Ignition Platform

Overview

The Ignition platform refers to the core installation of Ignition. It provides all the basic functionalities such as connecting to devices and databases, licensing, launching clients and sessions, and managing all Ignition modules.

From that, additional modules or projects can be built to suit any customer's needs. Being cross-platform means that a piece of software like Ignition can be installed on multiple types of operating system such as Windows, Linux, and OSX. With its modular architecture, the Ignition platform enables you to build a customized system using modules as the building blocks. Modules are discussed in detail in the Ignition Modules section of the User Manual.

The following are key elements to the Ignition Platform. Follow the links for more detailed information on each area.

Module	Description
Gateway	The Ignition Gateway is a web server, and when it is running, you access it from a web browser. In your web browser, type http://localhost:8088 to display the Gateway homepage. The Gateway Webpage is the hub for accessing all the functions of the Gateway server.
Database Connections	Connecting Ignition with an SQL database dramatically increases functionality. With a database, you can implement logging of data for historical analysis, tap into your data to discover trends and performance, create charts and reports, store Tag data, and store alarm logs.
Security	Security options in Ignition provide many ways to safeguard your data and applications. You control not only who accesses your systems, but when and where they can access them.
Designer	In Ignition Designer, your user interface design work is done. You create user interfaces with Vision components or Perspective components. Tags associated with components can instantly bind data to tables, charts, and graphs.
Tags	Tags are points of data and may have static values or dynamic values that come from an OPC address, an expression, or a SQL query. Tags offer a great amount of power in system design and configuration.
Alarming	Alarming enables you to easily create alarms, store alarm history, and design and manage your alarm notifications.
Localization and Languages	With Localization you can translate text into multiple languages in a project for display on client screens. The localization feature allows users located in different countries to set their default language so client screens can be displayed in their native language.
Expression Overview and Syntax	The expression language is used to define dynamic values for component properties and expression Tags. Expressions often involve one or more other values that are used to calculate a final value. In most cases, expressions only return a value.
SQL in Ignition	Ignition's ability to connect to databases greatly increases the functionality available to you. You can use databases to store history, create easy to search lists and configurations, and retrieve data from ERP or other systems.
Scripting	Most of the time when we talk about "scripting" in Ignition we are talking about Python scripting, or writing code in the Python language. Python is a general purpose programming language that was developed in the early 90s and has gained significant popularity in the 2000s. We like it because it is extremely readable, elegant, powerful, and easy to learn. As an added bonus, it gracefully interacts with Java, giving programmers an extremely powerful tool when paired with Ignition, which is written in Java.

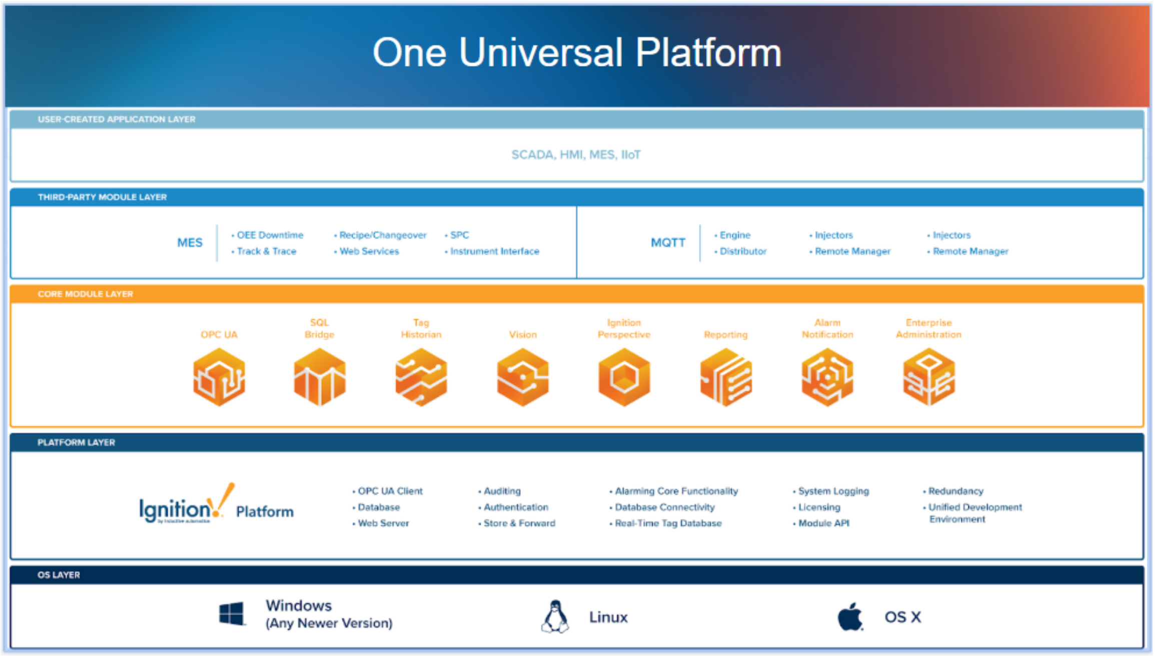
Modular Architecture and Software Stack

Ignition platform has a modular architecture. Modules are software applications that are built and integrated into the platform to offer additional functionality. The modules are similar to applications for a smartphone in how they are seamlessly integrated and provide additional capabilities.

Most of the main features of Ignition are actually provided by different modules such as the Perspective, Vision, and SQL Bridge modules.

The Ignition software stack is shown in the illustration below. You can see that the HMI/SCADA/MES module layers are built on the Ignition platform. Here are the different software layers in Ignition's modular architecture:

- **Operating System (OS) Layer**
  Provides basic computing resources such as the file system and access to the network.
- **Platform Layer**
  Provides all the basic functionalities such as connecting to devices and databases, licensing, launching clients, and managing all Ignition modules over the web.
- **HMI/SCADA Module Layer**
  Provides the core modules that enable real-time and historical data access, trends, and control.
- **Third-Party Module Layer**
  Additional modules provided by Strategic Partners and other developers to further extend Ignition's capabilities.
- **User Created Application Layer**
  The resulting project created for your organization. Developed internally, or by a third-party.



In This Section ...

Gateway

The Ignition Gateway is the primary software service that drives everything in Ignition. It is a single application that runs as a web server and accessed through a web browser. It connects to data and PLCs, executes modules, and communicates with clients. You can customize the Homepage to fit your needs using the Gateway Settings. From the Gateway, you activate Ignition, transfer licenses, backup and restore the Ignition Gateway, and setup redundancy. You can even set up a Gateway Network that allows two or more Gateways to connect to one another and share data.

On this page ...

Launching a Gateway

The Gateway is accessed through a web browser. The web browser, running on any machine, must have network access to the host that is running the Gateway.

By default, Ignition installs by using the **`8088`** port. For example, if the host's IP address is `10.0.28.30`, you access the Gateway via the URL: `http://10.0.28.30:8088` . When Ignition is installed on the computer that you are logged into, you can access it by typing `http://localhost:8088` to display the Gateway Homepage. Any other computer on the same network can access the Gateway by using the IP Address or Host Name of the computer where Ignition is installed: i.e., `http://192.#.#.#:8088`, and it will launch the Gateway and bring up the Gateway Homepage.

Starting and Stopping the Gateway

After installation, the Gateway starts automatically. The Gateway runs as a service, so you can use your operating system's normal mechanisms to start or stop the service. You can also start or stop the Gateway using the Gateway Command-line Utility.

Gateway Webpage

The Gateway Webpage performs a cadre of functions and is the hub for accessing all the functions of the Gateway server. It drives everything in Ignition! The Gateway Webpage is where you setup your licensing and activation, configure your Gateway settings, databases, devices, projects, modules, security, and alarming. When the Gateway server is running, you can connect to a device, connect to a database, launch the Designer, and launch a Vision client or Perspective session. You can also check the status of your system, network, agents, sessions, tasks, reports, and alarms.

The Gateway Webpage has three tabs on the left side of the page that lead you to the key sections of the server: **Home** , **Status** , and **Config.** The top of the page shows you the path of where you are on the Gateway Webpage. You can perform a host of Gateway functions from configuring your system and modules, checking the status of all your Gateway connections, to launching clients and sessions, and many tasks in between. You can even redirect the Gateway Homepage to display another URL.

The Gateway Webpage

Watch the Video

Home

The first time you go to the Gateway Homepage, it shows you several steps to help you get started. Once you're up and running, the **Home** tab lets you open Vision clients, Perspective sessions, and download the Designer Launcher. There are some resource links to help you get started with Ignition quickly: Inductive University, Quick Start Guide, Production Documentation where you can learn about Ignition modules, and the Appendix which contains a complete reference for components, expressions and scripting functions.

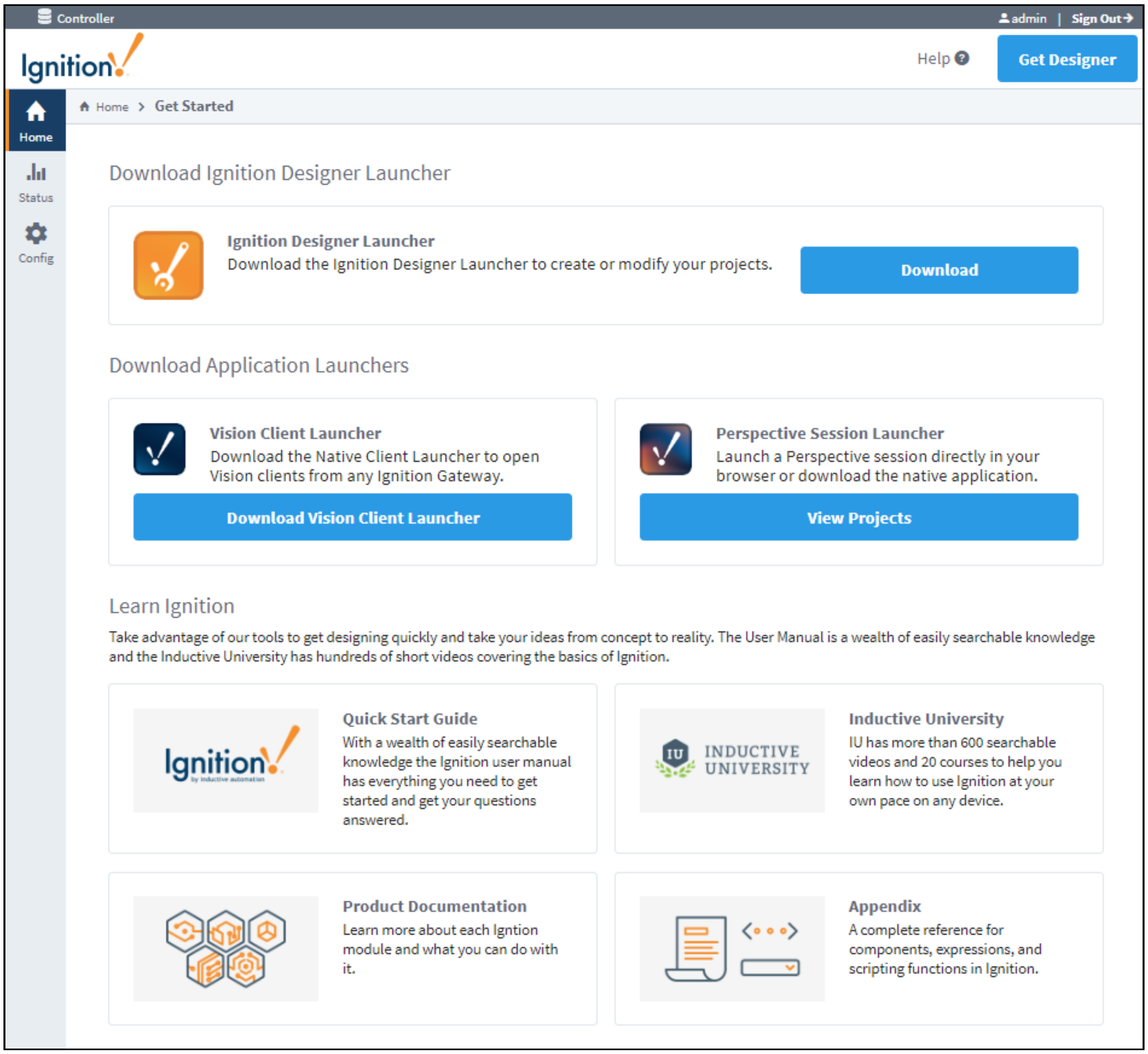
When you're first introduced to Ignition, the Homepage is the landing page. It's where you'll find the Designer Launcher, Vision Client Launcher, and Perspective Session Launcher.

- The Designer Launcher locates all Gateways that are available on your local network. Once you open the Designer, you can access existing projects or create new projects on the Gateway.

Customizing the Gateway Homepage

Watch the Video

- The Vision Client Launcher opens Vision Clients from any Ignition Gateway. It browses all Gateways for Vision projects that are available on your local network and remote locations. Once your Vision projects are added to the Vision Client Launcher, they will be displayed and all you have to do is click the link to launch a Vision Client.
- The Perspective Sessions Launcher opens a session directly in your browser or you can download the native application.



Download Ignition Designer Launcher

Ignition Designer Launcher
Download the Ignition Designer Launcher to create or modify your projects.

Download Application Launchers

Vision Client Launcher
Download the Native Client Launcher to open Vision clients from any Ignition Gateway.

Perspective Session Launcher
Launch a Perspective session directly in your browser or download the native application.

Learn Ignition

Take advantage of our tools to get designing quickly and take your ideas from concept to reality. The User Manual is a wealth of easily searchable knowledge and the Inductive University has hundreds of short videos covering the basics of Ignition.

Quick Start Guide
With a wealth of easily searchable knowledge the Ignition user manual has everything you need to get started and get your questions answered.

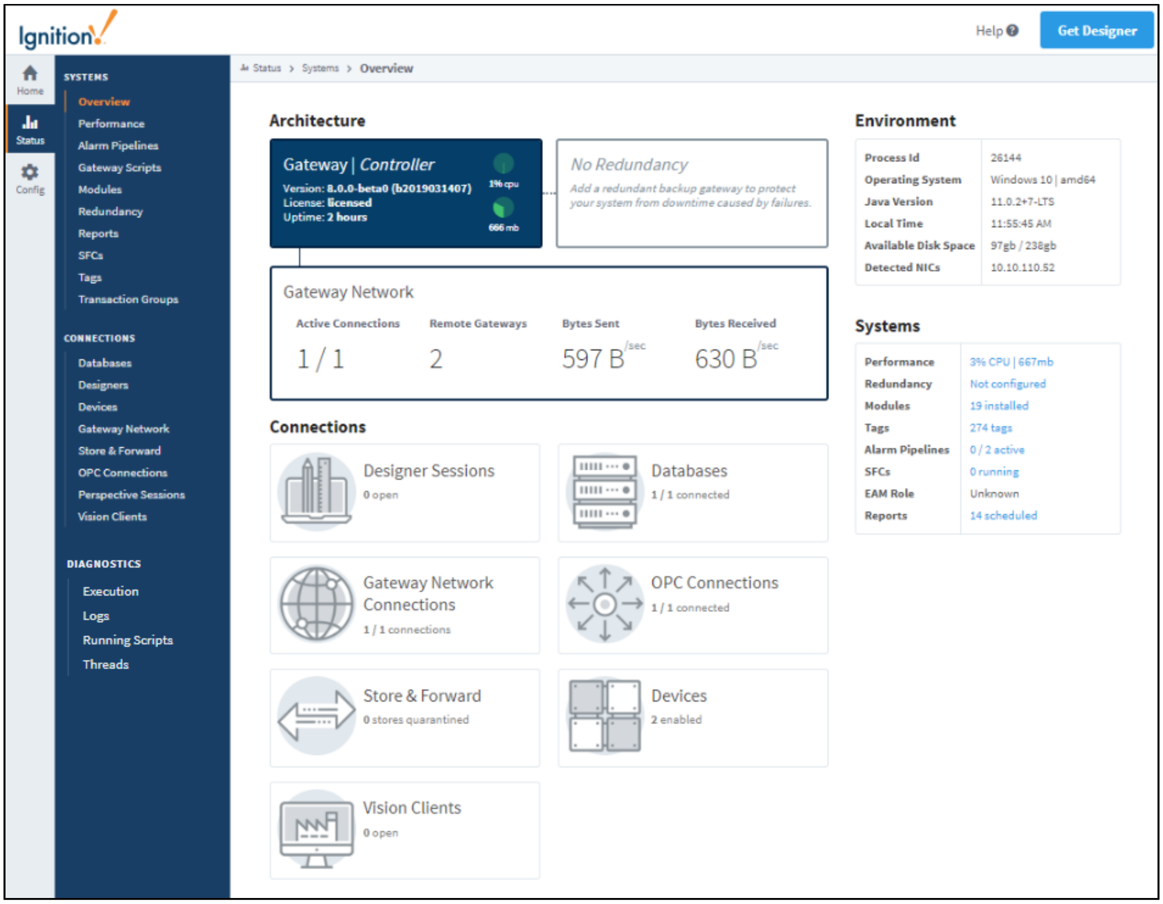
Inductive University
IU has more than 600 searchable videos and 20 courses to help you learn how to use Ignition at your own pace on any device.

