERACK":{
    "doc":["ERACK, Enable/Disable Event Acknowledge Code"],
    "formats":["B"],
    "bytes":1,
    "codes":{
      "0":"Accepted",
      "1":"Denied. At least one CEID does not exist",
      ">":["1","Other Errors"]}},
  "ERRCODE":{
    "doc":["ERRCODE, Code identifying an error"],
    "formats":["U"],
    "codes":{
      "0":"No error",
      "1":"Unknown object in Object Specifier",
      "2":"Unknown target object type",
      "3":"Unknown object instance",
      "4":"Unknown attribute name",
      "5":"Read-only attribute - access denied",
      "6":"Unknown object type",
      "7":"Invalid attribute value",
      "8":"Syntax error",
      "9":"Verification error",
      "10":"Validation error",
      "11":"Object identifier in use",
      "12":"Parameters improperly specified",
      "13":"Insufficient parameters specified",
      "14":"Unsupported option requested",
      "15":"Busy",
      "16":"Not available for processing",
      "17":"Command not valid for current state",
      "18":"No material altered",
```

```
"19": "Material partially processed",
"20": "All material processed",
"21": "Recipe specification related error",
"22": "Failed during processing",
"23": "Failed while not processing",
"24": "Failed due to lack of material",
"25": "Job aborted",
"26": "Job stopped",
"27": "Job cancelled",
"28": "Cannot change selected recipe",
"29": "Unknown event",
"30": "Duplicate report ID",
"31": "Unknown data report",
"32": "Data report not linked",
"33": "Unknown trace report",
"34": "Duplicate trace ID",
"35": "Too many data reports",
"36": "Sample period out of range",
"37": "Group size too large",
"38": "Recovery action currently invalid",
"39": "Busy with another recovery currently unable to perform
the recovery",
"40": "No active recovery action",
"41": "Exception recovery failed",
"42": "Exception recovery aborted",
"43": "Invalid table element",
"44": "Unknown table element",
"45": "Cannot delete predefined",
"46": "Invalid token",
"47": "Invalid parameter",
"48": "Load port does not exist",
"49": "Load port already in use",
"50": "Missing Carrier",
"32768": "Action will be performed at earliest opportunity",
"32769": "Action can not be performed now",
"32770": "Action failed due to errors",
"32771": "Invalid command",
"32772": "Client Alr",
"32773": "Duplicate ClientID",
"32774": "Invalid ClientType",
"32775": "IncompatibleVersions",
"32776": "Unrecognized ClientID (Client not currently
connected)",
"32777": "Failed (Completed Unsuccess-fully)",
"32778": "Failed (Unsafe) – External intervention required",
"32779": "Sensor-Detected Obstacle",
"32780": "Material Not Sent",
"32781": "Material Not Received",
"32782": "Material Lost",
```

```
        "32783": "Hardware Failure",
        "32784": "Transfer Cancelled" }},
    "ERRTEXT": {
        "doc": ["ERRTEXT, Text string describing the error noted in the
corresponding ERRCODE"],
        "formats": ["A"],
        "max": 120 }},
    "G": {
        "GRANT": {
            "doc": ["GRANT, Grant code"],
            "formats": ["B"],
            "bytes": 1,
            "codes": {
                "0": "Permission Granted",
                "1": "Busy, Try Again",
                "2": "No Space Available",
                "3": "Duplicate DATAID",
                ">": [3, "Equipment Specific Error Code"] }},
        "GRANT6": {
            "doc": ["GRANT6, Permission to send"],
            "formats": ["B"],
            "bytes": 1,
            "codes": {
                "0": "Permission granted",
                "1": "Busy, try again",
                "2": "Not interested",
                ">": [2, "Other err"] }},
    "H": {
        "HACK": {
            "doc": ["HACK, Host Command Parameter Acknowledge Code"],
            "formats": ["B"],
            "bytes": 1,
            "codes": {
                "0": "Acknowledge, command has been performed",
                "1": "Command does not exist",
                "2": "Cannot perform now",
                "3": "At least one parameter is invalid",
                "4": "Acknowledge, command will be performed with completion
signaled later by an event",
                "5": "Rejected, Already in Desired Condition",
                "6": "No such object exists" }},
    "L": {
        "LENGTH": {
            "doc": ["LENGTH, Length of the service program or process program
in bytes."],
            "formats": ["U"] },
        "LRACK": {
            "doc": ["LRACK, Link Report Acknowledge Code"],
            "formats": ["B"],
```

```
    "bytes":1,
    "codes":{
        "0":"Accepted",
        "1":"Denied. Insufficient space",
        "2":"Denied. Invalid format",
        "3":"Denied. At least one CEID link already defined",
        "4":"Denied. At least one CEID does not exist",
        "5":"Denied. At least one RPTID does not exist",
        ">":[5,"Other errors"]}}},
    "M":{
        "MDLN":{
            "doc":["MDLN, Equipment Model Type"],
            "formats":["A"],
            "max":20},
        "MEXP":{
            "doc":["MEXP, Message expected in the form SxxFyy where x is
stream and y is function."],
            "formats":["A"],
            "max":6},
        "MF":{
            "doc":["MF, Material Format code",
                "Items with format A will be a unit identifier for one of the
special SECS generic units, as specified in § 12."],
            "formats":["B","A"],
            "bytes":1,
            "codes":{
                "1":"Quantities in wafers",
                "2":"Quantities in cassette",
                "3":"Quantities in die or chips",
                "4":"Quantities in boats",
                "5":"Quantities in ingots",
                "6":"Quantities in leadframes",
                "7":"Quantities in lots",
                "8":"Quantities in magazines",
                "9":"Quantities in packages",
                "10":"Quantities in plates",
                "11":"Quantities in tubes",
                "12":"Quantities in waterframes",
                "13":"Quantities in carriers",
                "14":"Quantities in substrates"}}},
        "MHEAD":{
            "doc":["MHEAD, SECS message block header associated with message
block in error."],
            "formats":["B"]},
        "MID":{
            "doc":["MID, Material ID"],
            "formats":["B","A"],
            "max":80}},
    "O":{
```

```
"OBJACK":{
  "doc":["OBJACK, Acknowledge code"],
  "formats":["U1"],
  "codes":{
    "0":"Successful completion of requested data",
    "1":"Error"}},
"OBJSPEC":{
  "doc":["OBJSPEC, A text string that has an internal format and
that is used to point to a specific object instance.",
  "The string is formed out of a sequence of formatted
substrings, each specifying an object's type and identifier.",
  "The substring format has the following four fields:",
  "object type, colon character ':', object identifier, greater-
than symbol '>'"],
  "where the colon character ':' is used to terminate an object
type and the \"greater than\" symbol '>' is used to terminate an identifier
field.",
  "The object type field may be omitted where it may be
otherwise determined.",
  "The final '>' is optional."],
  "formats":["A"]},
"OBJTYPE":{
  "doc":["OBJTYPE, Identifier for a group or class of objects",
  "All objects of the same type must have the same set of
attributes available."],
  "formats":["A","U"]},
"OFLACK":{
  "doc":["OFLACK, Acknowledge code for OFF-LINE request"],
  "formats":["B"],
  "bytes":1,
  "codes":{
    "0":"OFF-LINE Acknowledge"}},
"ONLACK":{
  "doc":["ONLACK, Acknowledge code for ON-LINE request"],
  "formats":["B"],
  "bytes":1,
  "codes":{
    "0":"ON-LINE Accepted",
    "1":"ON-LINE Not Allowed",
    "2":"Equipment Already ON-LINE"}}},
"P":{
  "PARAMNAME":{
    "doc":["PARAMNAME, The name of a parameter in a request"],
    "formats":["A"]},
  "PARAMVAL":{
    "doc":["PARAMVAL, The value of the parameter named in PARAMNAME",
    "Values that are lists are restricted to lists of single items
of the same format type."],
    "formats":
```

```
[{"format":"L","item":"PARAMVAL"},"A","B","Bool","F","U","I"]},
  "PORTACTION":{
    "doc":["PORTACTION, tThe action to be performed on a port"],
    "formats":["A"]},
  "PPARM":{
    "doc":["PPARM, Process Parameter"],
    "formats":["A","Bool","F","U","I"]},
  "PPBODY":{
    "doc":["PPBODY, Process Program Body",
    "The process program describes to the equipment, in its own
language, the actions to be taken in processing the material it receives."],
    "formats":["A","B","B64","U","I"]},
  "PPGNT":{
    "doc":["PPGNT, Process Program Grant status"],
    "formats":["B"],
    "bytes":1,
    "codes":{
      "0":"OK",
      "1":"Already have",
      "2":"No space",
      "3":"Invalid PPID",
      "4":"Busy, try later",
      "5":"Will not accept",
      ">":["5","Other error"]}},
  "PPID":{
    "doc":["PPID, Process Program ID",
    "The format used in the PPID will be host dependent.",
    "For internal use of the equipment, the PPID can be treated as
a unique B pattern.",
    "If the local equipment is not prepared to display the
transmitted code, the display should be in hexadecimal form."],
    "formats":["A"],
    "max":120},
  "PRJOBID":{
    "doc":["PRJOBID, Text string which uniquely identifies a process
job"],
    "formats":["A"]},
  "PRPAUSEEVENT":{
    "doc":["PRPAUSEEVENT, The list of event identifiers, which may be
sent as an attribute value to a process job.",
    "When a process job encounters one of these events it will
pause, until it receives the PRJobCommand RESUME."],
    "formats":[{"format":"L","item":"CEID"}]},
  "PRPROCESSSTART":{
    "doc":["PRPROCESSSTART, Indicates that the process resource start
processing immediately when ready."],
    "formats":["Bool"],
    "codes":{
      "0":"Manual Start",
```

```
        "1": "Automatic Start" } },
    "PRRECIPEMETHOD": {
        "doc": ["PRRECIPEMETHOD, Indicates the recipe specification type, whether tuning is applied and which method is used."],
        "formats": ["U1"],
        "codes": {
            "1": "Recipe only",
            "2": "Recipe with variable tuning" } },
    "PRSTATE": {
        "doc": ["PRSTATE, Enumerated value"],
        "formats": ["U1"] },
    "PTN": {
        "doc": ["PTN, Material Port number"],
        "formats": ["B", "U1"],
        "bytes": 1 } },

    "R": {
        "RCMD": {
            "doc": ["RCMD, Remote Command"],
            "formats": ["A"] },
        "RCPPARNM": {
            "doc": ["RCPPARNM, The name of a recipe variable parameter."],
            "formats": ["A"],
            "max": 256 },
        "RCPPARVAL": {
            "doc": ["RCPPARVAL, The initial setting assigned to a recipe variable parameter."],
            "formats":
[{"format": "L", "item": "RCPPARVAL"}, "A", "B", "Bool", "F", "U", "I"],
            "max": 80 },
        "RCPSPEC": {
            "doc": ["RCPSEC, Recipe specifier. The object specifier of a recipe."],
            "formats": ["A"] },
        "REPGSZ": {
            "doc": ["REPGSZ, Reporting Group Size"],
            "formats": ["U"] },
        "RPTID": {
            "doc": ["RPTID, Report ID"],
            "formats": ["U"] } },

    "S": {
        "SHEAD": {
            "doc": ["SHEAD, Stored header related to the transaction timer"],
            "formats": ["B"] },
        "SLOTID": {
            "doc": ["SLOTID, Used to reference material by slot (a position that holds material/substrates) in a carrier.",
                "This item may be implemented as an array in some messages."],
            "formats": ["U1"] },
```