Product Documentation
Learn more about each Igntion module and what you can do with it.

Appendix
A complete reference for components, expressions, and scripting functions in Ignition.

Status

The **Status** page provides in-depth information about the status of the different parts of the Ignition system. The list of options on the left menu in the Status page changes based on what modules are installed. You can select any of the available options to get more detailed information.



Architecture

Gateway | *Controller*
Version: 8.0.0-beta0 (b2019031407)
License: licensed
Uptime: 2 hours
1% cpu
666 mb

No Redundancy
Add a redundant backup gateway to protect your system from downtime caused by failures.

Gateway Network

Active Connections	Remote Gateways	Bytes Sent	Bytes Received
1 / 1	2	597 B /sec	630 B /sec

Connections

Designer Sessions
0 open

Databases
1 / 1 connected

Gateway Network Connections
1 / 1 connections

OPC Connections
1 / 1 connected

Store & Forward
0 stores quarantined

Devices
2 enabled

Vision Clients
0 open

Environment

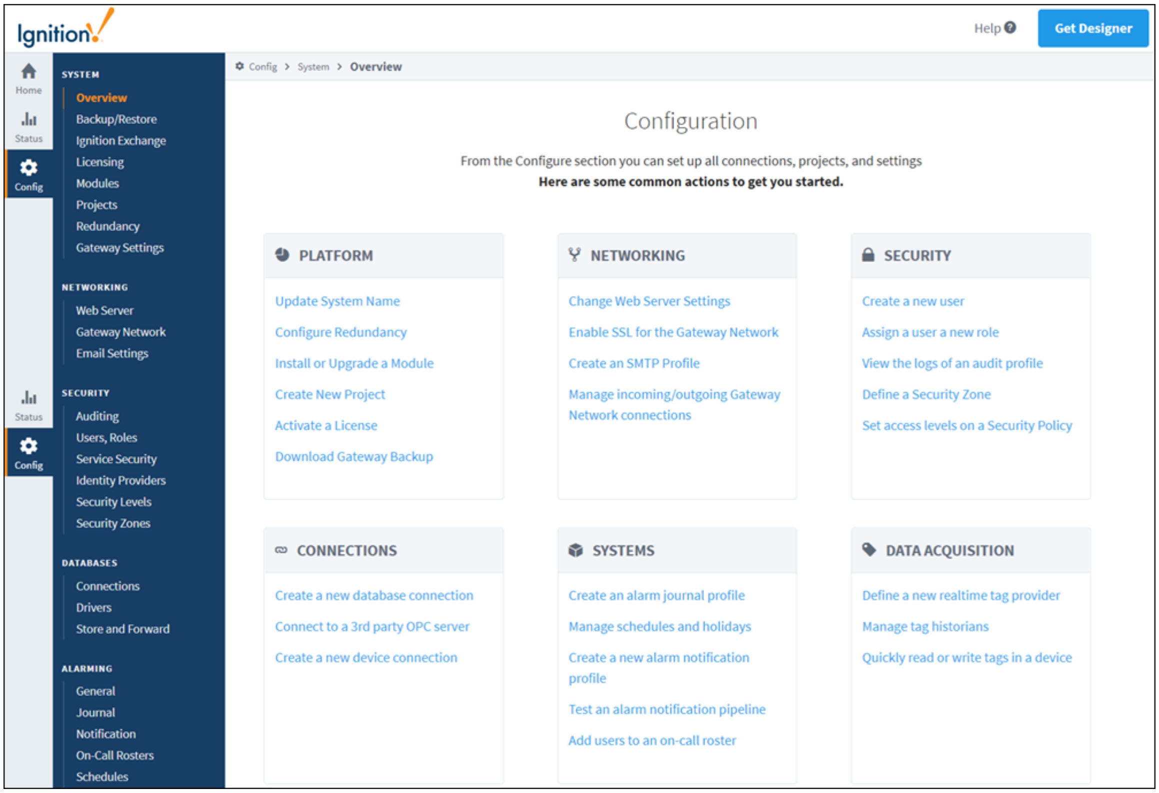
Process Id	26144
Operating System	Windows 10 | amd64
Java Version	11.0.2+7-LTS
Local Time	11:55:45 AM
Available Disk Space	97gb / 238gb
Detected NICs	10.10.110.52

Systems

Performance	3% CPU | 667mb
Redundancy	Not configured
Modules	19 installed
Tags	274 tags
Alarm Pipelines	0 / 2 active
SFCs	0 running
EAM Role	Unknown
Reports	14 scheduled

Config

If you are not already logged into your Gateway, **Config** tab presents you with a login screen. Enter the **username** and **password** for the Ignition administrator. This was the first account created during installation. From the Config page, you can setup all connections, projects, and perform all the Gateway and platform operations. The list of options on the left menu changes based on what modules are installed. You can select any of the available options to get more detailed information. This page also provides some links to common actions to help get you get started.



Configuration

From the Configure section you can set up all connections, projects, and settings

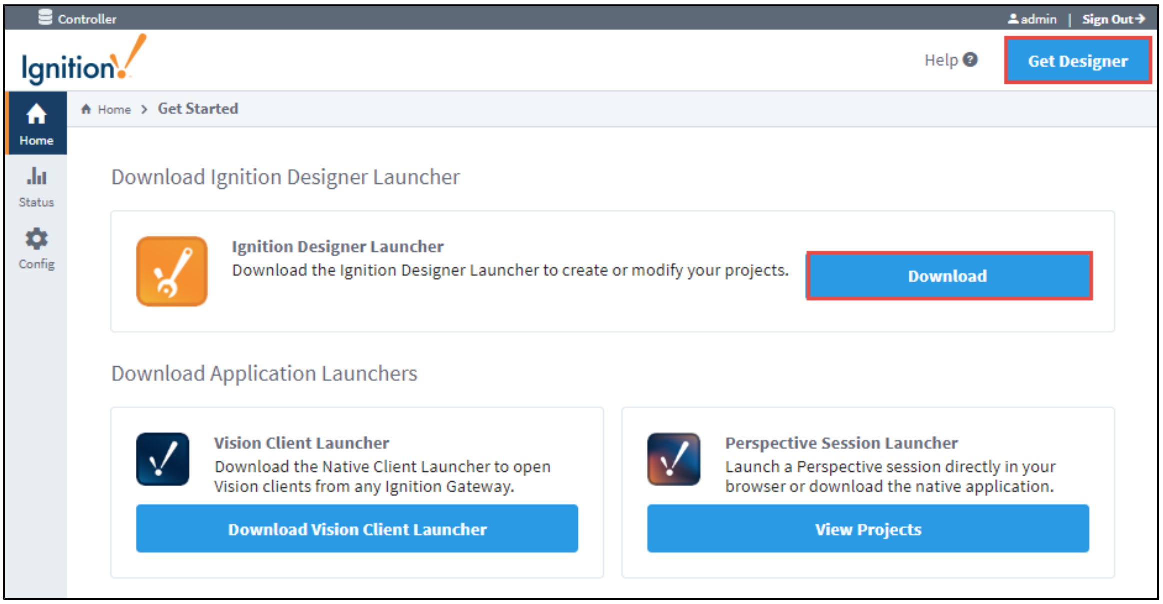
Here are some common actions to get you started.

PLATFORM
Update System Name
Configure Redundancy
Install or Upgrade a Module
Create New Project
Activate a License
Download Gateway Backup
NETWORKING
Change Web Server Settings
Enable SSL for the Gateway Network
Create an SMTP Profile
Manage incoming/outgoing Gateway Network connections
SECURITY
Create a new user
Assign a user a new role
View the logs of an audit profile
Define a Security Zone
Set access levels on a Security Policy
CONNECTIONS
Create a new database connection
Connect to a 3rd party OPC server
Create a new device connection
SYSTEMS
Create an alarm journal profile
Manage schedules and holidays
Create a new alarm notification profile
Test an alarm notification pipeline
Add users to an on-call roster
DATA ACQUISITION
Define a new realtime tag provider
Manage tag historians
Quickly read or write tags in a device

Designer Launcher

The Designer Launcher locates all the Gateways that are available on your local network. Once the Designer is open, you can access existing projects or create new projects on the Gateway.

The **Designer Launcher** button on the top right side of the page and the **Download** button opens the Designer Launcher Download page. Here you can find all the Designer Launchers for each operating system: Windows, Mac and Linux. Ignition automatically detects your operating system so all you have to do is download the launcher and follow the steps to install the Designer Launcher. To learn about what the Designer can do, go to the Designer section of this manual.



Download Ignition Designer Launcher

Ignition Designer Launcher

Download the Ignition Designer Launcher to create or modify your projects.

Download Application Launchers

Vision Client Launcher

Download the Native Client Launcher to open Vision clients from any Ignition Gateway.

Perspective Session Launcher

Launch a Perspective session directly in your browser or download the native application.

Gateway Command-line Utility

The Gateway Command-line Utility (GCU) is a lightweight standalone application that provides information about the Gateway. It performs high-level tasks that aren't available inside the Gateway webpage such as stopping and restarting the Gateway server, and setting ports used between the Gateway and clients. It shows the status of the Tomcat web server and the Ignition Gateway application. You can reset the Gateway password, and even launch the web browser to the Gateway webpage.

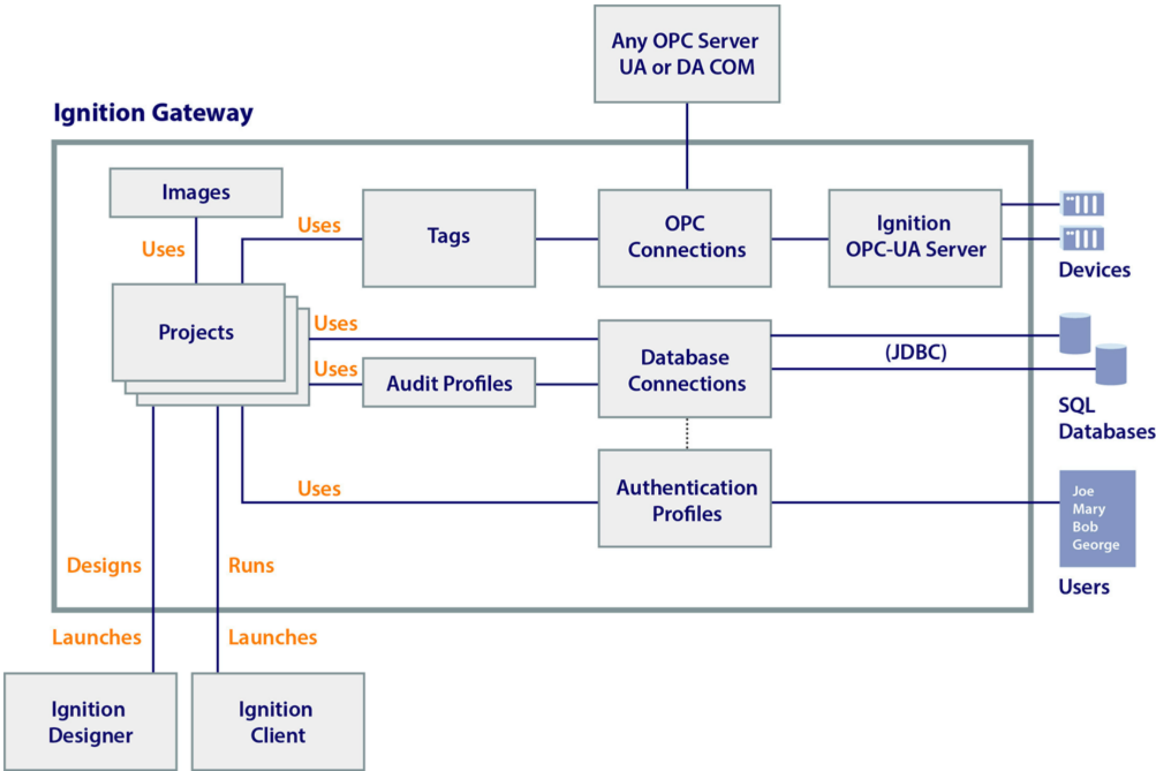
For more information, refer to the page on the Gateway Command-line Utility.

Gateway Architecture

The Gateway's architecture contains many parts, each one letting you perform a specific task. The Gateway parts are as follows:

- System Management
- Project and Module Management
- Security and Auditing Settings
- Database Connectivity
- Alarming and Modification Settings
- Tags Provider Settings
- OPC Connectivity
- Enterprise Administration Settings
- Projects - the most important part!

The following image shows the different parts of the Gateway and how projects work within the Gateway. Note the parts that are outside the Ignition Gateway box, non-Ignition OPC servers, databases, devices, etc., are all separate from Ignition but can be connected.



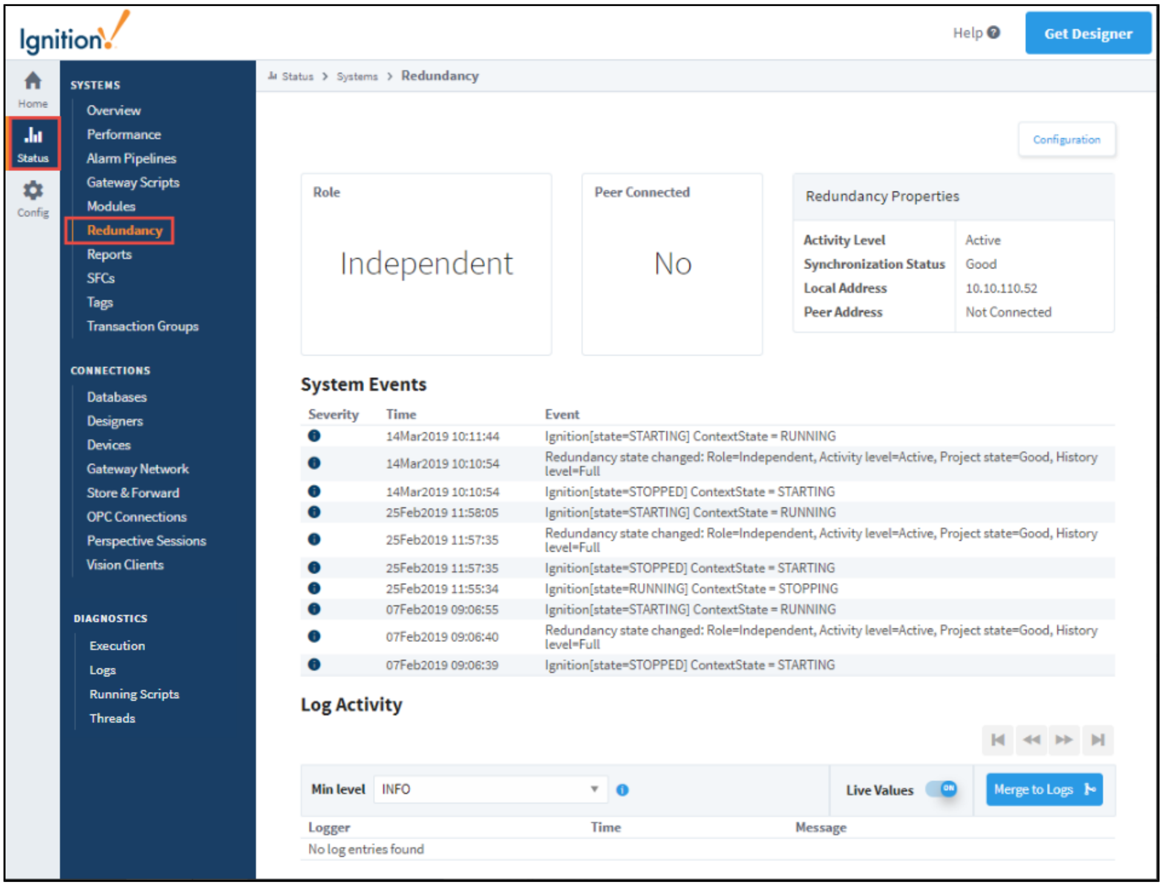
Gateway Redundancy

Ignition redundancy supports a 2-node system, meaning there are two copies of the Gateway running. One node is the Master Gateway and the other is the Backup Gateway or backup node. All projects, Gateway settings, etc., are shared between nodes. The master node manages the configuration then replicates it to the backup node.

When you have redundant systems in place, you can get detailed status information by going to Gateway webpage and selecting **Status > Redundancy** to view the system's status and events. For more about how Gateway redundancy works, go to the section on Redundancy.

How Redundancy Works

Watch the Video



Role

Independent

Peer Connected

No

Redundancy Properties

Activity Level	Active
Synchronization Status	Good
Local Address	10.10.110.52
Peer Address	Not Connected

System Events

Severity	Time	Event
●	14Mar2019 10:11:44	Ignition[state=STARTING] ContextState = RUNNING
●	14Mar2019 10:10:54	Redundancy state changed: Role=Independent, Activity level=Active, Project state=Good, History level=Full
●	14Mar2019 10:10:54	Ignition[state=STOPPED] ContextState = STARTING
●	25Feb2019 11:58:05	Ignition[state=STARTING] ContextState = RUNNING
●	25Feb2019 11:57:35	Redundancy state changed: Role=Independent, Activity level=Active, Project state=Good, History level=Full
●	25Feb2019 11:57:35	Ignition[state=STOPPED] ContextState = STARTING
●	25Feb2019 11:55:34	Ignition[state=RUNNING] ContextState = STOPPING
●	07Feb2019 09:06:55	Ignition[state=STARTING] ContextState = RUNNING
●	07Feb2019 09:06:40	Redundancy state changed: Role=Independent, Activity level=Active, Project state=Good, History level=Full
●	07Feb2019 09:06:39	Ignition[state=STOPPED] ContextState = STARTING

Log Activity

Min level INFO

Live Values ON

Merge to Logs

Logger	Time	Message
No log entries found		

In This Section ...

Home

The Homepage provides you with all of the tools you need to get started designing projects and launching Perspective Sessions and Vision Clients. There are three launchers on the Homepage that will quickly get you started:

- The Designer Launcher allows you to create or modify a project. Download the launcher and create a shortcut so it's always on your desktop when you need it.
- The Vision Client Launcher browses all Gateways for Vision projects that are available on your local network. Once your projects are added to the Vision Client Launcher, they will be displayed and available to open in a Vision Client. Simply download the launcher and create a shortcut on your desktop so it's always at your fingertips when you want to launch a Vision Client.
- The Perspective Session Launcher allows you to easily launch a session directly in your browser. Simply click the link and Ignition will display all your Perspective projects. Find your project and launch it!

While the top navigation bar is present no matter what tab you are in, the **Get Designer** button in the upper right corner does not require that a user login to the Gateway. This allows a user to launch the Designer right away, only logging in once the Designer has fully launched. See the Designer section for more information on launching and using the Designer.

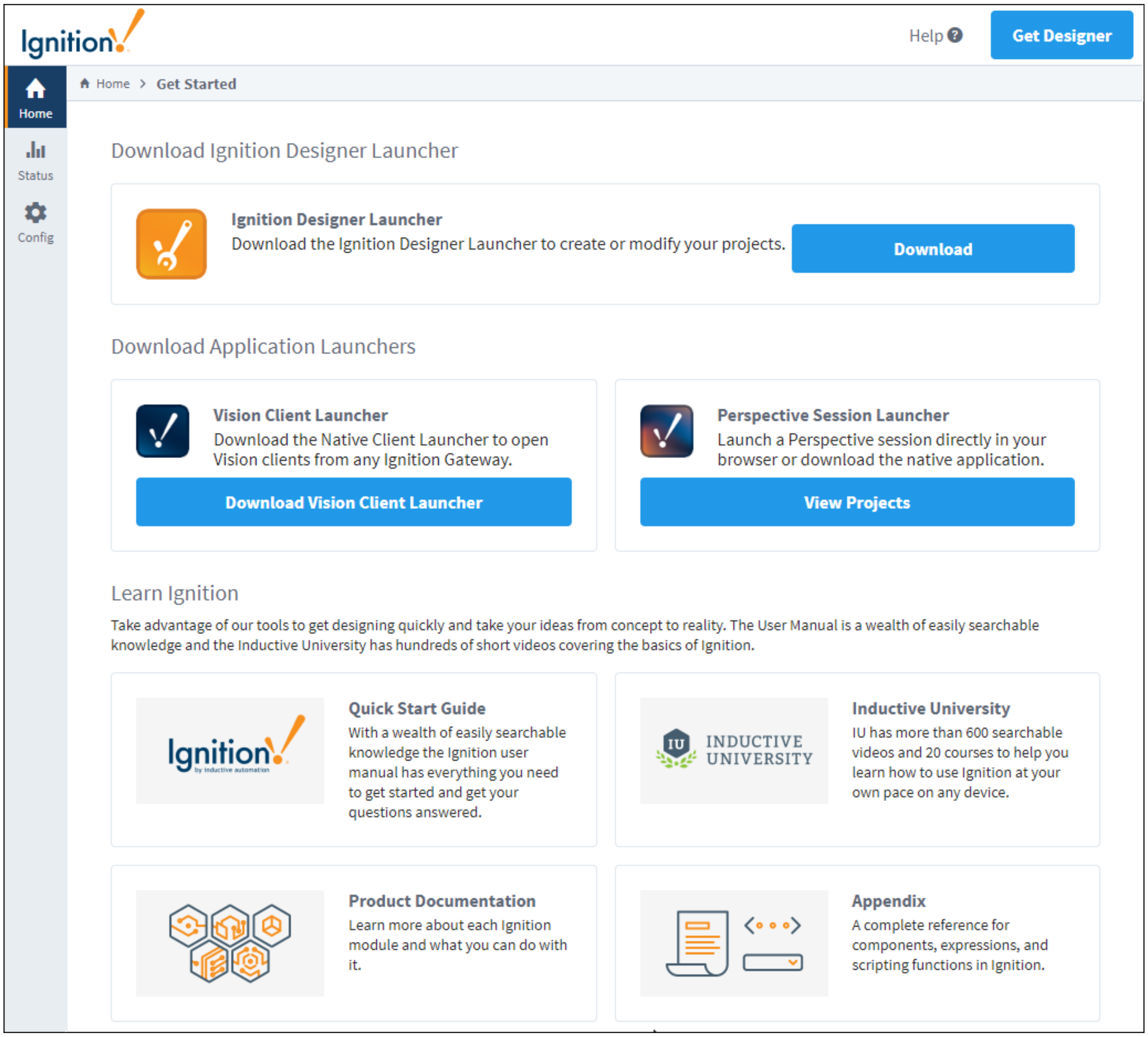
Resources

On the Homepage, we also provide quick links to several resources: Quick Start Guide, Inductive University, Product Documentation, and the Appendix which contains a complete reference for components, expressions, and scripting functions in Ignition.

On this page ...

Customizing the Gateway Homepage

Watch the Video



Download Ignition Designer Launcher

Ignition Designer Launcher
Download the Ignition Designer Launcher to create or modify your projects.

Download Application Launchers

Vision Client Launcher
Download the Native Client Launcher to open Vision clients from any Ignition Gateway.

Perspective Session Launcher
Launch a Perspective session directly in your browser or download the native application.

Learn Ignition

Take advantage of our tools to get designing quickly and take your ideas from concept to reality. The User Manual is a wealth of easily searchable knowledge and the Inductive University has hundreds of short videos covering the basics of Ignition.

Quick Start Guide
With a wealth of easily searchable knowledge the Ignition user manual has everything you need to get started and get your questions answered.

Inductive University
IU has more than 600 searchable videos and 20 courses to help you learn how to use Ignition at your own pace on any device.

Product Documentation
Learn more about each Ignition module and what you can do with it.

Appendix
A complete reference for components, expressions, and scripting functions in Ignition.

Related Topics ...

- Status
- Config
- Quick Start - Perspective Session
- Quick Start - Vision Client
- Inductive University

Status

Your Gateway at a Glance

The Status tab provides both an 'at a glance' overview of all of the systems in your Gateway, while also giving you the ability to drill down into specific systems and get a more in depth view of what is currently happening in your system. The Systems and Connections sections that are displayed on the left side of the Status page are based on what modules are currently installed. Some 3rd party modules could potentially add sections that are not discussed in this manual.

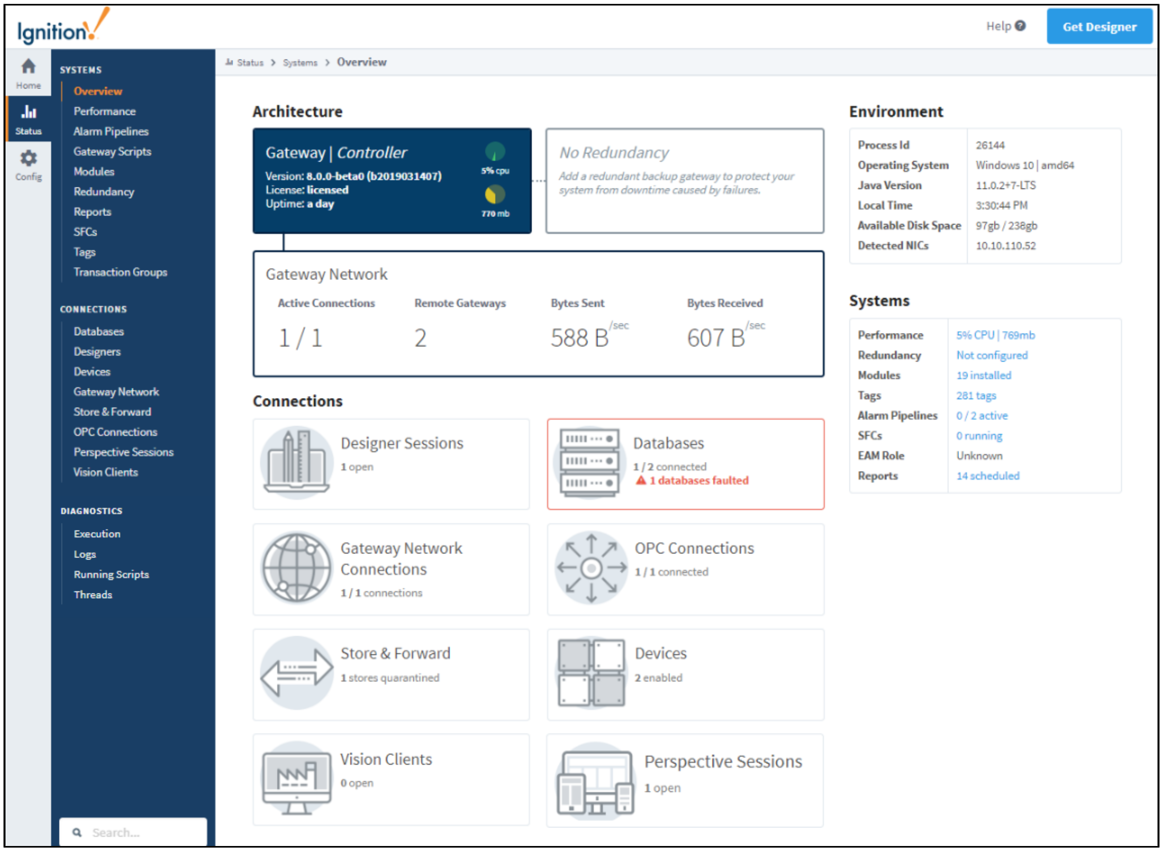
When first opening the Status tab, you will be taken to the Overview page. This page provides an overview of all Ignition systems, while also acting as a launchpad to all other sections in the Status tab. The pages in the Status tab are built around the idea of quickly drawing attention to problem areas by highlighting them. As you can see, this Gateway has a faulted database. The pages in the Status tab also allow you to 'drill down' into sections to see more information. Most objects can be clicked on, like the faulted database. Clicking either of these will take you to the appropriate section, allowing you to quickly find out what is wrong with a particular system.

On this page

...

Gateway Status Page Troubleshooting

Watch the Video



Architecture

Gateway | *Controller*
Version: **8.0.0-beta0 (b2019031407)**
License: **licensed**
Uptime: **a day**
5% cpu
770 mb

No Redundancy
Add a redundant backup gateway to protect your system from downtime caused by failures.