```
"SMPLN":{
  "doc":["SMPLN, Sample Number"],
  "formats":["U","I"]},
"SOFTREV":{
  "doc":["SOFTREV, Software Revision Code"],
  "formats":["A"],
  "max":20},
"STIME":{
  "doc":["STIME, Sample Time",
    "12 or 16 bytes, or Extended format as specified by the
TimeFormat equipment constant value setting.",
    "12-byte format: YYMMDDhhmmss",
    "16-byte format: YYYYMMDDhhmmsscc",
    "Extended (max 32 byte) format: format YYYY-MM-
DDThh:mm:ss.sTZD (see SEMI E148)"],
  "formats":["A"],
  "max":32},
"SV":{
  "doc":["SV, Status Variable"],
  "formats":
[{"format":"L","item":"SV"},"B","Bool","A","J","F","U","I"]},
"SVID":{
  "doc":["SVID, Status Variable ID",
    "Status variables may include any parameter that can be
sampled in time such as temperature or quantity."],
  "formats":["U"]},
"SVNAME":{
  "doc":["SVNAME, Status Variable Name"],
  "formats":["A"]},
"T":{
  "TEXT":{
    "doc":["TEXT, a single line of characters."],
    "formats":["A","B","Local","U","I"]},
  "TIACK":{
    "doc":["TIACK, Time Acknowledge Code"],
    "formats":["B"],
    "bytes":1,
    "codes":{
      "0":"OK",
      "1":"Error, not done"}},
  "TIAACK":{
    "doc":["TIAACK, Equipment Acknowledgement Code"],
    "formats":["B"],
    "bytes":1,
    "codes":{
      "0":"Everything correct",
      "1":"Too many SVIDs",
      "2":"No more traces allowed",
      "3":"Invalid period",
```

```

        "4": "Unknown SVID specified",
        "5": "Invalid REPGSZ",
        ">": [63, "Equipment specified error"]}},
    "TID": {
        "doc": ["TID, Terminal number"],
        "formats": ["B"],
        "bytes": 1,
        "codes": {
            "0": "Single or main terminal",
            ">": [0, "Additional terminals at the same equipment"]}},
    "TIME": {
        "doc": ["TIME, Time of day",
            "12, 16 bytes, or Extended format as specified by the
TimeFormat equipment constant value setting.",
            "12-byte format YYMMDDhhmmss",
            "16-byte format YYYYMMDDhhmmsscc",
            "Extended (max 32 byte) format YYYY-MM-DDThh:mm:ss.sTZD (see
SEMI E148)"],
        "formats": ["A"],
        "max": 32},
    "TOTSMP": {
        "doc": ["TOTSMP, Total Samples"],
        "formats": ["U"]},

    "TRID": {
        "doc": ["TRID, Trace Request ID."],
        "formats": ["U"]}},
    "U": {
        "UNITS": {
            "doc": ["UNITS, Units Identifier"],
            "formats": ["A", "B"]}},
    "V": {
        "V": {
            "doc": ["V, Variable data"],
            "formats":
[{"format": "L", "item": "V"}, "A", "B", "Bool", "J", "F", "U", "I"]},
        "VID": {
            "doc": ["VID, Variable ID"],
            "formats": ["U"]}}},
    "messages": {

        "defaults": {
            "doc": ["Default values for properties that are missing."],
            "block": "single",
            "reply": false,
            "direction": "h<->e",
            "body": null},

        "S1": {

```

```
    "doc":["Stream 1 Equipment Status",
        "This stream provides a means for exchanging information about the
status of the",
        "equipment, including its current mode, depletion of various
consumable items, and the status of transfer operations."],

    "S1F0":{
        "doc":["Abort Transaction",
            "Used in lieu of an expected reply to abort a transaction.",
            "Function 0 is defined in every stream and has the same
meaning in every stream."],
        "block":"single","direction":"h<->e"},

    "S1F1":{
        "doc":["Are You There Request",
            "Establishes if the equipment is on-line.",
            "A function 0 response to this message means the communication
is inoperative.",
            "In the equipment, a function 0 is equivalent to a timeout on
the receive timer after issuing S1,F1 to the host."],
        "block":"single","direction":"h<->e","reply":true,
        "autoReply":{"header":
{"stream":1,"function":2,"reply":false},"body":[]}},

    "S1F2":{
        "doc":["On Line Data",
            "Data signifying that the equipment is alive.",
            "Exception: The host sends a zero-length list to the
equipment."],
        "block":"single",
        "or":[
            {"direction":"h<-e","body":["MDLN","SOFTREV"]},
            {"direction":"h->e","body":[]}]},

    "S1F3":{
        "doc":["Selected Equipment Status Request",
            "A request to the equipment to report the values of selected
status variables.",
            "Exception: A zero-length list means report all SVIDs."],
        "block":"single","direction":"h->e","reply":true,
        "body":["SVID","..."]},

    "S1F4":{
        "doc":["Selected Equipment Status Data",
            "The equipment reports the value of each status variable
requested in the order requested.",
            "The host remembers the names of values requested.",
            "Exception: A zero-length list item for SV means that SVID
does not exist."],
```

```

        "block": "multiple", "direction": "h<-e",
        "body": ["SV", "..."]},

    "S1F11": {
        "doc": ["Status Variable Namelist Request",
            "A request to the equipment to report the name and units of
selected status variables.",
            "Exception: A zero length means report all SVIDs."],
        "block": "single", "direction": "h->e", "reply": true,
        "body": ["SVID", "..."]},

    "S1F12": {
        "doc": ["Status Variable Namelist Reply",
            "The equipment reports to the host the name and units of the
requested status variables.",
            "Exception: Zero-length A items for both SVNAME and UNITS
indicates that the SVID does not exist."],
        "block": "multiple", "direction": "h<-e",
        "body": [{"SVID", "SVNAME", "UNITS"}, "..."]},

    "S1F13": {
        "doc": ["Establish Communications Request",
            "The purpose of this message is to establish communication
between an equipment and host at an application level.",
            "The purpose of this message is to provide a formal means of
initializing communications at a logical application level both on",
            "power-up and following a break in communications. It should
be the following any period where host and Equipment SECS",
            "applications are unable to communicate. An attempt to send an
Establish Communications Request (S1,F13) should be repeated",
            "at programmable intervals until an Establish Communications
Acknowledge (S1,F14) is received within the transaction timeout",
            "period with an acknowledgement code accepting the
establishment.",
            "Exception: The host sends a zero-length list to the
equipment."],
        "block": "single", "reply": true,
        "autoReply": {"header":
{"stream": 1, "function": 14, "reply": false}, "body": [{"format": "B", "value": 0}, []]},
        "or": [
            {"direction": "h->e", "body": []},
            {"direction": "h<-e", "body": ["MDLN", "SOFTREV"]}]}},

    "S1F14": {
        "doc": ["Establish Communications Request Acknowledge",
            "Accept or deny Establish Communications Request (S1F13).",
            "MDLN and SOFTREV are on-line data and are valid only if
COMMACK = 0.",
            "Exception: The host sends a zero-length list for item 2 to

```

```
the equipment."],
    "block": "single",
    "or": [
        {"direction": "h<-e", "body": ["COMMACK", ["MDLN", "SOFTREV"]]},
        {"direction": "h->e", "body": ["COMMACK", []]}],

    "S1F15": {
        "doc": ["Request OFF-LINE",
            "The host requests that the equipment transition to the OFF-
LINE state."],
        "block": "single", "direction": "h->e", "reply": true,

    "S1F16": {
        "doc": ["OFF-LINE Acknowledge"],
        "block": "single", "direction": "h<-e",
        "body": "OFLACK"},

    "S1F17": {
        "doc": ["Request ON-LINE",
            "The host requests that the equipment transition to the ON-
LINE state"],
        "block": "single", "direction": "h->e", "reply": true,

    "S1F18": {
        "doc": ["ON-LINE Acknowledge"],
        "block": "single", "direction": "h<-e",
        "body": "ONLACK"},

    "S1F23": {
        "doc": ["Collection Event Namelist Request",
            "This function allows the host to retrieve information about
what collection event IDs are available in the equipment",
            "and which data values (DVVALs) are valid for each collection
event.",
            "Exception: A zero-length list means send information for all
CEIDs."],
        "block": "single", "direction": "h->e", "reply": true,
        "body": ["CEID", "..."]},

    "S1F24": {
        "doc": ["Collection Event Namelist",
            "The equipment reports to the host the information of the
collection events and associated VIDs of the CEIDs",
            "requested with the S1F23 message.",
            "A listed VID can be conditionally or unconditionally
associated with the CEID;",
            "it is the responsibility of the equipment supplier to
document whether the conditional VIDs are reported with S1F24.",
            "Exception: When both CENAME and the list of associated VIDs
```

```
are zero-length items, this indicates that ECID does not exist."],
    "block":"multiple","direction":"h<-e",
    "body":[["CEID","CENAME",["VID","..."]],"..."]}],

    "S2":{
        "doc":["Stream 2 Equipment Control and Diagnostics",
            "Messages which deal with control of the equipment from the host.",
            "This includes all remote operations and equipment self-diagnostics
and calibration but specifically",
            "excludes the control operations which are associated with material
transfer (see Stream 4)",
            "loading of executive and boot programs (Stream 8), and all file
and operating system calls",
            "(Streams 10, 13). See also continuations in Stream 17."],

        "S2F0":{
            "doc":["Abort Transaction",
                "Used in lieu of an expected reply to abort a transaction.",
                "Function 0 is defined in every stream and has the same
meaning in every stream."],
            "block":"single","direction":"h<->e"},

        "S2F13":{
            "doc":["Equipment Constant Request",
                "A request to the equipment to report the values of selected
constants.",
                "Constants such as for calibration, servo gain, alarm limits,
data collection mode, and other values that are changed infrequently can be
obtained using this message.",
                "Exception: A zero-length list means report all ECVs according
to a predefined order."],
            "block":"single","direction":"h->e","reply":true,
            "body":["ECID","..."]},

        "S2F14":{
            "doc":["Equipment Constant Data",
                "The equipment reports the value of each constant requested in
the order requested.",
                "Exception: A zero-length list item for ECV means that ECID
does not exist. The list format for this data item is not allowed, except in this
case."],
            "block":"multiple","direction":"h<-e",
            "body":["ECV","..."]},

        "S2F15":{
            "doc":["New Equipment Constant Send",
                "Change one or more equipment constants."],
            "block":"single","direction":"h->e","reply":true,
            "body":[["ECID","ECV"],"..."]},
```

```
"S2F16":{
  "doc":["New Equipment Constant Acknowledge",
    "Acknowledges or denies modification of equipment constants.",
    "If EAC contains a non-zero error code, the equipment should
not change any of the ECIDs specified in S2F15."],
  "block":"single","direction":"h<-e",
  "body":"EAC"},

"S2F17":{
  "doc":["Date and Time Request",
    "Get the date and time from the equipment.",
    "Useful to check equipment time base or for equipment to
synchronize with the host time base."],
  "block":"single","direction":"h<->e","reply":true},

"S2F18":{
  "doc":["Date and Time Data",
    "Actual time data from equipment.",
    "Exception: A zero-length item means no time exists"],
  "block":"single","direction":"h<->e",
  "body":"TIME"},