Gateway Network

Active Connections	Remote Gateways	Bytes Sent	Bytes Received
1 / 1	2	588 B /sec	607 B /sec

Connections

Designer Sessions
1 open

Databases
1 / 2 connected
⚠ 1 databases faulted

Gateway Network Connections
1 / 1 connections

OPC Connections
1 / 1 connected

Store & Forward
1 stores quarantined

Devices
2 enabled

Vision Clients
0 open

Perspective Sessions
1 open

Environment

Process Id	26144
Operating System	Windows 10 | amd64
Java Version	11.0.2+7-LTS
Local Time	3:30:44 PM
Available Disk Space	97gb / 238gb
Detected NICs	10.10.110.52

Systems

Performance	5% CPU | 769mb
Redundancy	Not configured
Modules	19 installed
Tags	281 tags
Alarm Pipelines	0 / 2 active
SFCs	0 running
EAM Role	Unknown
Reports	14 scheduled

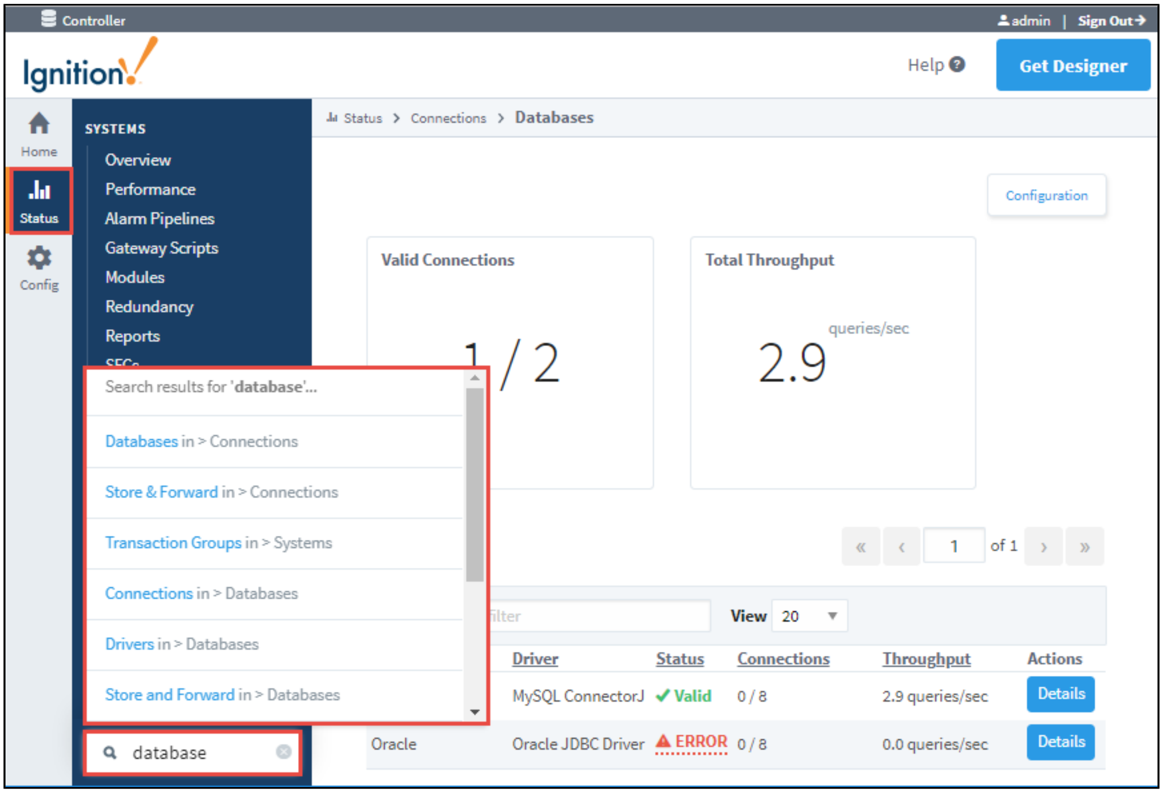
The Status section of the Gateway Webpage provides detailed information relating to the following parts:

Systems	
Overview	Provides a top-down view of many of the components of your Gateway. This view is also useful for determining what step might be next when setting up your Ignition Gateway for the first time. You can view the status of your database connections, device connections, OPC connections, the number of open Clients and the number of open Designers. The following feature is new in Ignition version **8.0.6** Click here to check out the other new features The **Status > Overview** page displays the number of Perspective Sessions currently running.
Performance	Displays the performance status for the Ignition system such as CPU, Memory and Threads.
Alarm Pipelines	Shows the status details about the alarm notification pipelines. You can see the status of an alarm and where the alarm is in the pipeline.
Gateway Scripts	Shows status details about all the Gateway event scripts running in Gateway. You can see information such as their execution status, whether the scripts are running or not, and so on.
Modules	A list of installed modules, their status, as well information about their version and current license state.
Redundancy	Lists information about the current state of Redundancy in Ignition. This information is only helpful when connected to another redundant Ignition server.
Reports	Displays information about the current and scheduled reports on the Gateway if you are using the Reporting module.
SFCs	Displays information about Sequential Function Chart (SFC) instances.
Tags	Lists information and statistics about all configured Tag Providers as well as a view into the Tag subscription model, scan classes, and what tags it is currently subscribed to.
Transaction Groups	Displays information about the current Transaction Groups.
Connections	
Databases	Displays a list of configured databases, and if they have a valid connection or not. Shows active queries, long running queries, the number of queries a second that are running, as well as a trend showing the percentage of queries that completed in that time.
Designers	Displays information on currently running Designer sessions along with some information about each session.
Devices	Displays a list of currently configured OPC UA devices and which are connected and which have a faulty connection. It also shows how many Tags Ignition is requesting from the device along with how often it is requesting them. This information is used to determine if the device is overloaded with too many requests too quickly, or if the device is being under utilized.
Gateway Network	Shows an overview of the status of all Gateways within the Gateway Network. It also provides some metrics for each Gateway, giving an idea of the rate of data transfer between two Gateways, as well as a list of connection events.
Store & Forward	Displays a list of Store and Forward engines, including status, as well as the number of records currently in each Store and Forward system.
OPC Connections	Displays a list of all current OPC connections and their status.
Perspective Sessions	Shows a list of current Perspective sessions and details about each session.
Vision Clients	Shows a list of current Vision Clients and details about each client.
Diagnostics	
Execution	Displays a status of all tasks that your Gateway runs on a schedule, such as duration and execution time of an alarm journal update or the average time it takes a Gateway to execute a Tag Group.
Logs	Displays errors caused by Gateway events like database or device connections, authentication profiles, alarm journals, and pipelines. Logs include a wealth of information about the running state of the Gateway.

Running Scripts	The following feature is new in Ignition version **8.0.11** Click here to check out the other new features Shows all actively running Gateway scripts, as well as providing a way to terminate any running script. In addition, Vision client and Designer consoles have a Running Scripts tab, which also lists running scripts and provides a way to terminate them.
Threads	Shows what each thread is doing in the Gateway, including their state and CPU usages.

Search

The Status tab also displays a search bar at the bottom left of each page. This search bar allows you to type in a word or phrase, and it will list all appropriate pages in both the Status and Config tabs to easily find all pages related to a specific system.



Search results for 'database'...

Databases in > Connections

Store & Forward in > Connections

Transaction Groups in > Systems

Connections in > Databases

Drivers in > Databases

Store and Forward in > Databases

	Driver	Status	Connections	Throughput	Actions
	MySQL ConnectorJ	✓ Valid	0 / 8	2.9 queries/sec	Details
Oracle	Oracle JDBC Driver	⚠ ERROR	0 / 8	0.0 queries/sec	Details

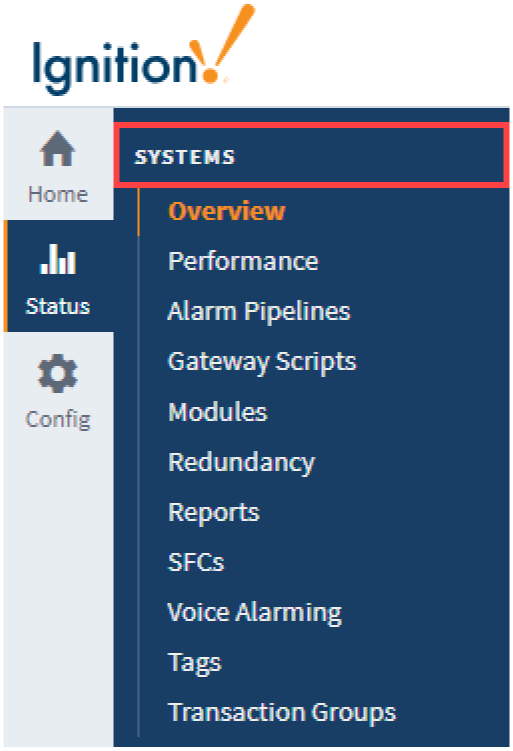
Related Topics ...

- Config
- Home

In This Section ...

Systems

The Systems section of the Status tab displays information regarding the internal Gateway systems. This includes systems such as the currently running Transaction Groups as well as active Alarm Pipelines. These sections are not necessarily concerned with any connections to external systems.



On this page ...

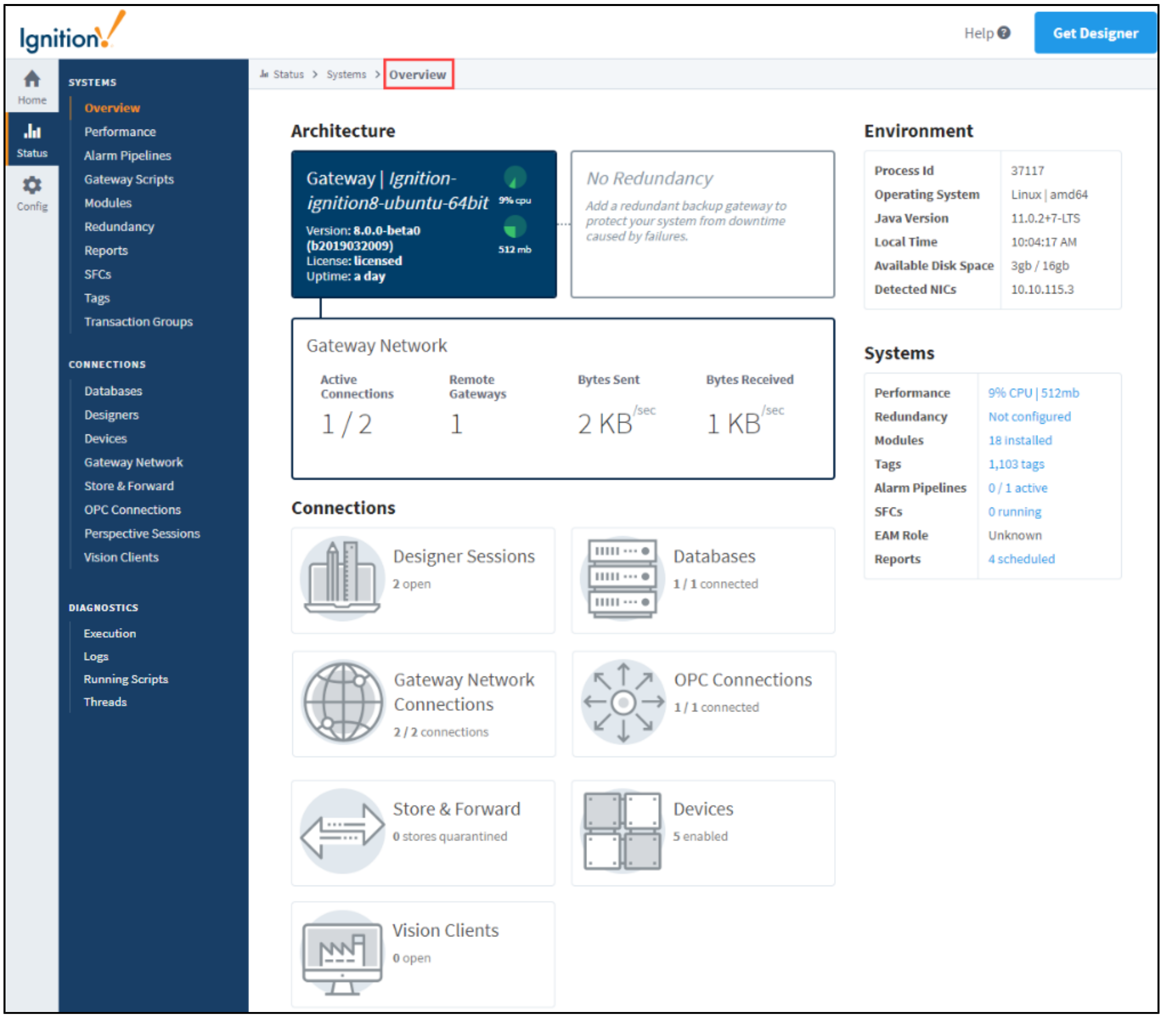
INDUCTIVE UNIVERSITY

Gateway Status Page Troubleshooting

Watch the Video

Overview Section

The Overview page is the first page that you see when navigating to the Status tab. It provides a visual synopsis of everything in the Gateway, as well as some basic information about the server Ignition is installed on. There are four sections on the page: **Architecture**, **Environment**, **Systems**, and **Connections**.



Architecture

Gateway | *Ignition-ignition8-ubuntu-64bit*
9% cpu
512 mb
Version: **8.0.0-beta0 (b2019032009)**
License: **licensed**
Uptime: **a day**

No Redundancy

Add a redundant backup gateway to protect your system from downtime caused by failures.

Gateway Network

Active Connections	Remote Gateways	Bytes Sent	Bytes Received
1 / 2	1	2 KB/sec	1 KB/sec

Connections

Designer Sessions
2 open

Databases
1 / 1 connected

Gateway Network Connections
2 / 2 connections

OPC Connections
1 / 1 connected

Store & Forward
0 stores quarantined

Devices
5 enabled

Vision Clients
0 open

Environment

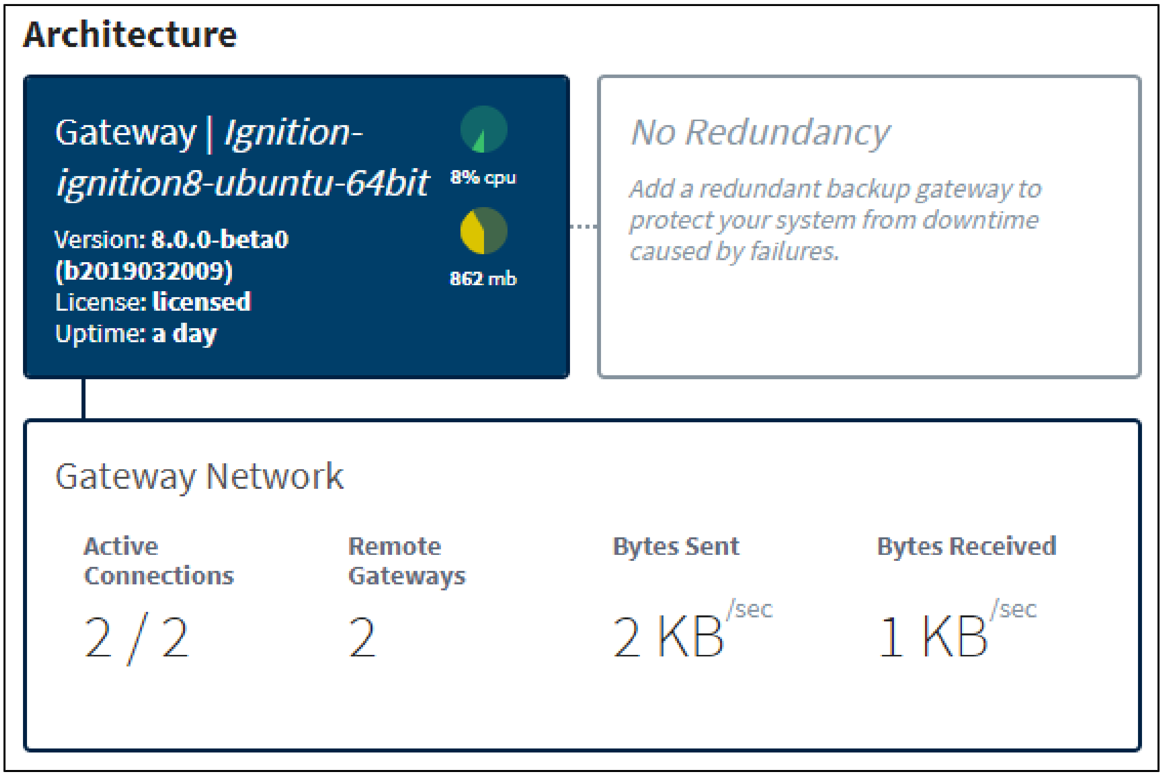
Process Id	37117
Operating System	Linux | amd64
Java Version	11.0.2+7-LTS
Local Time	10:04:17 AM
Available Disk Space	3gb / 16gb
Detected NICs	10.10.115.3

Systems

Performance	9% CPU | 512mb
Redundancy	Not configured
Modules	18 installed
Tags	1,103 tags
Alarm Pipelines	0 / 1 active
SFCs	0 running
EAM Role	Unknown
Reports	4 scheduled

Architecture

The Architecture segment provides a top down view of your Gateway. You can see your Ignition version and resource usage, as well as similar information with a redundant Gateway, if one is connected. You also get an overview of the Gateway Network including how many active connections there are, and what is the amount of data being transferred between them.



Architecture

Gateway | *Ignition-ignition8-ubuntu-64bit*

Version: **8.0.0-beta0 (b2019032009)**
License: **licensed**
Uptime: **a day**

No Redundancy

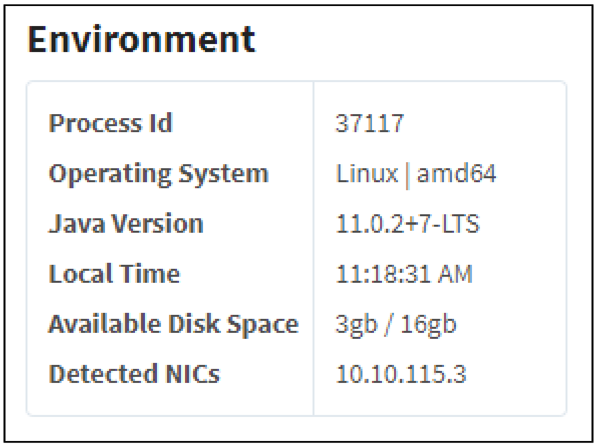
Add a redundant backup gateway to protect your system from downtime caused by failures.

Gateway Network

Active Connections	Remote Gateways	Bytes Sent	Bytes Received
2 / 2	2	2 KB/sec	1 KB/sec

Environment

The Environment segment provides some basic information about your server such as the Operating System (OS), Java version, available disk space, and IP address of detected network interface cards (NIC)..

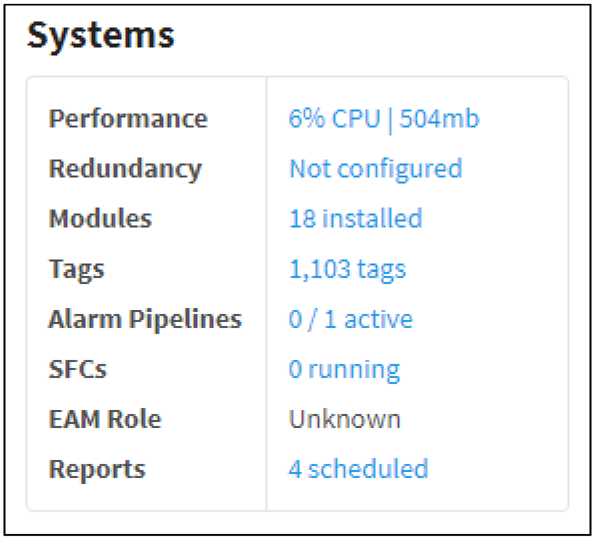


Environment

Process Id	37117
Operating System	Linux | amd64
Java Version	11.0.2+7-LTS
Local Time	11:18:31 AM
Available Disk Space	3gb / 16gb
Detected NICs	10.10.115.3

Systems

The Systems segment summarizes each of Ignition's systems. Here, you can see how many modules you have installed as well as how many Tags are configured in your Gateway, along with a handful of other information. Click on any of the links in blue to find out more information about a particular system.

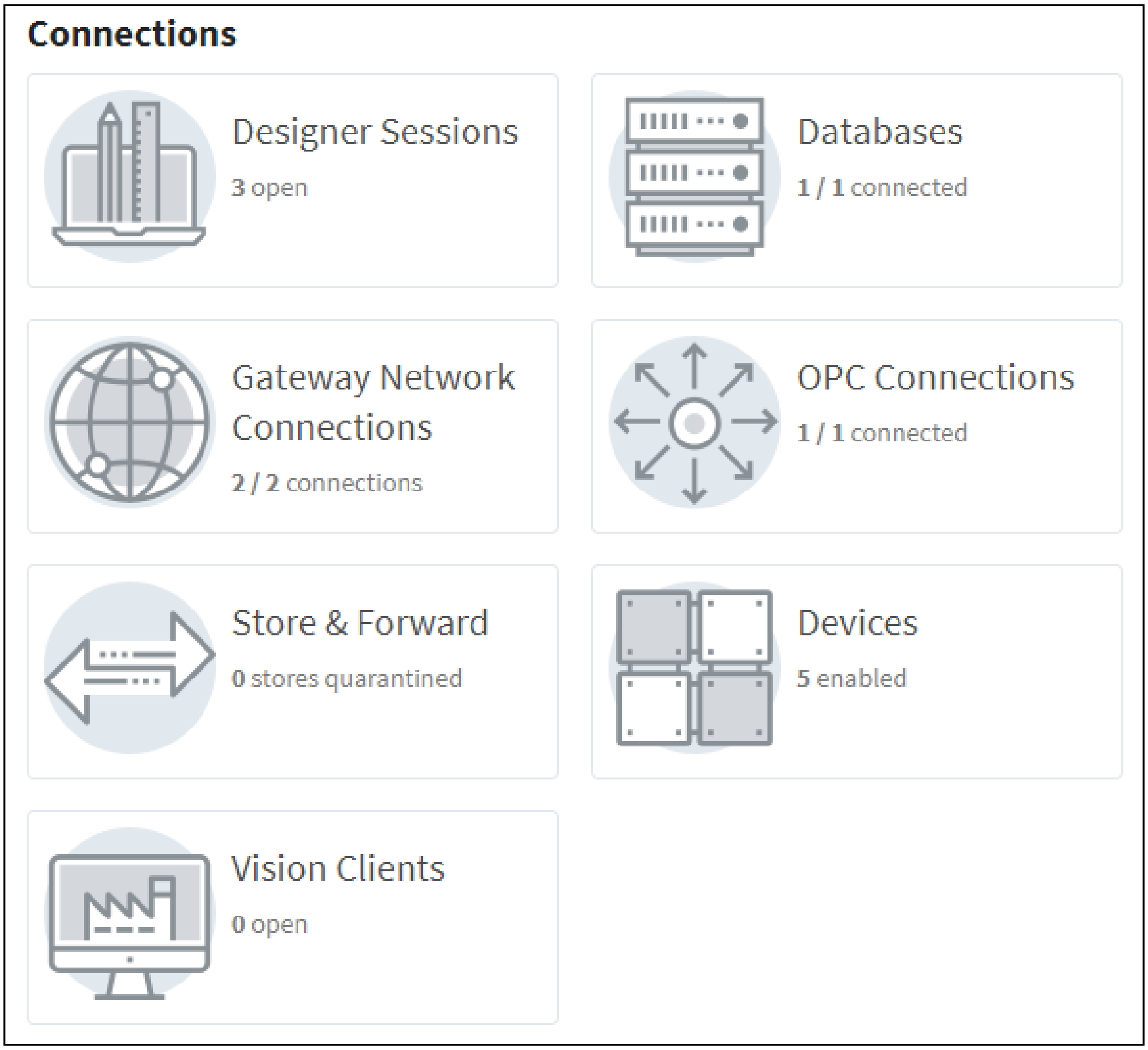


Systems

Performance	6% CPU | 504mb
Redundancy	Not configured
Modules	18 installed
Tags	1,103 tags
Alarm Pipelines	0 / 1 active
SFCs	0 running
EAM Role	Unknown
Reports	4 scheduled

Connections

The Connections segment shows all of the systems that Ignition is connected to. If any of these have issues, they will be outlined in red. Clicked on the connection to be taken to the relevant page within the Status tab.



Connections

Designer Sessions
3 open

Databases
1 / 1 connected

Gateway Network Connections
2 / 2 connections

OPC Connections
1 / 1 connected

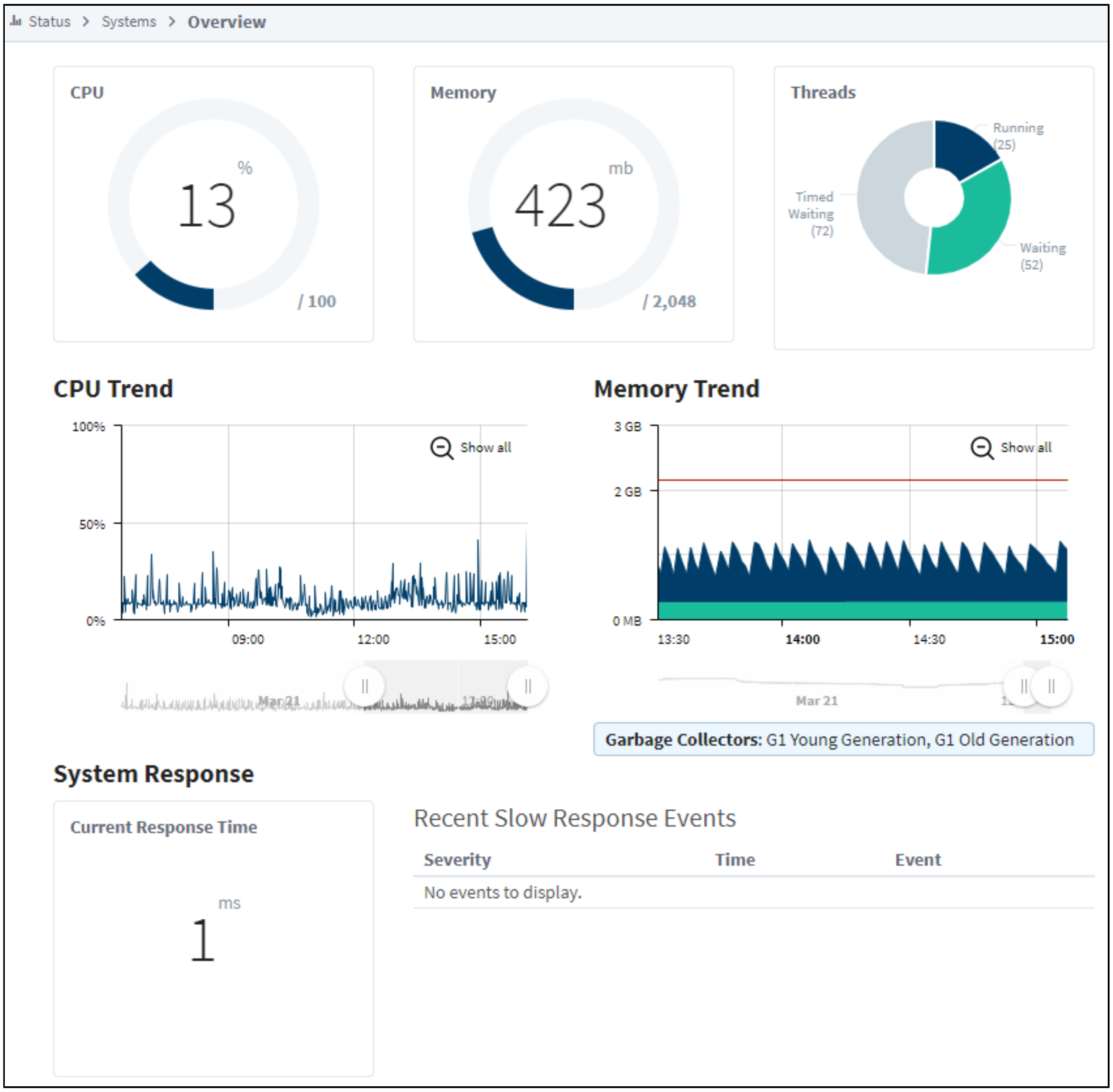
Store & Forward
0 stores quarantined

Devices
5 enabled

Vision Clients
0 open

Performance

The Performance page displays information on the resource usage of the Gateway. This page enables you to quickly assess your Gateway's overall health and determine if the Gateway is being taxed too heavily for the server it's running on. You can see the current CPU usage, the current memory usage (in megabytes) out of the total allocated for the Gateway, and how CPU threads are performing. In the CPU Trend and Memory Trend sections there are sliders you can move to set the time range anywhere with in the previous 24 hours. The lower portion of the page shows the current system response time as well as a log of any recent slow response events.

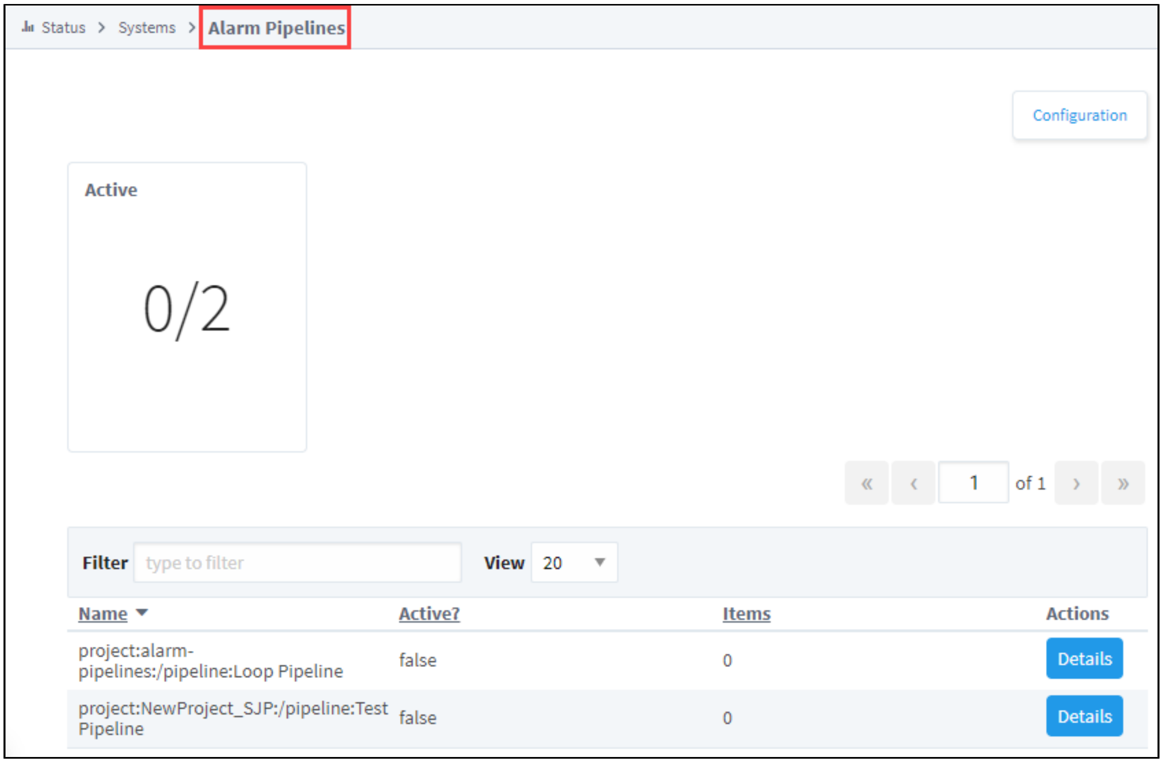


Garbage Collectors: G1 Young Generation, G1 Old Generation

Severity	Time	Event
No events to display.		

Alarm Pipelines

The Alarm Pipelines page lists the currently configured alarm pipelines in the Gateway. Each pipeline displays the number of alarm items currently running through it. Click on the Details button to the right to open up a new page that will show the alarm pipeline status and logs. There will also be a set of tabs at the top of the page. One for the Pipeline Status that is already open, the other for Pipeline Logs. This log viewer will act as a miniature log viewer that will only show logs that pertain to this particular pipeline.