"S2F23":{
  "doc":["Trace Initialize Send",
    "Report the current values of selected status variables at a
regular interval of time.",
    "Status variables exist at all times. This function provides a
way to sample a subset of those status variables as a function of time.",
    "The trace data is returned on S6,F1 and is related to the
original request by the TRID Multiple trace requests may be made to",
    "that equipment allowing it. If equipment receives S2F23 with
the same TRID as a trace function that is currently in progress,",
    "the equipment should terminate the old trace and then
initiate the new trace. A trace function currently in progress may be",
    "terminated by S2,F23 with TRID of that trace and TOTSMF =
0.",
    "If S2F23 is multi-block, it must be preceded by the
S2F39/S2F40 Inquire/Grant transaction. Some equipment may support",
    "only single-Block S6F1, and may refuse a S2F23 message which
would cause a multi-block S6F1.",
    "Each equipment shall document its trace performance limits.
The Host Computer shall not send an S2F23 which exceeds the",
    "equipment's performance limits, or the equipment may operate
incorrectly.",
    "TRID = Trace Request ID, DSPER = Data Sample Period, TOTSMF =
Total samples to be made.",
    "REPGSZ = Reporting Group Size, SVID = Status Variable ID"],
  "block":"multiple","direction":"h->e","reply":true,"CommonID":
```

```

["TRID"],
    "body":["TRID","DSPER","TOTSM","REPGSZ",["SVID","..."]]},

"S2F24":{
    "doc":["Trace Initialize Acknowledge"],
    "block":"single","direction":"h<-e",
    "body":{
        "doc":["Equipment Acknowledgment Code"],
        "formats":["B"],
        "bytes":1,
        "codes":{
            "0":"Everything correct",
            "1":"Too many SVIDs",
            "2":"No more traces allowed",
            "3":"Invalid period",
            "4":"Unknown SVID specified",
            "5":"Invalid REPGSZ",
            ">":["63","Equipment specified error"]}}},

"S2F29":{
    "doc":["Equipment Constant Namelist Request",
        "A request to the equipment to report information about
selected constants.",
        "This function allows the host to retrieve basic information
about what equipment constants are available in the equipment.",
        "Exception: A zero-length list means send information for all
ECIDs."],
    "block":"single","direction":"h->e","reply":true,
    "body":["ECID","..."]]},

"S2F30":{
    "doc":["Equipment Constant Namelist",
        "The equipment reports to the host information about certain
constants.",
        "Exception: Zero-length A items for ECNAME, ECMIN, ECMAX,
ECDEF, and UNITS indicates that the ECID does not exist."],
    "block":"multiple","direction":"h<-e",
    "body":["ECID","ECNAME","ECMIN","ECMAX","ECDEF","UNITS"],"..."]]},

"S2F31":{
    "doc":["Date and Time Set Request",
        "Set the date and time in the equipment.",
        "Useful to synchronize the equipment time with the host time
base."],
    "block":"single","direction":"h->e","reply":true,
    "body":"TIME"},

"S2F32":{
    "doc":["Date and Time Set Acknowledge"],

```

```
"block":"single","direction":"h<-e",
"body":"TIACK"},

"S2F33":{
  "doc":["Define Report",
    "The purpose of this message is for the host to define a group
of reports for the equipment.",
    "The type of report to be transmitted is designated by a
BOOLEAN \"Equipment Constant\".",
    "An \"Equipment Constant Value\" of \"False\" means that an
\"Event Report\" (S6F11) will be sent, and a value of \"True\" means",
    "that an \"Annotated Event Report\" (S6F13) will be sent. If
S2F33 is Multi-block, it must be preceded by the S2F39/S2F40 Inquire/Grant
transaction.",
    "Exception 1: A list of zero-length following DATAID deletes
all report definitions and associated links. See S2F35 (Link Event/Report).",
    "#Exception 2: A list of zero-length following RPTID deletes
report type RPTID. All CEID links to this RPTID are also deleted."],
  "block":"multiple","direction":"h->e","reply":true,
  "body":["DATAID",[["RPTID",["VID","..."]],"..."]]},

"S2F34":{
  "doc":["Define Report Acknowledge"],
  "block":"single","direction":"h<-e",
  "body":"DRACK"},

"S2F35":{
  "doc":["Link Event Report",
    "The purpose of this message is for the host to link n reports
to an event (CEID).",
    "These linked event reports will default to 'disabled' upon
linking.",
    "That is, the occurrence of an event would not cause the
report to be sent until enabled.",
    "See S2F37 for enabling reports.",
    "If S2F35 is Multi-block, it must be preceded by the
S2F39/S2F40 Inquire/Grant transaction.",
    "#Exception: A list of zero length following CEID deletes all
report links to that event."],
  "block":"multiple","direction":"h->e","reply":true,
  "body":["DATAID",[["CEID",["RPTID","..."]],"..."]]},

"S2F36":{
  "doc":["Link Event Report Acknowledge",
    "If an error condition is detected the entire message is
rejected (i.e., partial changes are not allowed)."],
  "block":"single","direction":"h<-e",
  "body":"LRACK"},
```

```
"S2F37":{
  "doc":["Enable/Disable Event Report",
    "The purpose of this message is for the host to enable or
disable reporting for a group of events (CEIDs).",
    "The reporting for unlisted (CEIDs) is not affected.",
    "Exception: A list of zero length following <CEED> means all
CEIDs."],
  "block":"single","direction":"h->e","reply":true,
  "body":["CEED",["CEID","..."]]},

"S2F38":{
  "doc":["Enable/Disable Event Report Acknowledge",
    "If an error condition is detected the entire message is
rejected, i.e., partial changes are not allowed."],
  "block":"single","direction":"h<-e",
  "body":["ERACK"]},

"S2F39":{
  "doc":["Multi-block Inquire",
    "If a S2,F23, S2,F33, S2,F35, S2,F45, or S2,F49 message is
more than one block, this transaction must precede the message."],
  "block":"single","direction":"h->e","reply":true,
  "body":["DATAID","DATALENGTH"]]},

"S2F40":{
  "doc":["Multi-block Grant",
    "Grant permission to send multi-block message."],
  "block":"single","direction":"h<-e",
  "body":["GRANT"]},

"S2F41":{
  "doc":["Host Command Send",
    "The Host requests the Equipment perform the specified remote
command with the associated parameters."],
  "block":"single","direction":"h->e","reply":true,
  "body":["RCMD",[["CPNAME","CPVAL"],"..."]]},

"S2F42":{
  "doc":["Host Command Acknowledge",
    "If command is not accepted due to one or more invalid
parameters (i.e., HCAK = 3),",
    "then a list of invalid parameters will be returned containing
the parameter name and reason for being invalid."],
  "block":"single","direction":"h<-e",
  "body":["HCAK",[["CNAME","CPACK"],"..."]]}},

"S3":{
  "doc":["Stream 3 Materials Status",
    "The functions of the material status stream are used to
```

```
communicate information and actions related to material",
    "including carriers and material-in-process, time-to-completion
information, and extraordinary material occurrences."],

    "S3F0":{
        "doc":["Abort Transaction",
            "Used in lieu of an expected reply to abort a transaction.",
            "Function 0 is defined in every stream and has the same
meaning in every stream."],
        "block":"single","direction":"h<->e"},

    "S3F15":{
        "doc":["Materials Multi-Block Inquire",
            "This message requests permission to send a multi-block
message based upon a maximum length of the total message.",
            "It must be sent prior to sending any multi-block primary
message in Stream 3."],
        "block":"single","direction":"h->e","reply":true,
        "body":["DATAID","DATALENGTH"]},

    "S3F16":{
        "doc":["Materials Multi-Block Grant",
            "This message grants or denies permission to send a multi-
block primary message in Stream 3."],
        "block":"single","direction":"h<-e",
        "body":"GRANT"},

    "S3F17":{
        "doc":["Carrier Action Request",
            "This message requests an action to be performed for a
specified carrier.",
            "If multi-block, this message must be preceded by the
S3F15/F16 transaction.",
            "Exception: If variable list has no items, then no carrier
attributes are included.",
            "If CARRIERID is not a zero-length item, then PTN may be
omitted (a zero-length item)",
            "ATTRID and ATTRDATA may be substituted for CATTRID and
CATTRDATA respectively.",
            "ReticlePodLocationID may be used as one of <CATTRID> when the
CARRIERACTION is PodRelease and the carrier is not at a Load Port."],
        "block":"multiple","direction":"h->e","reply":true,
        "body":["DATAID","CARRIERACTION","CARRIERID","PTN",
[[["CATTRID","CATTRDATA"],"..."]]],

    "S3F18":{
        "doc":["Carrier Action Acknowledge",
            "This message acknowledges the carrier action request.",
            "Exception: If variable list contains no items then no errors
```

```
exist"],
    "block":"single","direction":"h<-e",
    "body":["CAACK",[["ERRCODE","ERRTEXT"],"..."]]},

    "S3F25":{
        "doc":["Port Action Request",
            "This message requests an action be performed for a port.",
            "Exception: If variable list contains no items, then no
parameters are provided."],
        "block":"single","direction":"h->e","reply":true,
        "body":["PORTACTION","PTN",[["PARAMNAME","PARAMVAL"],"..."]]},

    "S3F26":{
        "doc":["Port Action Acknowledge",
            "This message acknowledges the port action request.",
            "Exception: If variable list contains no item then no errors
exist."],
        "block":"single","direction":"h<-e",
        "body":["CAACK",[["ERRCODE","ERRTEXT"],"..."]]},

    "S3F27":{
        "doc":["Change Access",
            "The Host requests the Equipment to change the Access Mode for
the specified Load Ports.",
            "ACCESSMODE specifies the desired Access Mode. PTN specifies a
desired Load Port Number.",
            "If variable list contains no items then the command applies
to all Load Ports on the equipment.",
            "If any specified port is already in the specified Access
Mode, then the Equipment shall accept the command,",
            "and toggle all loadports to specified mode.",
            "If the Equipment is unable to change one or more of specified
Port(s) to the specified Access Mode,",
            "then the Equipment shall accept the command (with appropriate
response acknowledgement)",
            "and shall change only the Access Mode of those Port(s)
allowed by the equipment,",
            "supplying the host with an indication that not all ports were
successfully changed."],
        "block":"single","direction":"h->e","reply":true,
        "body":["ACCESSMODE",[["PTN"],"..."]]},

    "S3F28":{
        "doc":["Change Access Acknowledge",
            "If the command is successful, CAACK = 0, and variable list
size = 0.",
            "If the command was successful for some ports, CAACK = 6, and
variable list size > 0."],
        "block":"single","direction":"h<-e",
```

```
    "body":["CAACK",[[{"PTN","ERRCODE","ERRTEXT"},"..."]]]},

    "S5":{
        "doc":["Stream 5 Exception Handling",
            "This stream contains messages regarding binary and analog
            equipment exceptions.",
            "Exceptions are classified into two categories: errors and
            alarms."],

        "S5F0":{
            "doc":["Abort Transaction",
                "Used in lieu of an expected reply to abort a transaction.",
                "Function 0 is defined in every stream and has the same
                meaning in every stream."],
            "block":"single","direction":"h<->e"},

        "S5F1":{
            "doc":["Alarm Report Send",
                "This message reports a change in or presence of an alarm
                condition.",
                "One message will be issued when the alarm is set and one
                message will be issued when the alarm is cleared.",
                "Irrecoverable errors and attention flags may not have a
                corresponding clear message."],
            "block":"single","direction":"h<-e","reply":true,
            "autoReply":{"header":
{"stream":5,"function":2,"reply":false},"body":{"format":"B","value":0}},
            "body":["ALCD","ALID","ALTX"]},

        "S5F2":{
            "doc":["Alarm Report Acknowledge",
                "Acknowledge or error."],
            "block":"single","direction":"h->e",
            "body":["ACKC5"]},

        "S5F3":{
            "doc":["Enable/Disable Alarm Send",
                "This message will change the state of the enable bit in the
                equipment.",
                "The enable bit determines if the alarm will be sent to the
                host.",
                "Alarms which are not controllable in this way are unaffected
                by this message.",
                "Exception: A zero-length item for ALID means all alarms."],
            "block":"single","direction":"h->e","reply":true,
            "body":["ALED","ALID"]},

        "S5F4":{
            "doc":["Enable/Disable Alarm Acknowledge",
```

```
        "Acknowledge or error."],
        "block":"single","direction":"h<-e",
        "body":"ACKC5"}},

    "S5F5":{
        "doc":["List Alarms Request",
            "This message requests the equipment to send binary and analog
alarm information to the host.",
            "A zero-length item means send all possible alarms regardless
of the state of ALED."],
        "block":"single","direction":"h->e","reply":true,
        "body":"ALID..."}},

    "S5F6":{
        "doc":["List Alarm Data",
            "This message contains the alarm data known to the
equipment.",
            "A zero-length item returned for ALCD or ALTX means that value
does not exist."],
        "block":"multiple","direction":"h<-e",
        "body":[["ALCD","ALID","ALTX"],"..."]}},

    "S6":{
        "doc":["Stream 6 Data Collection",
            "This stream is intended to cover the needs of in-process
measurements and equipment monitoring."],

        "S6F0":{
            "doc":["Abort Transaction",
                "Used in lieu of an expected reply to abort a transaction.",
                "Function 0 is defined in every stream and has the same
meaning in every stream."],
            "block":"single","direction":"h<->e"},

        "S6F1":{
            "doc":["Trace Data Send",
                "This function sends samples to the host according to the
trace setup done by S2F23.",
                "Trace is a time-driven form of equipment status.",
                "Even if S6F1 is multi-block, it is not preceded by an
Inquire/Grant transaction, because the Host S2F23 is an implicit grant.",
                "Some equipment may support only single-block S6F1, and may
refuse an S2F23 (Trace Initiate Send) message which would cause a multi-block
S6,F1.",
                "Exception: A zero-length STIME means no value is given and
that the time is to be derived from SMPLN along with knowledge of the request.",
                "TRID = Trace Request ID, SMPLN = Sample Number, STIME =
Sample Time, SV = Status Variable Value"],
            "block":"multiple","direction":"h<-e","or":[{"reply":true},
```

```
{"reply":false}],
    "CommonID":["TRID"],"realtime":true,"SQLTags":true,
    "autoReply":{"header":
{"stream":6,"function":2,"reply":false},"body":{"format":"B","value":0}},
    "body":["TRID","SMPLN","STIME",["SV","..."]]},

    "S6F2":{
        "doc":["Trace Data Acknowledge",
            "Acknowledge or error."],
        "block":"single","direction":"h->e",
        "body":"ACKC6"},

    "S6F5":{
        "doc":["Multi-block Data Send Inquire",
            "If the discrete data report S6F3, F9, F11, F13 can involve
more than one block, this transaction must precede the transmission."],
        "block":"single","direction":"h<-e","reply":true,
        "autoReply":{"header":
{"stream":6,"function":6,"reply":false},"body":{"format":"B","value":0}},
        "body":["DATAID","DATALENGTH"]]},

    "S6F6":{
        "doc":["Multi-block Grant"],
        "block":"single","direction":"h->e",
        "body":"GRANT6"},

    "S6F11":{
        "doc":["Event Report Send",
            "Sends a report when an event occurs.",
            "The purpose of this message is for the equipment to send a
defined, linked, and enabled group of reports to the host upon the",
            "occurrence of an event (CEID).",
            "If S6F11 is Multi-block, it must be preceded by the
S6,F5/S6,F6 Inquire/Grant transaction.",
            "Exception: If there are no reports linked to the event a
'null' report is assumed.",
            "A zero-length list for number of reports means there are no
reports linked to the given CEID."],
        "block":"multiple","direction":"h<-e","reply":true,
        "CommonID":["DATAID"],"realtime":true,"SQLTags":["DATAID","CEID"],
        "autoReply":{"header":
{"stream":6,"function":12,"reply":false},"body":{"format":"B","value":0}},
        "body":["DATAID","CEID",[[["RPTID",["V","..."]],"..."]]},

    "S6F12":{
        "doc":["Event Report Acknowledge"],
        "block":"single","direction":"h->e",
        "body":"ACKC6"},

```

```
"S6F15":{
  "doc":["Event Report Request",
    "The purpose of this message is for the host to demand a given
report group from the equipment."],
  "block":"single","direction":"h->e","reply":true,
  "body":"CEID"},

"S6F16":{
  "doc":["Event Report Data",
    "Equipment sends reports linked to given CEID to host.",
    "Identical to structure of S6F11.",
    "A zero-length list for number of reports means there are no
reports linked to the given CEID."],
  "block":"multiple","direction":"h<-e",
  "body":["DATAID","CEID",[["RPTID",["V","..."]],"..."]]},

"S6F19":{
  "doc":["Individual Report Request",
    "The purpose of this message is for the host to request a
defined report from the equipment."],
  "block":"single","direction":"h->e","reply":true,
  "body":"RPTID"},

"S6F20":{
  "doc":["Individual Report Data",
    "Equipment sends variable data defined for the given RPTID to
the host."],
  "block":"multiple","direction":"h<-e",
  "body":["V","..."]]},

"S7":{
  "doc":["Stream 7 Process Program Management",
    "The functions in this stream are used to manage and transfer
process programs.",
    "Process programs are the equipment-specific descriptions that
determine the procedure to be",
    "conducted on the material by a single piece of equipment.",
    "Methods are provided to transfer programs as well as establish
the link between the process program",
    "and the material to be processed with that program."],

  "S7F0":{
    "doc":["Abort Transaction"],
    "block":"single","direction":"h<->e"},

  "S7F1":{
    "doc":["Process Program Load Inquire",
      "This message is used to initiate the transfer of a process
program or to select from stored programs.",
      "The message may be used to initiate the transfer of an
```

```
unformatted process program (S7F3/S7F4)",
    "or a formatted process program (S7F23/S7F24),
(S7F31/S7F32)."],
    "block": "single", "direction": "h<->e", "reply": true,
    "body": ["PPID", "LENGTH"]},

    "S7F2": {
        "doc": ["Process Program Load Grant",
            "This message gives permission for the process program to be
loaded."],
        "block": "single", "directions": "h<->e",
        "body": "PPGNT"},

    "S7F3": {
        "doc": ["Process Program Send",
            "The program is sent.",
            "If S7F3 is multi-block, it must be preceded by the S7F1/S7F2
Inquire/Grant transaction."],
        "block": "multiple", "direction": "h<->e", "reply": true,
        "body": ["PPID", "PPBODY"]},

    "S7F4": {
        "doc": ["Process Program Acknowledge"],
        "block": "single", "direction": "h<->e",
        "body": "ACKC7"},

    "S7F5": {
        "doc": ["Process Program Request",
            "This message is used to request the transfer of a process
program."],
        "block": "single", "direction": "h<->e", "reply": true,
        "body": "PPID"},

    "S7F6": {
        "doc": ["Process Program Data",
            "Exception: A zero-length list means request denied."],
        "block": "multiple", "direction": "h<->e",
        "body": {"or": [{"PPID", "PPBODY"}, []]}},

    "S7F17": {
        "doc": ["Delete Process Program Send",
            "This message is used by the host to request the equipment to
delete process programs from equipment storage.",
            "Exception: If variable list size = 0, delete all."],
        "block": "single", "direction": "h->e", "reply": true,
        "body": ["PPID", "..."]},

    "S7F18": {
        "doc": ["Delete Process Program Acknowledge"],
```

```
        "block": "single", "direction": "h<-e",
        "body": "ACKC7"},

    "S7F19": {
        "doc": ["Current EPPD Request",
            "This message is used to request the transmission of the
current equipment process program directory (EPPD).",
            "This is a list of all the PPIDs of the process programs
stored in the equipment."],
        "block": "single", "direction": "h->e", "reply": true},

    "S7F20": {
        "doc": ["Current EPPD Data",
            "This message is used to transmit the current EPPD."],
        "block": "multiple", "direction": "h<-e",
        "body": ["PPID", "..."]},

    "S7F23": {
        "doc": ["Formatted Process Program Send",
            "This message allows movement of formatted process programs
between a piece of equipment and its host system.",
            "The values of MDLN and SOFTREV are obtained from the PCD used
to generate the process program.",
            "If S7F23 is multi-block, it must be preceded by the S7F1/F2
Inquire/Grant transaction."],
        "block": "multiple", "direction": "h<->e", "reply": true,
        "body": ["PPID", "MDLN", "SOFTREV", [{"CCODE",
["PPARM", "..."]}], "..."]]},

    "S7F24": {
        "doc": ["Formatted Process Program Acknowledge",
            "Acknowledges reception of a formatted process program at its
destination and whether the process program",
            "was accepted by the interpreter.",
            "A returned status of \"accepted\" by the interpreter means
only that the message is understood.",
            "The validity of the contents of the process program is
determined through a separate transaction (S7F27/S7F28)."],
        "block": "single", "direction": "h<->e",
        "body": "ACKC7"},

    "S7F25": {
        "doc": ["Formatted Process Program Request",
            "This message is used by either equipment or host to request a
particular process program from the other."],
        "block": "single", "direction": "h<->e", "reply": true,
        "body": "PPID"},

    "S7F26": {
```

```
        "doc":["Formatted Process Program Data",
            "This message transfers a process program in response to a
request for the PPID.",
            "The values of MDLN and SOFTREV are obtained from the PCD used
to generate the process program.",
            "Exception: A zero length list indicates the request was
denied."],
        "block":"multiple","direction":"h<->e",
        "body":["PPID","MDLN","SOFTREV",["CCODE",
["PPARM","..."]],"..."]]},

    "S9":{
        "doc":["Stream 9 System Errors",
            "This stream provides a method of informing the host that a
message block has been received",
            "which cannot be handled or that a timeout on a transaction
(receive) timer has occurred.",
            "The messages indicate either a Message Fault or a Communications
Fault has occurred but",
            "do not indicate a Communications Failure has occurred."],

        "S9F0":{
            "doc":["Abort Transaction",
                "Used in lieu of an expected reply to abort a transaction.",
                "Function 0 is defined in every stream and has the same
meaning in every stream."],
            "block":"single","direction":"h<->e"},

        "S9F1":{
            "doc":["Unrecognized Device ID",
                "The device ID in the message header did not correspond to any
known device ID in the node detecting the error."],
            "block":"single","direction":"h<-e",
            "body":"MHEAD"},

        "S9F3":{
            "doc":["Unrecognized Stream Type",
                "The equipment does not recognize the stream in the message
header."],
            "block":"single","direction":"h<-e",
            "body":"MHEAD"},

        "S9F5":{
            "doc":["Unrecognized Function Type",
                "The equipment does not recognize the function in the message
header."],
            "block":"single","direction":"h<-e",
            "body":"MHEAD"},
    }
```

```
"S9F7":{
  "doc":["Illegal Data",
    "This message indicates that the stream and function were
recognized, but the message format was incorrect."],
  "block":"single","direction":"h<-e",
  "body":"MHEAD"},