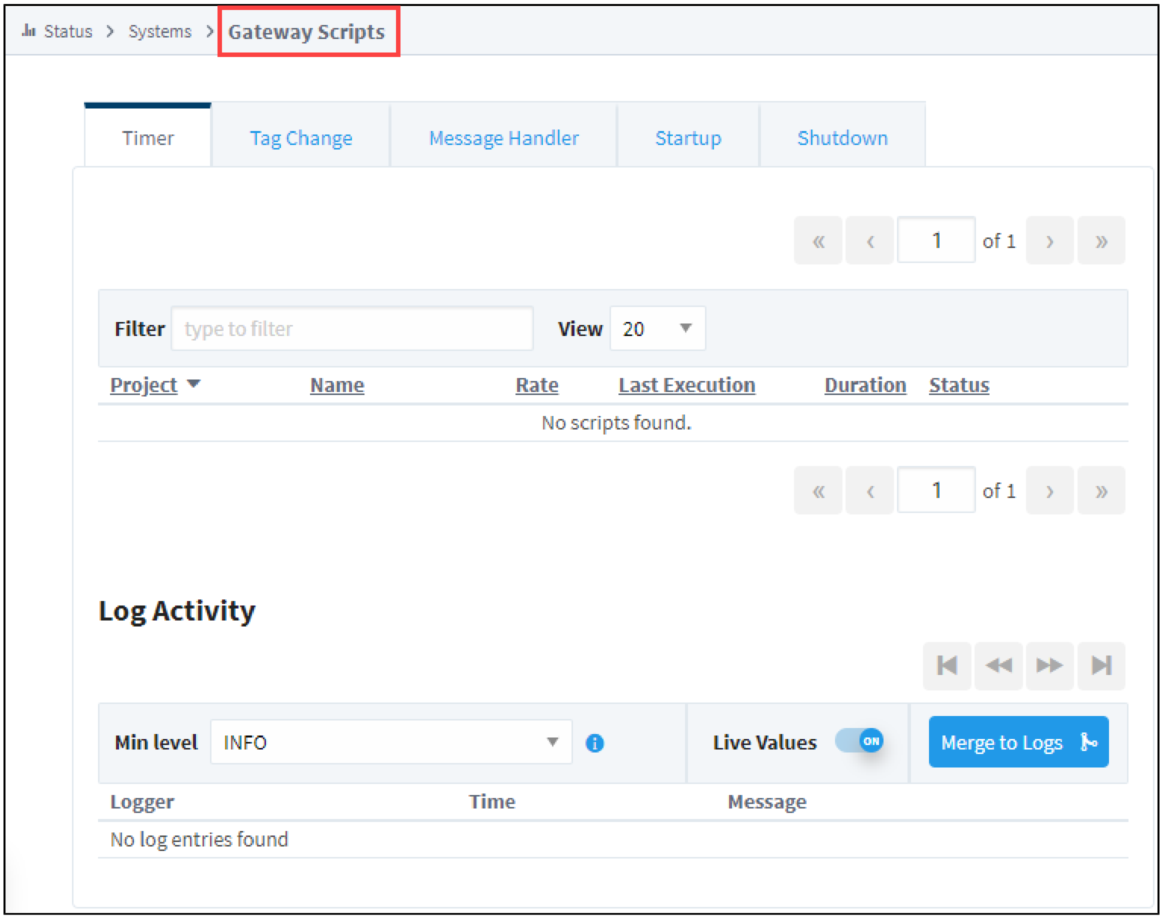


Name	Active?	Items	Actions
project:alarm-pipelines:/pipeline:Loop Pipeline	false	0	Details
project:NewProject_SJP:/pipeline:Test Pipeline	false	0	Details

Gateway Scripts

The Gateway Scripts page displays a list of all currently configured Gateway scripts. These are scripts that are configured in the Gateway Event Scripts section of the project. The tabs at the top of the page will swap between the different types of Gateway Event Scripts: Timer, Tag Change, Message Handler, Startup, and Shutdown. They provide useful information such as the name of the project that it is running in as well as the last time it ran, the duration and status (if it was successful or not).

With this page you can quickly verify that your scripts are running properly. If a script has an error, click on the error for more details about what went wrong with the script. All of the logs for that section are also contained in the log viewer at the bottom of the page, to make it easy to see a list of all of the logged errors for that set of Gateway Event Scripts.

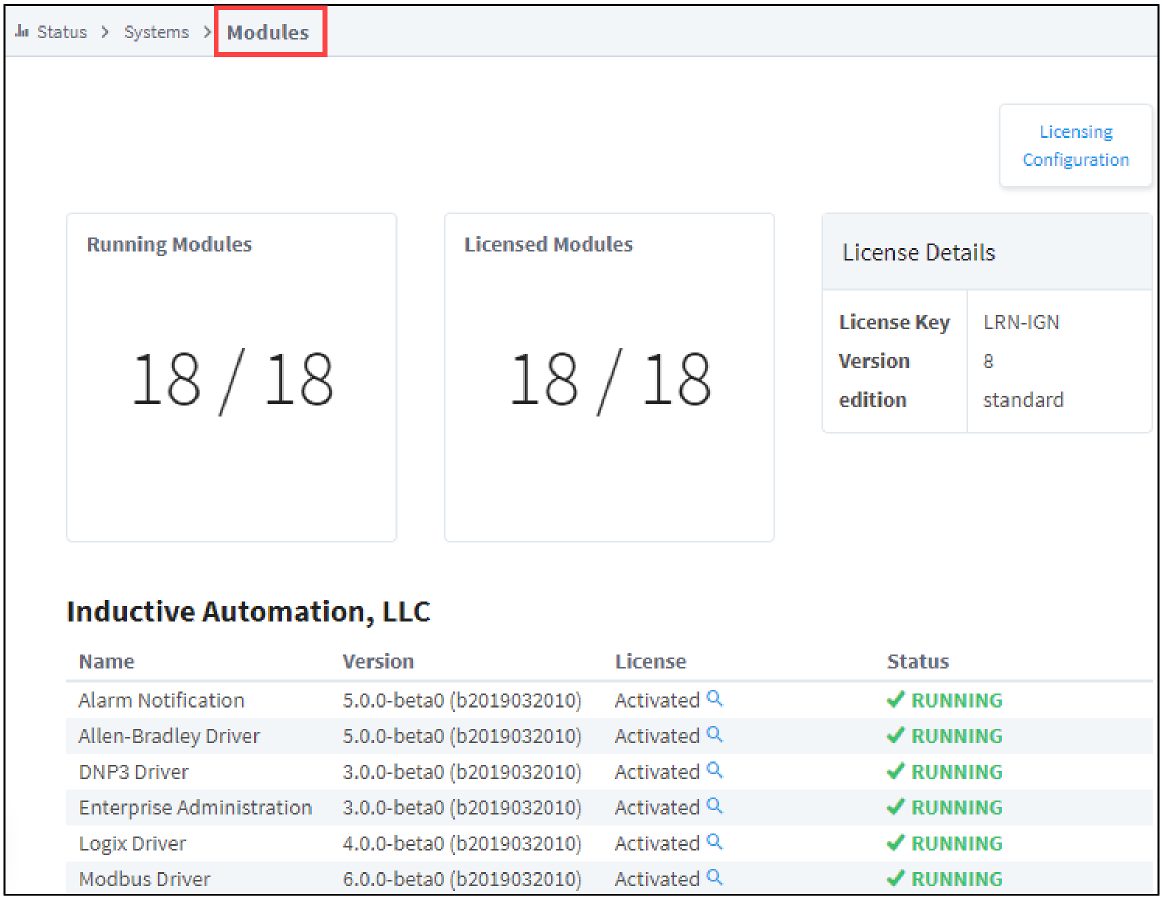


Project	Name	Rate	Last Execution	Duration	Status
No scripts found.					

Logger	Time	Message
No log entries found		

Modules

The Modules page shows a list of the currently installed modules, their current version, and if they are running properly with a license. The top of the page shows a quick count of the running and licensed modules, so you can easily tell if anything is not working. It is then easy to scroll down through the list to see which modules are having issues.



Running Modules

18 / 18

Licensed Modules

18 / 18

License Details

License Key	LRN-IGN
Version	8
edition	standard

Inductive Automation, LLC

Name	Version	License	Status
Alarm Notification	5.0.0-beta0 (b2019032010)	Activated	✓ RUNNING
Allen-Bradley Driver	5.0.0-beta0 (b2019032010)	Activated	✓ RUNNING
DNP3 Driver	3.0.0-beta0 (b2019032010)	Activated	✓ RUNNING
Enterprise Administration	3.0.0-beta0 (b2019032010)	Activated	✓ RUNNING
Logix Driver	4.0.0-beta0 (b2019032010)	Activated	✓ RUNNING
Modbus Driver	6.0.0-beta0 (b2019032010)	Activated	✓ RUNNING

Redundancy

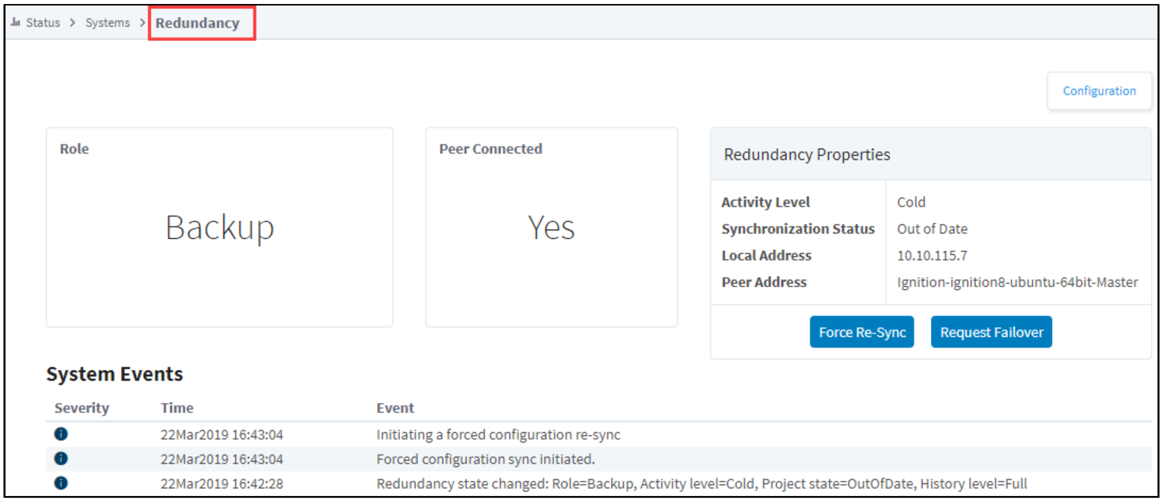
The Redundancy section displays information regarding the redundant system, if one is configured. It easily shows the Role of the Gateway you are viewing, the status of the connected Gateway, and their IP addresses. The trends on this page give a snapshot of the last few minutes of communication between the two Gateways. The first trend shows the data that is being sent and received between the two Gateways, the second shows the state updates that have recently occurred. The bottom of the page contains a log of the last system events, to easily track major events between the Gateways.

Force Re-Sync

The **Force Re-Sync** button forces a full synchronization of the redundant configuration state. The backup node will be forced to restart.

Request Failover

The **Request Failover** button switches the active node in a redundant pair.



Role

Backup

Peer Connected

Yes

Redundancy Properties

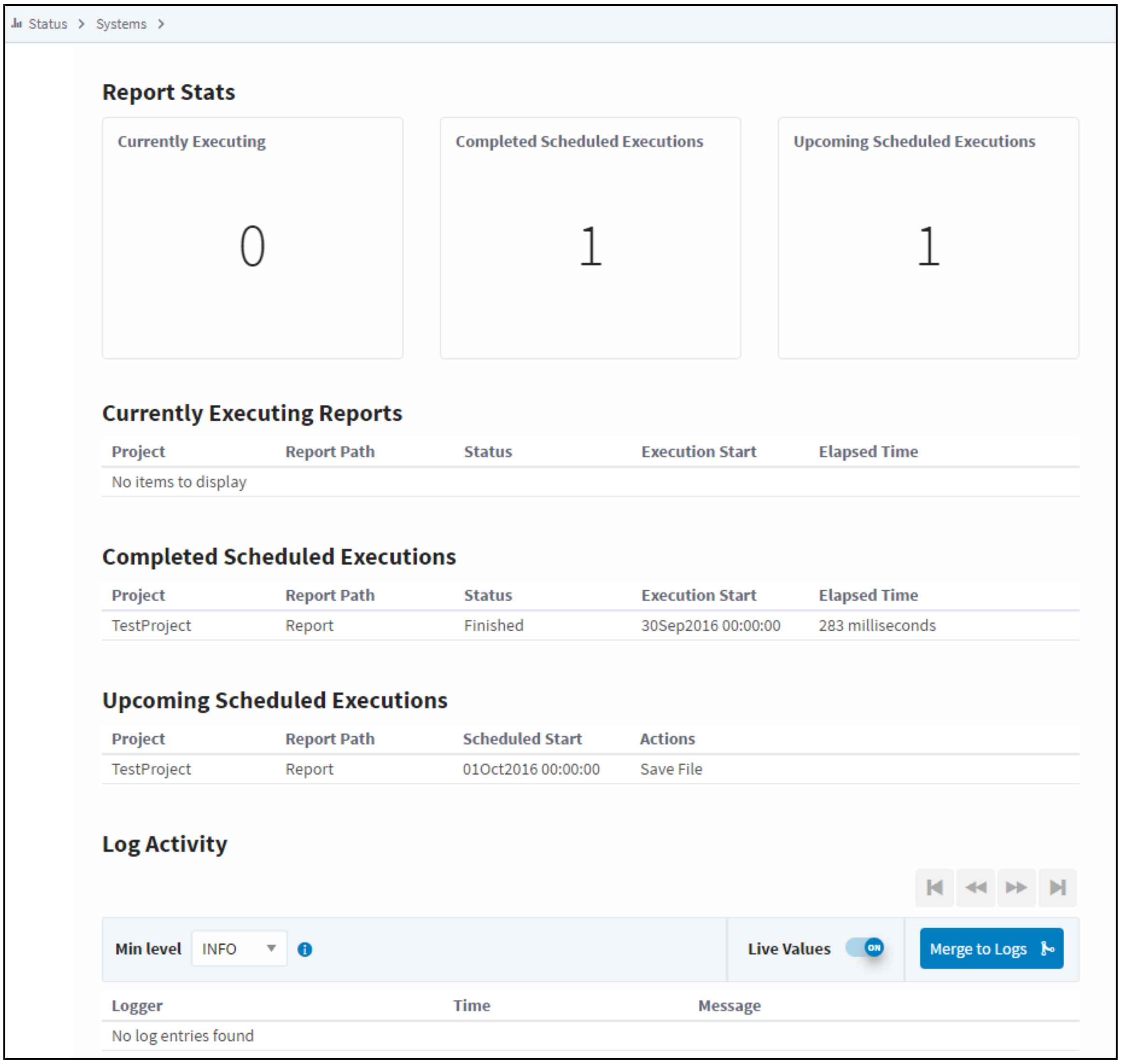
Activity Level	Cold
Synchronization Status	Out of Date
Local Address	10.10.115.7
Peer Address	Ignition-ignition8-ubuntu-64bit-Master

System Events

Severity	Time	Event
ℹ	22Mar2019 16:43:04	Initiating a forced configuration re-sync
ℹ	22Mar2019 16:43:04	Forced configuration sync initiated.
ℹ	22Mar2019 16:42:28	Redundancy state changed: Role=Backup, Activity level=Cold, Project state=OutOfDate, History level=Full

Reports

The Reports page shows information on any Reports that have at least one scheduled action set up. The top of the page gives a quick count of reports that are executing an action, have executed an action, or are going to execute an action. Below you can also find a list of Reports in those states as well, to quickly see what project they are located in, or what their last execution time was. As with most other pages, there is also a log of all events related to reports at the bottom of the page.