"S9F9":{
  "doc":["Transaction Timer Timeout",
    "This message indicates that the host failed to respond to a
request within the expected amount of time. The transaction has been aborted.",
    "It is up to the host to respond to this error in an
appropriate manner to keep the system operational."],
  "block":"single","direction":"h<-e",
  "body":"SHEAD"},

"S9F11":{
  "doc":["Data Too Long",
    "This message to the host indicates that the equipment has
been sent more data than it can handle."],
  "block":"single","direction":"h<-e",
  "body":"MHEAD"},

"S9F13":{
  "doc":["Conversation Timeout",
    "Data were expected but none were received within a reasonable
length of time. Resources have been cleared."],
  "block":"single","direction":"h<-e",
  "body":["MEXP","EDID"]}},

"S10":{
  "doc":["Stream 10 Terminal Services",
    "The functions of this stream is to pass textual messages
between operator terminals attached to",
    "processing and/or testing equipment and the host. The
equipment makes no attempt to interpret",
    "the text of the message, but merely passes it from terminal
keyboard to the host or from the host to the display of the terminal.",
    "Management of human response times to information displayed
on terminals is the responsibility of the host."],

  "S10F0":{
    "doc":["Abort Transaction",
      "Used in lieu of an expected reply to abort a transaction.",
      "Function 0 is defined in every stream and has the same
meaning in every stream."],
    "block":"single","direction":"h<->e"},

  "S10F3":{
```

```
"doc":["Terminal Display, Single"],
"block":"single","direction":"h->e","reply":true,
"body":["TID","TEXT"]}},

"S10F4":{
  "doc":["Terminal Display, Single Acknowledge"],
  "block":"single","direction":"h<-e",
  "body":["ACKC10"]}},

"S14":{
  "doc":["Stream 14 Object Services",
    "The functions in this stream are used for generic functions
concerning objects,",
    "including obtaining information about objects and setting values
for an object."],

  "S14F0":{
    "doc":["Abort Transaction",
      "Used in lieu of an expected reply to abort a transaction.",
      "Function 0 is defined in every stream and has the same
meaning in every stream."],
    "block":"single","direction":"h<->e"},

    "S14F9":{
      "doc":["Create Object Request",
        "This message is used to request an object owner to create an
object instance.",
        "OBJSPEC specifies the object owner.",
        "Exception: If OBJSPEC is a null-length item, no object
specifier is provided.",
        "If variable list size = 0, no specific attribute settings are
requested for the new object."],
      "block":"multiple","direction":"h<->e","reply":true,
      "body":["OBJSPEC","OBJTYPE",[["ATTRID","ATTRDATA"],"..."]]},

      "S14F10":{
        "doc":["Create Object Acknowledge",
          "This message is used to acknowledge the success or failure of
creating the new object specified.",
          "If successful, OBJSPEC is the object specifier of the new
object.",
          "The list of attributes returned is dependent upon the type of
object specified.",
          "Exception: If OBJSPEC is a null-length item, no object was
created.",
          "If the attribute list size = 0, then no attributes of the new
```

```

object are returned.",
    "If the errors list size = 0, no errors were detected."],
    "block":"multiple","direction":"h<->e",
    "body":["OBSPEC",
        [["ATTRID","ATTRDATA"],"..."],
        [["OBJACK",[["ERRCODE","ERRTEXT"],"..."]]]]],

    "S16":{
        "doc":["Stream 16 Processing Management",
            "This stream provides protocol for a set of messages that enable
the control of material processing at equipment and equipment resources."],

        "S16F0":{
            "doc":["Abort Transaction",
                "Used in lieu of an expected reply to abort a transaction.",
                "Function 0 is defined in every stream and has the same
meaning in every stream."],
            "block":"single","direction":"h<->e"},

        "S16F1":{
            "doc":["Multi-block Process Job Data Inquire",
                "If any of Processing Management messages are larger than one
block, then this transaction must precede that message."],
            "block":"single","direction":"h->e","reply":true,
            "body":["DATAID","DATALENGTH"]},

        "S16F2":{
            "doc":["Multi-block Process Job Data Grant",
                "Message to indicate if permission is granted to transmit a
multi-block Job Data message."],
            "block":"single","direction":"h<-e",
            "body":["GRANT"]},

        "S16F15":{
            "doc":["PRJobMultiCreate",
                "Use this single message to Create Multiple Process Jobs, each
of which may be unique in its association of material to process recipe.",
                "If multi-block, this message must be preceded by the S16F1/F2
transaction.",
                "Exception: The list for specifying material is empty when no
material is specified for the process job.",
                "The form of data for item 3 depends on the value in MF."],
            "block":"multiple","direction":"h->e","reply":true,
            "body":["DATAID",[["PRJOBID","MF",{ "or":[["CARRIERID",
["SLOTID","..."]],"..."],
["MID","..."]]]],
["PRRECIPEMETHOD","RCPSPEC",
["RCPPARNM","RCPPARVAL"],"..."],"PRPROCESSSTART","PRPAUSEEVENT"]]]],

```

```

    "S16F16":{
        "doc":["PRJobMultiCreate Acknowledge",
            "This message acknowledges the request and reports any errors
in the creation of a process job.",
            "ERRTEXT contains the identifier of process jobs that were not
created.",
            "Exception: If the list of errors is empty then no errors
exist."],
        "block":"single","direction":"h<-e",
        "body":[["PRJOBID","..."],["ACKA",
[["ERRCODE","ERRTEXT"],"..."]]]},

    "S16F17":{
        "doc":["RJobDequeue",
            "Used to remove process jobs from the equipment for jobs that
have not begun processing.",
            "Exception: If the list size = 0, then de-queue all."],
        "block":"single","direction":"h->e","reply":true,
        "body":["PRJOBID","..."]},

    "S16F18":{
        "doc":["PRJobDequeue Acknowledge",
            "Acknowledge the request to de-queue and report any errors.",
            "ERRTEXT will contain the identifier of any jobs that were not
de-queued.",
            "Exception: If error list size = 0, no errors exist."],
        "block":"single","direction":"h<-e",
        "body":[["PRJOBID","..."],["ACKA",
[["ERRCODE","ERRTEXT"],"..."]]]},

    "S16F19":{
        "doc":["PRGetAllJobs",
            "Requests the equipment to return a list of process jobs which
have not completed.",
            "They may be running or waiting to run."],
        "block":"single","direction":"h->e","reply":true},

    "S16F20":{
        "doc":["PRGetAllJobs Send",
            "Returns the requested list of process jobs.",
            "Exception: If the list of jobs = 0, then no process jobs are
running or waiting to run."],
        "block":"single","direction":"h<-e",
        "body":[["PRJOBID","PRSTATE"],"..."]}]
}
}

```

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

SECS/GEM Messages

SECS Message Basics

From the SECS-II standard:

SECS-II defines the method of conveying information between equipment and host in the form of messages. These messages are organized into categories of activities, called streams, which contain specific messages, called functions.

Each SECS message specifies a stream and a function. Specific SECS messages are often referred to in this format: **S(number)F(number)**. For example **S1F13** refers to SECS message Stream 1, Function 13.

Requests and Responses

There are two kinds of messages:

- **Request:** sometimes called the primary message. These originate from the Host, which in the context of Ignition, is the Ignition Gateway. Requests are odd-number functions, such as S1F1.
- **Response:** sometimes known as a secondary message. These originate from the Equipment. Responses are even-number functions, such as S1F2

Thus, the Host can send an S1F1 Request, which establishes if the Equipment is on-line. If so, then the Equipment will respond with an S1F2 Response, which signifies that the Equipment is on-line. Each request can only have a single response message, or no response message if none is needed. This Request-and-Response pattern is the basis of the module, and the standard.

SECS Message Format

Every SECS message is a JSON formatted string, which contains a header. The header contains the stream, function, and if the message expects a response message. Optionally, the SECS message can have a body, which consists of a list of one or more data items. Each data item can also be a list

of other data items. This forms the basis for which all SECS messages should look like, regardless of if they are a request or response.

SECS Message Example

The following is an example of a S1F11 request:

S1F11 Request Message

```
{
  "header":{
    "doc":"Status Variable Namelist Request",
    "stream":1,
    "function":11,
    "reply":true
  },
  "body":[
    {
      "doc":"SVID, Status Variable ID",
      "format":"U4",
      "value":4
    }
  ]
}
```

When making this request from a script, the **sendRequest** function takes in the header values as parameters, with the whole body as another parameter.

Python - Send an S1F11 request to the Equipment

```
body = [{"format":"U4", "value":4}]

transactionID = system.secsgem.sendRequest("S1F11",True,body,"Equip4")
```

A typical response to the above request would look something like this:

S1F12 Response Message

```

{
  "header":{
    "doc":"Status Variable Namelist Reply",
    "stream":1,
    "function":12,
    "reply":false
  },
  "body":[
    [
      {
        "doc":"SVID, Status Variable ID",
        "format":"U4",
        "value":4
      },
      {
        "doc":"SVNAME, Status Variable Name",
        "format":"A",
        "value":"RandomBinaryData"
      },
      {
        "doc":"UNITS, Units Identifier",
        "format":"A",
        "value":"B"
      }
    ]
  ]
}

```

Each response message would be received using the [getResponse](#) function, which would use the transactionID from the sendRequest call above, and then do something with the response such as printing it.

Python - Get an S1F12 response from the Equipment

```

response = system.secsgem.getResponse(transactionID, "Equip4")

print response

```

This simple request/response between the Gateway and Equipment forms the basis for communication using the SECS/GEM module. However, there are some other tools that allow you to receive the response message in different ways: through Tags and Message Handlers.



CAUTION

In order to send a certain function request, the equipment connection must have that function defined in the [SDL File](#).

SECS Messages through Tags

Any SECS messages that contain data can be captured in Tags that can then be used within a project. This can be really useful for something like data coming back from tool traces within the S6F1 messages. When properly configured, the system will automatically create the appropriate Tags within the equipment's folder in the SECSGEM Realtime Tag Provider. While the Tags that get created can be bound to components, they cannot be written to and they can't be edited which means no Tag History, no Alarming, etc. Also note that if the Gateway is ever restarted those Tags will disappear, so they will need to be recreated. To have responses generate Tags, the function must be configured to do so within the SDL File. This is done by adding an object called 'SQLTags' with a value of 'true', along with a 'CommonID' array that holds an identifier. The default SDL file is preconfigured to create Tags for S6F1 messages. Sending a S2F23 request with the appropriate trace parameters to the tool will produced Tags based on the response message. With each new S6F1 message that is received, the Tags will update their values. Using the built in simulator, you can easily create Tag values by calling the S2F23 request with the body below.

Simulator S2F23 Body

Body of an S2F23 request

```
[
  {
    "doc": "TRID, Trace Request ID.",
    "format": "U4",
    "value": 1
  },
  {
    "doc": "DSPEP, Data sample period.",
    "format": "A",
    "value": "000002"
  },
  {
    "doc": "TOTSMP, Total Samples",
    "format": "U4",
    "value": 10
  },
  {
    "doc": "REPGSZ, Reporting Group Size",
    "format": "U4",
    "value": 1
  },
  [
    {
      "doc": "SVID Status Variable ID",
      "value": 1,
      "format": "U4"
    },
    {
      "doc": "SVID Status Variable ID",
      "value": 2,
      "format": "U4"
    },
    {
      "doc": "SVID Status Variable ID",
      "value": 3,
      "format": "U4"
    }
  ]
]
```

An Example with S2F14 using the Simulator

This example sets up the S2F14 message to create Tags that we can see in the Tag provider. It assumes the equipment connection is made to a simulator with the default SDL file. In order to capture other messages in Tags, the SDL File will need to be configured to allow that. Here, we setup the SDL File to have S2F14 to create Tags.

1. Go to the Gateway Webpage Configure section, and navigate to SECSGEM > Equipment.
 - i. On the equipment connection, edit the SDL File.
2. Locate the S2F14 message in the list.
 - i. Add "SQLTags" as an Auto type object with a value of "true" directly under the S2F14 root.
 - ii. Create an array called "CommonID" directly under the S2F14 root.
 - a. Add "0: ECV" to the CommonID array as an Auto type object.
3. Save the SDL file and restart the equipment connection.
4. Once the equipment connection has been reestablished, send it an S2F13 message with the body below:

S2F13 Body

S2F13 Body

```
[
  {
    "doc": "ECID, Equipment Constant ID",
    "format": "U4",
    "value": 200
  },
  {
    "doc": "ECID, Equipment Constant ID",
    "format": "U4",
    "value": 201
  },
  {
    "doc": "ECID, Equipment Constant ID",
    "format": "U4",
    "value": 203
  },
  {
    "doc": "ECID, Equipment Constant ID",
    "format": "U4",
    "value": 205
  },
  {
    "doc": "ECID, Equipment Constant ID",
    "format": "U4",
    "value": 206
  },
  {
    "doc": "ECID, Equipment Constant ID",
    "format": "U4",
    "value": 207
  }
]
```

SECS Messages through Message Handlers

Similar to how SECS message data can be captured in tags, they can instead be made to call a custom Python script. To implement a custom Python script, the message will need to call a **Message Handler** with the name **onSecsGemRealtimeUpdate**. This Message Handler must have that exact name, though a handler can be specified for both the Client Event Scripts and the Gateway Event Scripts. Within the Message Handler, the SECS message will pass in information into the payload object.

Payload Key	Value
CommonID	The common ID of the message.
Equipment	The name of the equipment that the message is coming from.
Direction	The direction of the message.
RequestResponse	If this was from a request or a response.
TxID	The transaction ID of the message.
Reply	If this message expected a reply.
Message	The message contents.

This script can handle the message in any way such as storing data in the database or writing to Tags. The example below will take the information and create a logger object with it that will then be printed to the console.

Python - Example of a onSecsGemRealtimeUpdate Message Handler

This message handler is very simple in that it will create a logger with the SECS message data that prints to the console.

```
commonID = payload['CommonID']
equipment= payload['Equipment']
direction = payload['Direction']
reqResp = payload['RequestResponse']
txId = payload['TxID']
reply = payload['Reply']
message = payload['Message']

msg = "CommonID=" + commonID
msg += ", Equipment=" + equipment
msg += ", Direction=" + direction
msg += ", RequestResponse=" + reqResp
msg += ", TxID=" + str(txId)
msg += ", Reply=" + str(reply)
msg += ", Message=" + message
logger = system.util.getLogger("SECSGEM.Gateway.onSecsGemRealtimeUpdate")
logger.info(msg)
```

Once an onSecsGemRealtimeUpdate Message Handler has been specified, it needs to be specified within the Gateway. Within the Gateway Webpage Configure section, the **Module Settings** page under the SECS/GEM header will have Properties to specify which project holds the Gateway Message Handler and which project holds the Client Message Handler. These can be the same project or different projects.

SECS/GEM Module Settings

Realtime Message Handlers	
Gateway Message Handler Project	myProject Specifies a project that must contain a Gateway message handler called 'onSecsGemRealtimeUpdate'. Incoming realtime value updates will send messages to this handler and include the message information in the handler payload. (default:)
Client Message Handler Project	myProject Specifies a project that must contain a client message handler called 'onSecsGemRealtimeUpdate'. Incoming realtime value updates will send messages to this handler and include the message information in the handler payload. (default:)

An Example with S2F14 using the Simulator

This example sets up the S2F14 message to call the Gateway Message Handler that we setup. It assumes the equipment connection is made to a simulator with the default SDL file. In order to capture message with the onSecsGemRealtimeUpdate, the SDL File will need to be configured to allow that. Here we setup the SDL File to have S2F14 to go through the message handler.

1. Create a Gateway Message Handler called **onSecsGemRealtimeUpdate**.
2. Go to the Gateway Webpage Configure section, and navigate to the SECS/GEM > Module Settings page.
 - i. Specify the project that the Gateway Message Handler was created on in the Gateway Message Handler Project setting.
3. Within the Gateway Webpage Configure section, navigate to SECS/GEM > Equipment.
 - i. On the equipment connection, edit the SDL File.
4. Locate the S2F14 message in the list.
 - i. Add "realtime" as an Auto type object with a value of "true" directly under the S2F14 root.
 - ii. Create an array called "CommonID" directly under the S2F14 root.
 - a. Add "0: ECV" to the CommonID array as an Auto type object.
5. Save the SDL file and restart the equipment connection.
6. Once the equipment connection has been reestablished, send it an S2F13 message with the body below:

S2F13 Body

S2F13 Body

```
[
  {
    "doc": "ECID, Equipment Constant ID",
    "format": "U4",
    "value": 200
  },
  {
    "doc": "ECID, Equipment Constant ID",
    "format": "U4",
    "value": 201
  },
  {
    "doc": "ECID, Equipment Constant ID",
    "format": "U4",
    "value": 203
  },
  {
    "doc": "ECID, Equipment Constant ID",
    "format": "U4",
    "value": 205
  },
  {
    "doc": "ECID, Equipment Constant ID",
    "format": "U4",
    "value": 206
  },
  {
    "doc": "ECID, Equipment Constant ID",
    "format": "U4",
    "value": 207
  }
]
```

Custom SECS Message Response Handlers

For messages that expect a response, the system is setup to automatically send a response based on what is configured in the SDL file. However, a custom response can be created using a Message Response Handler, allowing you to build custom responses completely in the Python scripting language. Only one custom response to a stream function can be configured per equipment connection, but each equipment connection can have multiple custom responses, each to a different

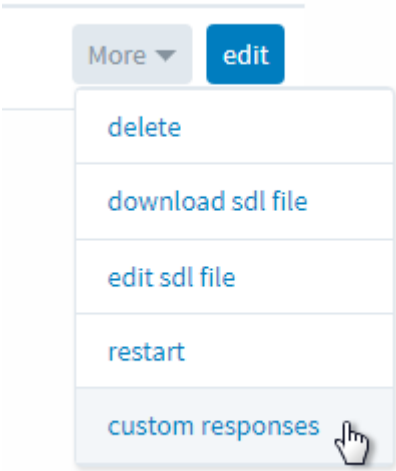
stream function. By setting up a custom Message Response Handler for a stream function, the handler will be used for all responses to the specified stream function.

A Message Response Handler is setup on an equipment connection and pointed at a Gateway Event Script Message Handler. Each Message Handler will have a payload object that will contain information from the default message response.

Payload Key	Value
Equipment	The name of the equipment that the message is coming from.
TxID	The transaction ID of the message.
SystemBytes	The system bytes of the response.
Message	The message contents.

Using the data given to us in the payload, the Message Handler can then manipulate it in any way before calling the `system.secsgem.sendResponse` function, which will send the response back to the equipment.

New in 7.9.15 As of 7.9.15, custom response handlers will execute when receiving messages over the SECS-1 (serial) protocol. Once a Message Handler has been created, the equipment connection needs to specify that it will be using that as a custom message response. To do this, navigate to the Gateway Webpage Configure section, navigate to SECS/GEM > Equipment. For the equipment connection, click the more button on the right and select the Custom Responses option:



On the Message Response Handlers page, we can setup a link between the stream function response we want to intercept and which Message Handler will be used to handle it. Each Message

Response Handler has the following settings.

Setting	Description
Enabled	If true, the specified script message handler will be fired when the specified StreamFunction is encountered.
Intercept Stream Function	The SECS stream and function type of the message that will be intercepted. Use a value such as "S6F2"
Gateway Message Handler	The name of the script message handler that will create the SECS response message. This message handler must be created in a project to provide custom responses, and must be a Gateway Event Script message handler.
Project Name	The project that contains the script message handler specified above.

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

SECS/GEM Simulator

Overview

The SECS/GEM Module includes the capability to add and configure equipment simulators. Equipment simulators include basic functionality and have the ability to handle and respond to a number of SECS messages. You can add status variables, equipment constants, reports and collection events to a simulator, allowing you to model a simulator after an existing tool. You can also add hard-coded responses to messages that do not exist in the simulator. These functions allow the simulator to act as a stand-in for tools that do not provide any type of emulator, or only provide one at great cost.

Simulator SDL File

A simulator is driven from data in its SDL file. This file is the same format as the SDL file used for equipment connections, but it also defines variables, reports, collection events, and hard-coded responses. The simulator SDL file is read when a simulator is started.

You can [modify the simulator SDL file](#) within the Gateway. Navigate to Configuration area and click on the SECS/GEM Simulator link. Click on the "sim variables" link next to the simulator you want to modify. The SDL file will be displayed in a tree, allowing you to browse its contents. You can change individual values on an object in the tree. Click the "Save SDL File" button at the top to save any changes you made. You can also click the "Edit Raw SDL File" link to edit the text of SDL file directly. This can be handy when you need to change several objects at once, or see a full view of the SDL file. Click the "Save SDL File" button at the top to save any changes you made. Don't forget to restart the simulator and its equipment connection after you made changes to the SDL file.

Simulator SDL Utilities

Modifying the simulator SDL file works well for small adjustments, but attempting to create variables, reports and collection events from scratch directly within the SDL file is extremely painful. To eliminate this pain, the Simulator Variables page has a number of [utilities that can automate the creation of these objects](#). Note that after these objects are created, you will still need to modify the simulator SDL file directly to modify or delete the objects.

Default Messages

By default, the Simulator can respond to the following requests.

- **S1F1** - Are You Online
- **S1F3** - Selected Equipment Status Request
- **S1F11** - Status Variable Namelist Request
- **S1F13** - Establish Communications Request
- **S1F15** - Request OFF-LINE
- **S1F17** - Request ON-LINE
- **S2F13** - Equipment Constant Request
- **S2F15** - New Equipment Constant Send
- **S2F17** - Data and Time Request
- **S2F23** - Trace Initialize Send
- **S2F29** - Equipment Constant Namelist Request
- **S2F31** - Date and Time Set Request
- **S2F33** - Define Report

While the SDL can be modified, note that the Simulator can only respond to the requests listed above: you can add new messages to the simulator's SDL, like S999F1 and S999F2, but the Simulator by default will not respond to any requests using S999F1. If you wish to add new responses, you can create an [Echo Response](#) to simulate a response to any request not listed above. See the [Echo Response](#) section for more details.

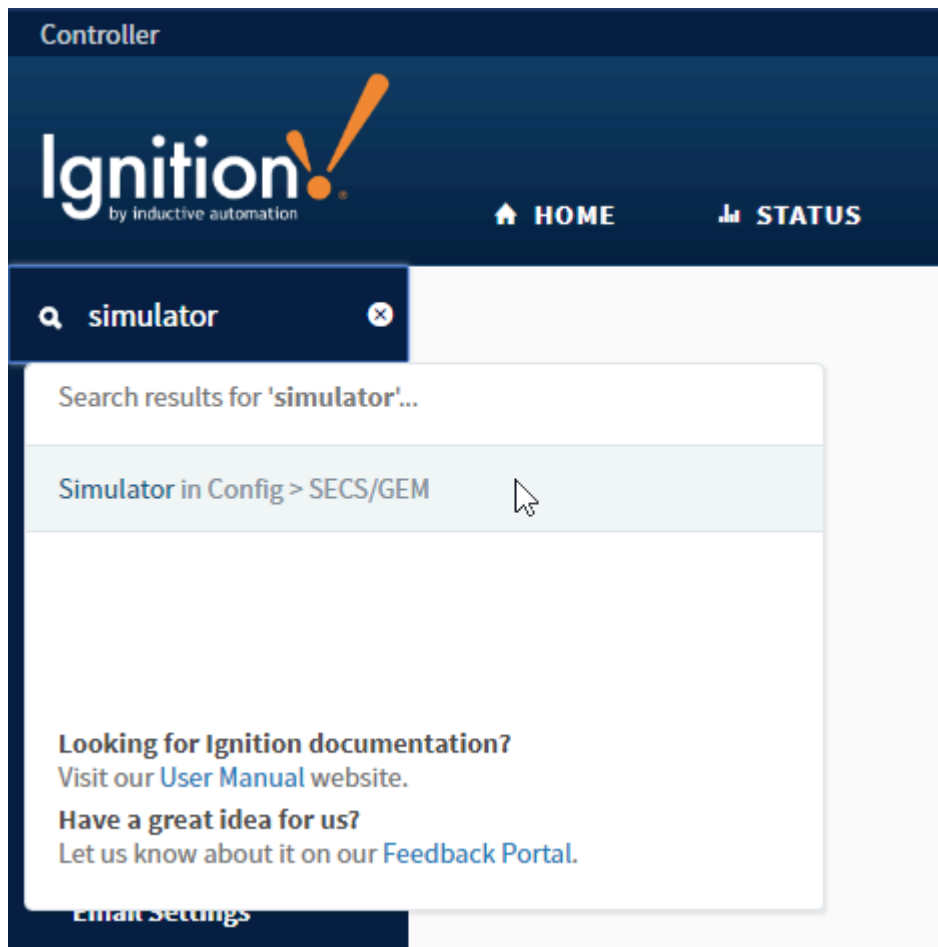
The simulator implements the Communications State Model and the Control State Model from the GEM standard (SEMI E30). For example, sending an S1F15 message to a simulator will cause the simulator to move to the OFF-LINE state and will thereafter respond to most messages with S1F0. Sending an S1F17 message to the simulator will move it back into an online state. When control states change, an S6F11 report message is sent with the updated control state. The default startup control states are defined in the simulator SDL file.

Configuring a Simulator

The following section details how to configure a SECS/GEM simulator.

1. Navigate to the SECS/GEM Simulator page:
 - From the **Configure** section of the Gateway webpage, scroll down the list of links on the left until you see the **SECS/GEM** section. click on the **Simulator** link.

- Alternatively, use the search bar located in the upper left corner of any page on the Gateway, and search for `simulator`.



2. Once on the Simulators page, click on the **Create New Simulator** link.
3. From here we can create a new simulator. In the very least, you will need to specify a name for the simulator.
4. If this is the first SECS/GEM simulator device connection, then you can use the default values for the rest of the properties and skip to the next step.

Subsequent simulators require a different port: two simulators cannot run on the same port. Note that there are multiple port properties.

- If the **Connection Mode** is set to Active, then the simulator will use the **Active Port**.
 - If the **Connection Mode** is set to Passive, then the simulator will use the **Passive Port**.
 - If the **Connection Mode** is set to Alternating, then the simulator will try to use either the **Active Port** or **Passive Port**. As the name implies, the connection will attempt to alternate between the Active and Passive settings until a connection is established.
5. Click on the **Create New Simulator** button to save your configuration and create the simulator device. You will see the Simulators page again.

6. The simulator status will toggle between "Connecting..." and "Not Connected" until you create an equipment connection that connects to the simulator's IP address and port. Once the Equipment Connection is established, the simulator will report a status of "Communicating". You can then run SECS/GEM commands against the simulator's equipment connection and receive responses.

Simulator Properties

When configuring a simulator, there are a number of properties that can be configured.

Property Name	Description	Default
Simulator Name	Name of the simulator.	
Simulator Description	A description of the simulator.	
Enabled	Whether the simulator is enabled or disabled. A disabled simulator will prevent communication with the Equipment Connection.	TRUE
Active IP Address	IP Address of Equipment Connection to connect to. Used when simulator is in Active mode.	localhost
Active Port	The Port number of the Equipment Connection to connect to when simulator is in Active mode. When using multiple simulators, the port numbers need to be unique.	5000
Passive IP Address	IP Address of Ignition that the Equipment Connection will connect to when the simulator is in Passive mode.	localhost
Passive Port	Port number that the Equipment Connection will connect to when the simulator is in Passive mode. When using multiple simulators, the port numbers need to be unique.	5000
Device ID	Unique identifier of equipment. Must be an integer.	0
Connection Mode	Method used to connect. • ACTIVE- Will attempt to connect to the equipment at the given Active IP Address and Active Port. • PASSIVE - Will listen for a connection from the equipment at the given Passive IP Address and Passive Port. • ALTERNATING - Will switch between Active mode and Passive mode until a connection is made.	PAS

Advanced Properties

Property Name	Description	Default
T3 Reply Timeout	Specifies the seconds that the simulator will wait for an expected SECS message reply.	45
T5 Connect Separation Timeout	Specifies the seconds which must elapse between successive attempts to connect to an Equipment Connection after disconnection.	10
T6 Control Transaction Timeout	Specifies the seconds which a control transaction (such as LinkTest or Select) may remain open before it is considered a communications failure.	5
T7 Not Selected Timeout	Seconds which a TCP/IP connection can remain in NOT SELECTED state (i.e., no HSMS activity) before it is considered a communications failure. This timeout is only used in Passive mode.	10
T8 Network Intercharacter Timeout	Maximum seconds between successive bytes of a single HSMS message which may expire before it is considered a communications failure. This applies to HSMS messages received from an Equipment Connection.	5

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Additional SECS/GEM Simulator Features

Default Variables

Each simulator is initialized with a number of status variables and equipment constants for demonstration purposes. The status variables and equipment constants are "live" objects within the simulator. This means that the standard SECS/GEM commands (such as S1F3) can be run to view the variable definitions and current values. Equipment constant values can also be changed using the S2F15 command. See the "Simulator SDL file" section below for information on how to change the values directly in the simulator.

Import Status Variable Definitions

If you can capture status variable definitions from an existing tool using the S1F11 command, you can automatically create status variables within the simulator. Save your capture to a .txt file with the following format (substitute your own status variables):