Report Stats

Currently Executing	Completed Scheduled Executions	Upcoming Scheduled Executions
0	1	1

Currently Executing Reports

Project	Report Path	Status	Execution Start	Elapsed Time
No items to display				

Completed Scheduled Executions

Project	Report Path	Status	Execution Start	Elapsed Time
TestProject	Report	Finished	30Sep2016 00:00:00	283 milliseconds

Upcoming Scheduled Executions

Project	Report Path	Scheduled Start	Actions
TestProject	Report	01Oct2016 00:00:00	Save File

Log Activity

Min level: INFO

Live Values: ON

Logger	Time	Message
No log entries found		

SFCs

The Sequential Function Charts (SFCs) section displays a list of all of the currently configured SFCs along with a count of the currently running charts at the top of the page. Click the **Details** button next to any SFC for details about that particular Chart.

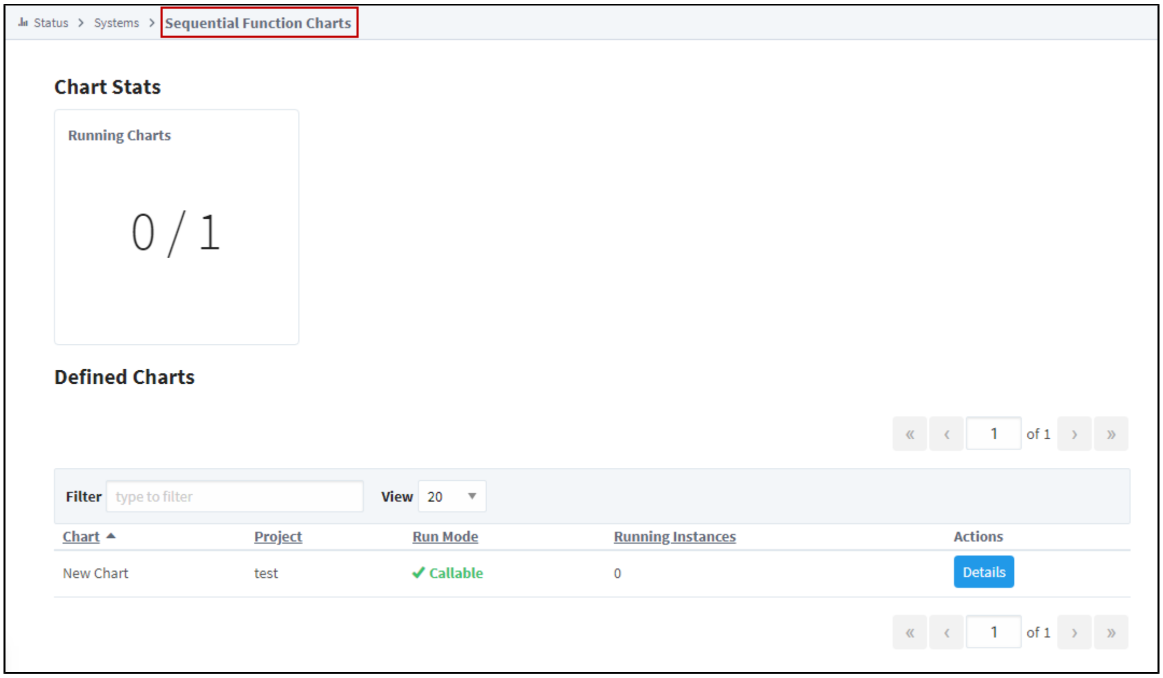


Chart Stats

Running Charts

0 / 1

Defined Charts

Chart ▲	Project	Run Mode	Running Instances	Actions
New Chart	test	✓ Callable	0	Details

On the Details for an individual SFC, there are two tabs: Chart Instances and Chart Logs. It contains a list of all currently running Chart instances, and allows you to swap tabs to look at any logged events for that particular chart.

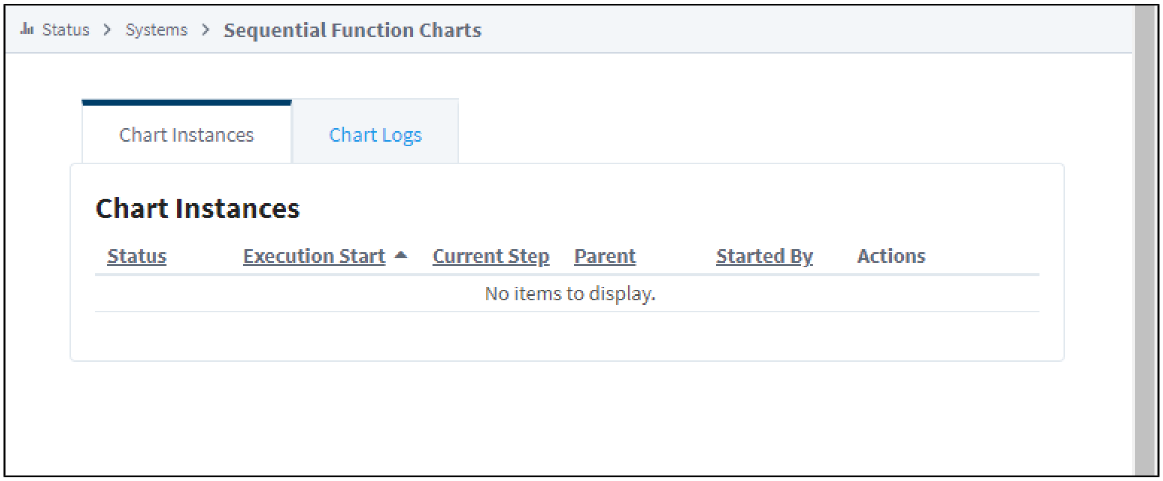
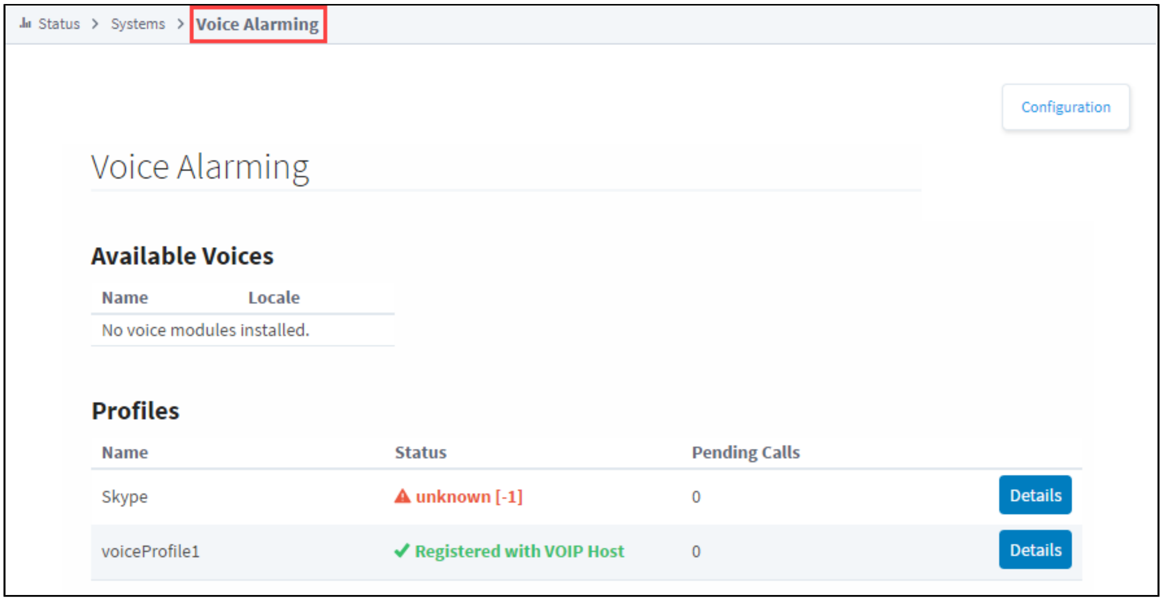


Chart Instances

Status	Execution Start ▲	Current Step	Parent	Started By	Actions
No items to display.					

Voice Alarming

The Voice Alarming page provides details about the currently configured Voice Notification Profiles on the Gateway. This status can be used to quickly see which Notification Profiles are not working. Clicking the details button enables you to see the current calls with that Voice Notification profile and the current queue of calls waiting to be completed.



Voice Alarming

Available Voices

Name	Locale
No voice modules installed.	

Profiles

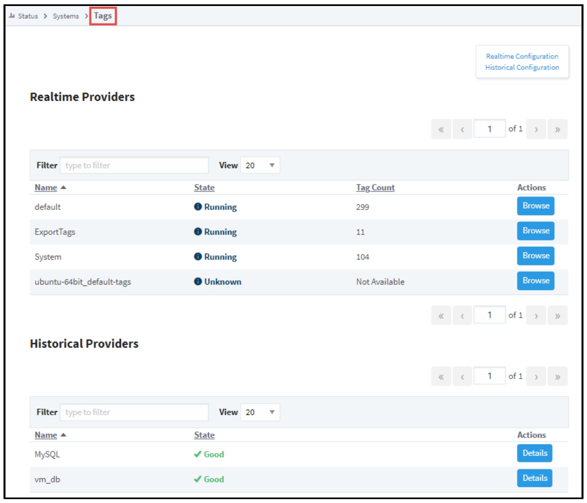
Name	Status	Pending Calls
Skype	⚠ unknown [-1]	0
voiceProfile1	✓ Registered with VOIP Host	0

Tags

The Tags page contains a lot of information about the Gateway's configured Tags and Tag Providers. The first page will show a list of all realtime and historical Tag Providers that make it easy to see if any of them are having issues. Clicking on an errored provider will show an error message that may help fix the problem. Clicking on the **Details** button to the left will open up a page with more information about the Tags in that particular provider.

Tag Diagnostics

Watch the Video



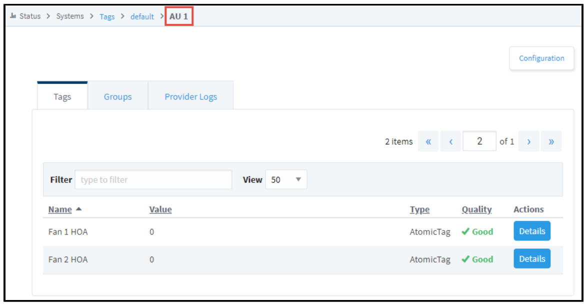
Realtime Providers

Name ▲	State	Tag Count	Actions
default	● Running	299	Browse
ExportTags	● Running	11	Browse
System	● Running	104	Browse
ubuntu-64bit_default-tags	● Unknown	Not Available	Browse

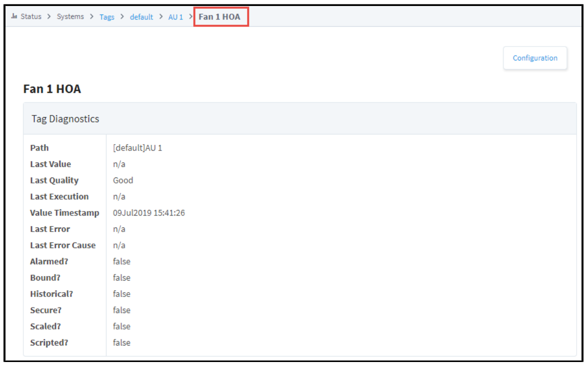
Historical Providers

Name ▲	State	Actions
MySQL	✓ Good	Details
vm_db	✓ Good	Details

The details of the Tag Provider will show a list of all currently configured Tags in that Tag Provider, and show some basic information about them. You can browse through the Tag structure to get more information about each Tag. When clicking the Details button for an individual Tag, you will be brought to a Tag Diagnostics screen that will provide more detailed information about that particular Tag, such as if it has a script written in its Tag event scripts and what its last value was, among other information.



Name ▲	Value	Type	Quality	Actions
Fan 1 HOA	0	AtomicTag	✓ Good	Details
Fan 2 HOA	0	AtomicTag	✓ Good	Details

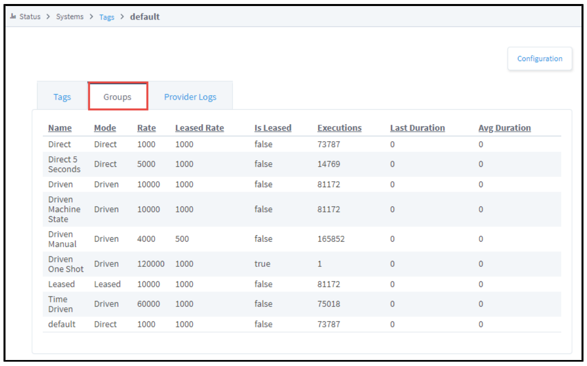


Fan 1 HOA

Tag Diagnostics

Path	[default]AU 1
Last Value	n/a
Last Quality	Good
Last Execution	n/a
Value Timestamp	09Jul2019 15:41:26
Last Error	n/a
Last Error Cause	n/a
Alarmed?	false
Bound?	false
Historical?	false
Secure?	false
Scaled?	false
Scripted?	false

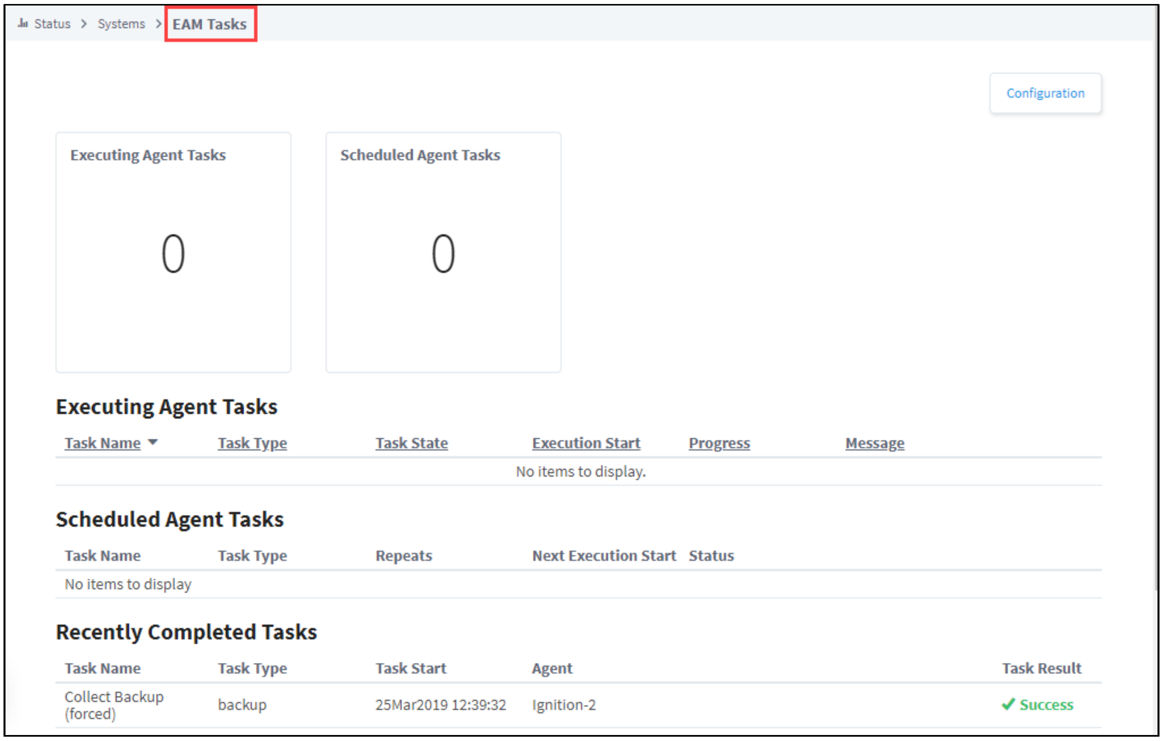
The Groups Tab allows you to see the Tag Groups configured. It also shows the number of executions for each Tag Group, the last time the Group was executed and the average duration in milliseconds. Provider Logs display any events that relate to that particular provider.