```

{
  "body":[
    [
      {
        "doc":"SVID, Status Variable ID",
        "format":"U4",
        "value":268435456
      },
      {
        "doc":"SVNAME, Status Variable Name",
        "format":"A",
        "value":"LMMode"
      },
      {
        "doc":"UNITS, Units Identifier",
        "format":"A",
        "value":""
      }
    ],
    [
      {
        "doc":"SVID, Status Variable ID",
        "format":"U4",
        "value":268435457
      },
      {
        "doc":"SVNAME, Status Variable Name",
        "format":"A",
        "value":"LMState"
      },
      {
        "doc":"UNITS, Units Identifier",
        "format":"A",
        "value":""
      }
    ]
  ]
}

```

On the Simulator Variables page, click on the "Import status variable definitions" link. Choose your text file and click "Begin Import". If there were no errors during import, you will be returned to the Simulator Variables page. You can see your new status variables in the tree under data > Status Variables or by viewing the raw SDL file. Note that all imported status variables have a format of "UNKNOWN" and a value of "UNKNOWN", due to the fact that the S1F11 command does not send status variable format or value. You must manually set these values before the status variables can be read by the simulator. After the format and value of the new status variables are set, you can save the simulator SDL file,

and restart the simulator and its equipment connection. The new status variables will now be readable within the simulator.

Import Equipment Constant Definitions

If you can capture equipment constant definitions from an existing tool using the S2F29 command, you can automatically create equipment constants within the simulator. Save your capture to a .txt file with the following format (substitute your own equipment constants):

```

{
  "body":[
    [
      {
        "doc":"SVID, Status Variable ID",
        "format":"U4",
        "value":268435456
      },
      {
        "doc":"SVNAME, Status Variable Name",
        "format":"A",
        "value":"LMMode"
      },
      {
        "doc":"UNITS, Units Identifier",
        "format":"A",
        "value":""
      }
    ],
    [
      {
        "doc":"SVID, Status Variable ID",
        "format":"U4",
        "value":268435457
      },
      {
        "doc":"SVNAME, Status Variable Name",
        "format":"A",
        "value":"LMState"
      },
      {
        "doc":"UNITS, Units Identifier",
        "format":"A",
        "value":""
      }
    ]
  ]
}

```

On the Simulator Variables page, click on the "Import equipment constant definitions" link. Choose your text file and click "Begin Import". If there were no errors during import, you will be returned to the Simulator Variables page. You can see your new equipment constants in the tree under data -> Equipment Constants or by viewing the raw SDL file. Restart the simulator and its equipment connection. The new equipment constants will now be readable within the simulator.

Echo Responses

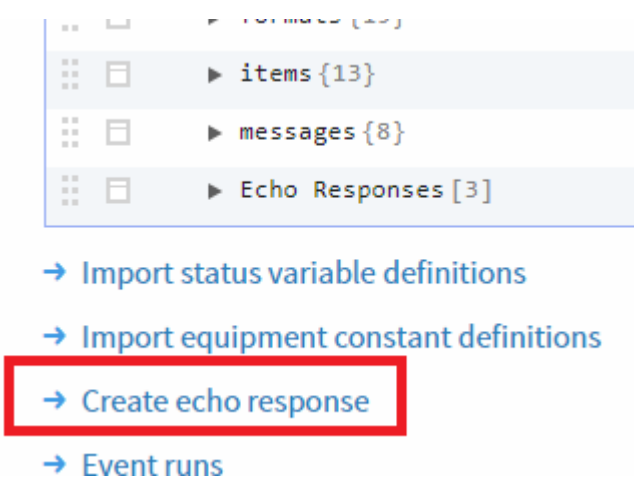
There are a number of SECS/GEM messages that are not implemented in the default simulator. However, you can still use this utility to provide a hard-coded response to these messages.

When creating an Echo Response, you are simply creating a dedicated response to a specified request. Echo Responses can be used to override the responses contained in the Simulator's SDL.

Additionally, you can create an Echo Response to a message that isn't included in the Simulator's Default Messages, allowing you to have the Simulator emulate another tool. In these cases, it you will need to capture the output of an actual tool's response to a message or know the exact format of the response. Furthermore, you will need to modify the Simulator's SDL file to include the message both the request and response.

Creating an Echo Response

On the Simulator Variables page, click on the **Create echo response** link.



The following properties will appear:

Property Name	Description
Name	The name of the response. Used purely to help you identify specific responses when viewing the responses from the SDL.
Stream and Function	The Stream and Function that this Echo will respond to.
Echo Response	The message to return. If emulating another tool, you will want to modify the Echo Response to match an actual response from the tool.

Example

1. If you wanted to create a custom response to S1F1, specify Stream 1, Function 1, and use the Message Below:

```
{
  "header":{
    "stream":1,
    "function":2,
    "reply":false,
    "doc":"On Line Data"
  },
  "body":[
    {
      "format":"A",
      "value":"SimOne",
      "doc":"MDLN, Equipment Model Type"
    },
    {
      "format":"A",
      "value":"1",
      "doc":"SOFTREV, Software Revision Code"
    }
  ]
}
```

2. Click **Save**.
3. Click the **Save SDL File** button.
4. Restart the Simulator.
5. Now, whenever the simulator receives an S1S1 message, it will response with your custom S1F2 message.

NOTE

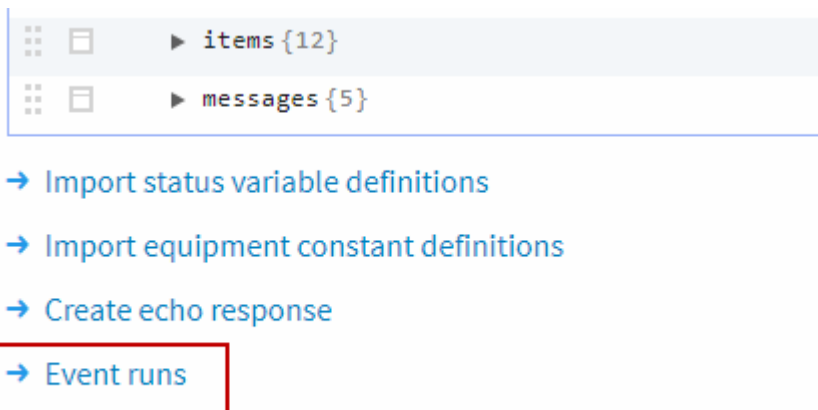
If you added a new function, you may also need to define the message in the messages section of both the simulator SDL file and the equipment connection SDL file. You may also need to add new format items to the items section of both SDL files. Failure to do so can result in "message not found" or "format not found" errors when you attempt to send the command to the simulator.

Event Runs

The Simulator features an Event Run feature, which allows you to create a list of S6F11 messages at fixed intervals that simulate report generation. An Event Run consists of one or more events, with each event containing a S6F11 message that will be sent to the simulator when the event fires.

Add Event Runs

Each simulator can have multiple event runs specified, each with a unique set of events. To access a simulator's event runs, go to the Gateway Webpage Configure section and navigate to the SECS/GEM > Simulator page, and in the **More** menu to the right of the simulator select the **sim variables** option. At the bottom of the SDL editor tool, there will be a link for **Event Runs** that will take you to the page where new event runs can be created.



On the Event Runs page, a new event run can be created by clicking the **Create new Event Run** link, where you can specify the name of the new event run.

Add Events

Once an event run has been made, new events can be added to it that will execute in order from when the event was started. An event run can have any number of events within it. Events must be from S6F11 messages that have been previously captured, which can be copied from the Messages table in the database. To add a new event, navigate to the Event Runs page and click on the **More** menu to the right of the event run that you wish to add events to. Select the events option to navigate to the Events page where you can click on the **Create new Event** link to create a new event in that event run. Each event has a few different properties that can be configured.

Property Name	Description
Event Name	The name of the event.
Index	The index of the event. Note, each event must have a different index in the event run.
Delay	The delay in seconds from the last event until this event fires.
Send Data	A JSON formatted message that will be sent to the equipment connection when the event fires. The message must include header and body fields.

Below is an example of a message for the Send Data property.

JSON - A Sample Send Data Message for an Event Run

```

{
  "header":{
    "stream":6,
    "function":11,
    "reply":true,
    "doc":"Event Report Send"
  },
  "body":[
    {
      "format":"U4",
      "value":2,
      "doc":"DATAID, Data ID"
    },
    {
      "format":"U4",
      "value":3,
      "doc":"CEID, Collected Event ID"
    },
    [
      [
        {
          "format":"U4",
          "value":1,
          "doc":"RPTID, Report ID"
        },
        [
          {
            "format":"A",
            "value":"150408103706",
            "doc":"V, Variable data"
          },
          {
            "format":"A",
            "value":"OFF-LINE/HOST OFF-LINE",
            "doc":"V, Variable data"
          }
        ]
      ]
    ]
  ]
}

```

Using Event Runs

To use an event run, it needs to be started. To start an event run navigate to the Event Runs page, and in the **More** menu to the right of the event run you wish to start select the **start** option. If at any

time you wish to **cancel** the event run, simply select the cancel option from the **More** menu.

Import Events

Instead of manually entering in events, multiple events can be imported from previously captured messages in the Messages table. Simply click on the Import Events link on the Events page. This will allow you to set up an import with various parameters.

Import Parameter	Description
Datasource	The datasource to pull events from.
Equipment Name	The name of the equipment connection that the events are coming from.
Equipment Table Prefix	The prefix of the specified equipment's database tables.
Range Start	The starting date and time to import events from.
Range End	The ending date and time to import events from.

Once the import parameters have been set up, clicking Begin Import at the bottom will start the import process, which will pull any possible events based on the specified parameters.

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Basic SECS/GEM Troubleshooting

Equipment Connections

On Equipment Connection Startup Database Tables Cannot be Created

If you are using MySQL, make sure the latest MySQL JDBC driver is installed in Ignition. If not, you can download it from the MySQL website at <http://dev.mysql.com/downloads/connector/j/>, or from the **Configure > Databases > Drivers** section of the Gateway webpage.

Unable to Establish Equipment Connection at Setup

If you are unable to create an equipment connection after configuring the connection settings, check the Active and Passive Port numbers. Verify that these port numbers are not being used by another piece of equipment or simulator. Everytime you configure another piece of equipment, the Active and Passive Ports numbers must be unique. They cannot be the same as another piece of existing equipment.

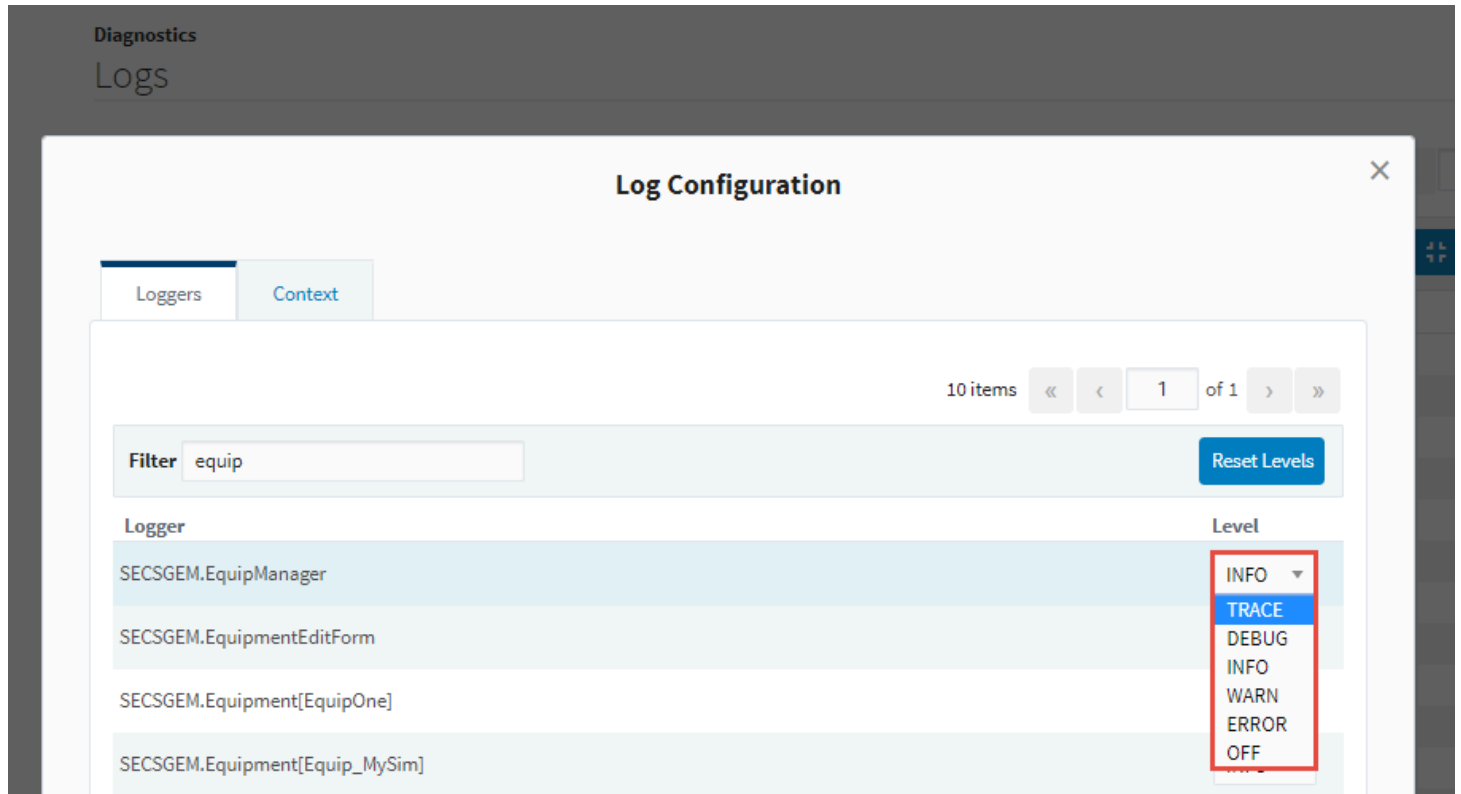
To check the port numbers, go to the **Configure > SECS/GEM Equipment** section of the Gateway webpage. Locate your equipment name, and open the **Equipment Connection** page. Verify that the Active and Passive Port numbers are not being used by another piece of equipment. If so, change the Active and Passive Ports to a unique number, then save your changes. The Equipment Connections page will update and your equipment will have a status of "Communicating."

Logging

Each Equipment Connection has a logger associated with it. The Equipment Connection loggers and their severity level can be seen in the **Status** section of the Gateway webpage under **Diagnostics > Logs**. By default, only messages with a severity level of INFO or more severe such as WARNING or ERROR are logged. A lot of data about what an Equipment Connection is doing can be logged by

changing the log level of the Equipment Connection to a less severe level. At the TRACE log level, all SECS messages sent and received by an Equipment Connection are logged.

To change the severity level of a logger, click the **Settings**  button to open the Log Configuration window. Locate the logger you want change, and click on the dropdown arrow on the right side of the window, and select the severity you want to update.



Wrapper.log File

Messages and log events can be viewed from Ignition's log file called "**wrapper.log**" as well as the Status section of the Gateway webpage under **Diagnostic > Logs**. If something is not working as expected, it can be very useful to turn on ALL logging for an Equipment Connection and view the wrapper.log file. Often an exception error or log message will explain what is happening.

The file path location of the wrapper.log file is different for each operating system:

- **Windows** - `Program Files\Inductive Automation\Ignition\logs`
- **Linux** - `/var/log/ignition`
- **Mac OS X** - `/user/local/ignition/logs`

When logging data on Linux it can be useful to run the command `tail -f /var/log/ignition/wrapper.log`. This command displays current log messages on the screen as they occur. A similar tail program can be used on Windows.

Failure to Send a SECS Message

All SECS messages that are sent and received are validated against the **SECS Definition Language (SDL) file**. A common reason for failure to send and receive a SECS message is it is either not defined or doesn't match a definition in the SDL file. Any messages that fail to validate against the SDL file will be inserted into the Errors database table for the equipment connection as a validation error along with information about why it didn't validate. If either a sent or received message occurs, a log message describing the problem is written to the wrapper.log file, as well as displayed in the Diagnostic Logs for the equipment connection.

To define or update a SECS message, the SDL file can be edited in the Gateway Webpage from the Equipment Connections page using the built in SDL editor, or downloaded from the Equipment Connections page so it can be edited in a text editor and re-uploaded.