Name	Mode	Rate	Leased Rate	Is Leased	Executions	Last Duration	Avg Duration
Direct	Direct	1000	1000	false	73787	0	0
Direct 5 Seconds	Direct	5000	1000	false	14769	0	0
Driven	Driven	10000	1000	false	81172	0	0
Driven Machine State	Driven	10000	1000	false	81172	0	0
Driven Manual	Driven	4000	500	false	165852	0	0
Driven One Shot	Driven	120000	1000	true	1	0	0
Leased	Leased	10000	1000	false	81172	0	0
Time Driven	Driven	60000	1000	false	75018	0	0
default	Direct	1000	1000	false	73787	0	0

EAM Tasks

The EAM Tasks section will only show up once the EAM module is configured. This page shows information regarding the currently running and scheduled tasks, as well as tasks that were recently executed and whether they were successful or not. From here you can pause a scheduled task, or see the error that caused a previous execution to fail.



Executing Agent Tasks

0

Scheduled Agent Tasks

0

Executing Agent Tasks

Task Name	Task Type	Task State	Execution Start	Progress	Message
No items to display.					

Scheduled Agent Tasks

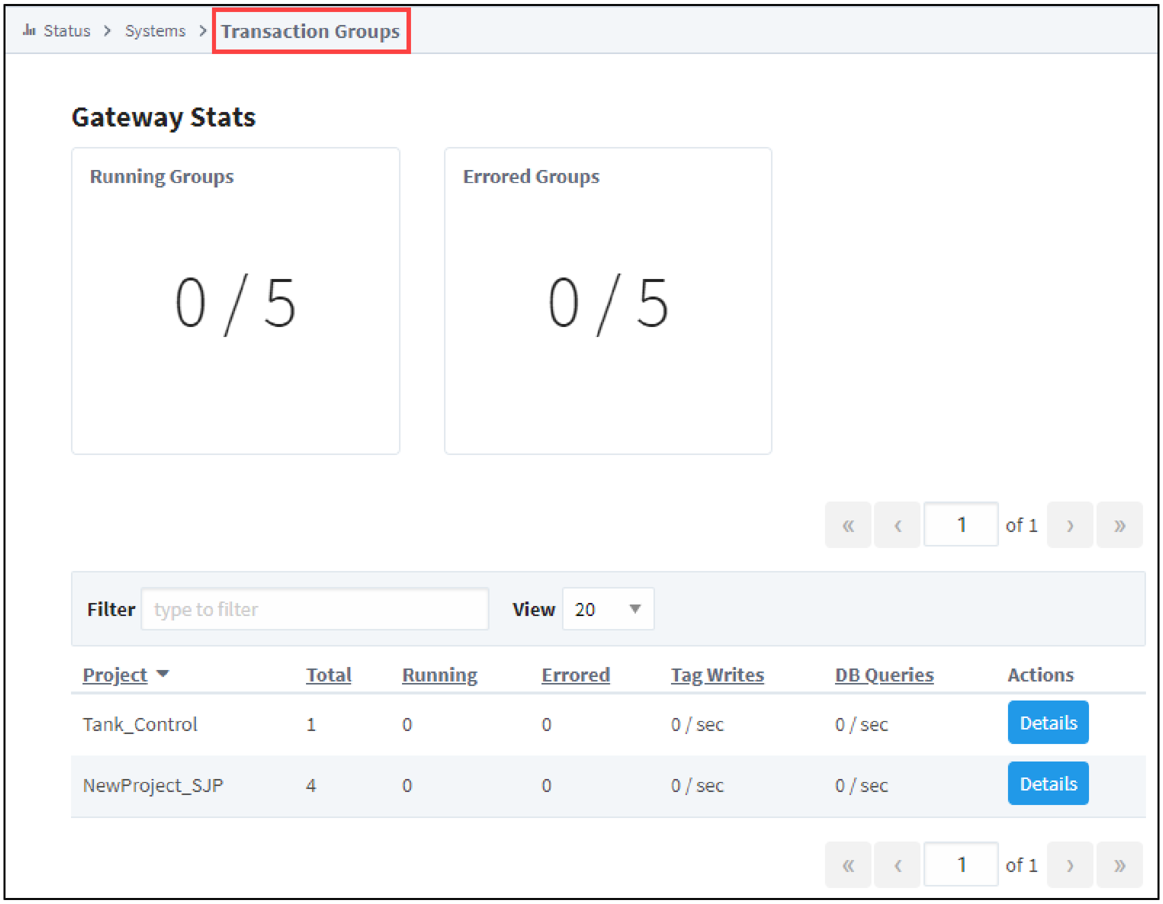
Task Name	Task Type	Repeats	Next Execution Start	Status
No items to display				

Recently Completed Tasks

Task Name	Task Type	Task Start	Agent	Task Result
Collect Backup (forced)	backup	25Mar2019 12:39:32	Ignition-2	✓ Success

Transaction Groups

The Transaction Groups page makes it easy to get a quick count of the currently running Transaction Groups as well as the currently errored groups. Below the counts is a list of Projects that contain Transaction Groups and some basic information like how many are running and how many database queries is it generating. Clicking the details button to the right for one of the Projects brings up more information about that Project's Groups. There are some counts on the number of Groups in various states, as well as a list of the groups. Any Groups that have problems can be clicked on to see what the error is to get a better idea of why the Group is unable to run.



Gateway Stats

Running Groups

0 / 5

Errored Groups

0 / 5

Filter: type to filter

View: 20

Project	Total	Running	Errored	Tag Writes	DB Queries	Actions
Tank_Control	1	0	0	0 / sec	0 / sec	Details
NewProject_SJP	4	0	0	0 / sec	0 / sec	Details

Related Topics ...

- Connections
- Diagnostics - Logs

Connections

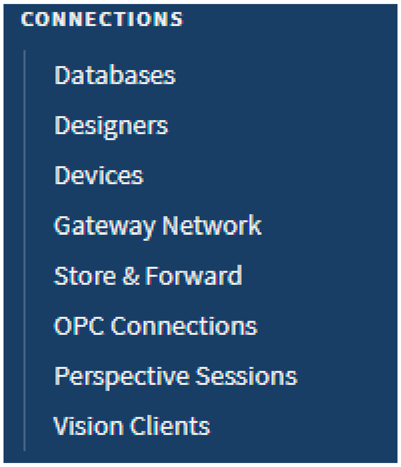
Status of Gateway Connections

On this page ...

- Status of Gateway Connections
- Log Activity

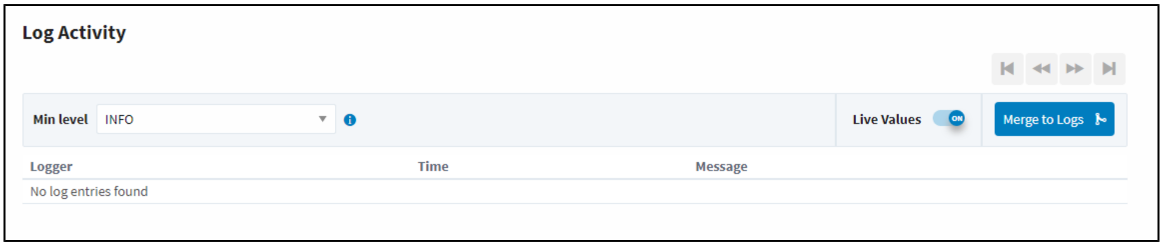
The Connections section of the Status tab on the Gateway Webpage contains information regarding the status of Gateway connections to external systems. The list of systems displayed under the Connections section is based on what modules are installed. It can include EAM Agents, Databases, Devices, Store and Forward, and the Gateway Network including other Gateways within the Gateway Network. The Connections section allows you to drill down and open up more specific information to easily find problems with anything connected to the Gateway, and even find crucial information about a faulty connection.

On the left side of the Status section of the Gateway Webpage, you'll see a list of all your connections. Click on any system to open a detailed page to see all the available options, and to get more detailed information. Some third party modules could potentially add sections that are not discussed in this manual.



Log Activity

Most of the systems in the Connection section have a Log Activity area somewhere within the details of the connection. This Log Activity area functions much like the Logs in the Diagnostics section of the Status page in that you see a list of all log messages depending on the logging level you are looking at as well as the filters you have setup. However, what makes the Log Activity areas in each Connections page unique is that they are filtered to only show loggers for that particular system. This helps you narrow down any potential problems by showing you only the information that pertains to the section you are looking at.



Logger	Time	Message
No log entries found		

Related Topics ...

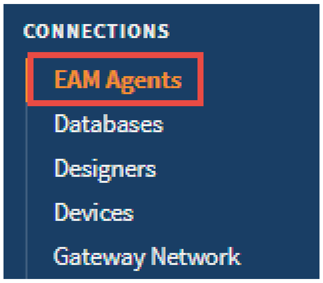
- Diagnostics - Logs

In This Section ...

Connections - EAM Agents

EAM Agents

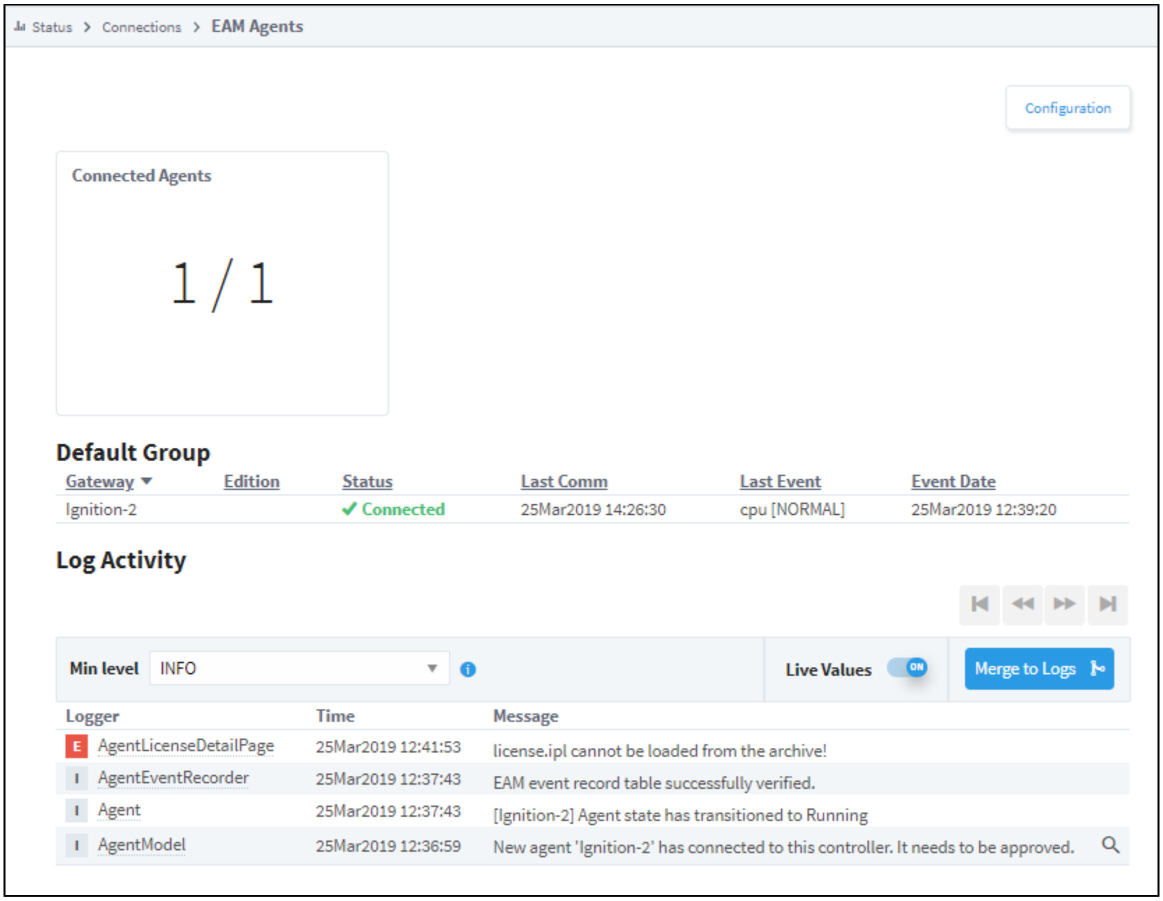
The EAM Agents page shows a list of all the currently configured Agents, including information on the connection status. Agents can be organized into groups allowing you to group agents by location or agent function. If no groups exist, all Agents will be in the Default Group.



EAM Agents Page

The EAM Agents page contains some useful information about all of your Agent connections.

Attribute	Description
EAM Agent Connections	
EAM Agents	Number of connected agents out of the number of configured agents.
Default Group and all groups	
Gateway	Name of the Agent.
Edition	Identifies a Standard vs Edge Gateway. Edge edition shows "edge," and if it's a Standard Gateway, this field is left blank.
Status	Current status of the Agent connection.
Last Comm	Date and time recorded for the last communication with the Agent.
Last Event	Name of the event last recorded with the Agent.
Event Date	Date and time recorded for the last occurring event with the Agent.
Log Activity	Shows any EAM Agent activity along with the Time the log was created and a brief message. You can find more of these same type of messages in the Gateway Logger which is found in the **Status** section under **Diagnostics > Logs**, and the Wrapper Logger file under **Program Files > Inductive Automation> Ignition > logs > wrapper.log**.



Connected Agents

1 / 1

Default Group

Gateway	Edition	Status	Last Comm	Last Event	Event Date
Ignition-2		✓ Connected	25Mar2019 14:26:30	cpu [NORMAL]	25Mar2019 12:39:20

Log Activity

Logger	Time	Message
E AgentLicenseDetailPage	25Mar2019 12:41:53	license.ipl cannot be loaded from the archive!
I AgentEventRecorder	25Mar2019 12:37:43	EAM event record table successfully verified.
I Agent	25Mar2019 12:37:43	[Ignition-2] Agent state has transitioned to Running
I AgentModel	25Mar2019 12:36:59	New agent 'Ignition-2' has connected to this controller. It needs to be approved.

Related Topics ...

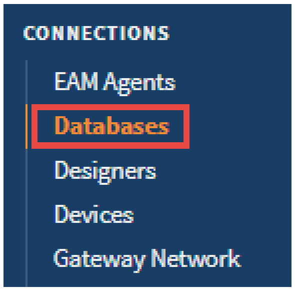
- Enterprise Administration
- Creating a Controller
- Adding an Agent

Connections - Databases

The Databases page shows a list of configured databases, and if they have a valid connection or not. Clicking on the **Details** button to the right of a connection will either show the full error if the connection is faulted, or it will bring you to a Details page for that database connection. On the Details page, you can easily see any active queries, long running queries, the number of queries a second that are running, as well as a trend showing the percentage of queries that completed in that time.

On this page

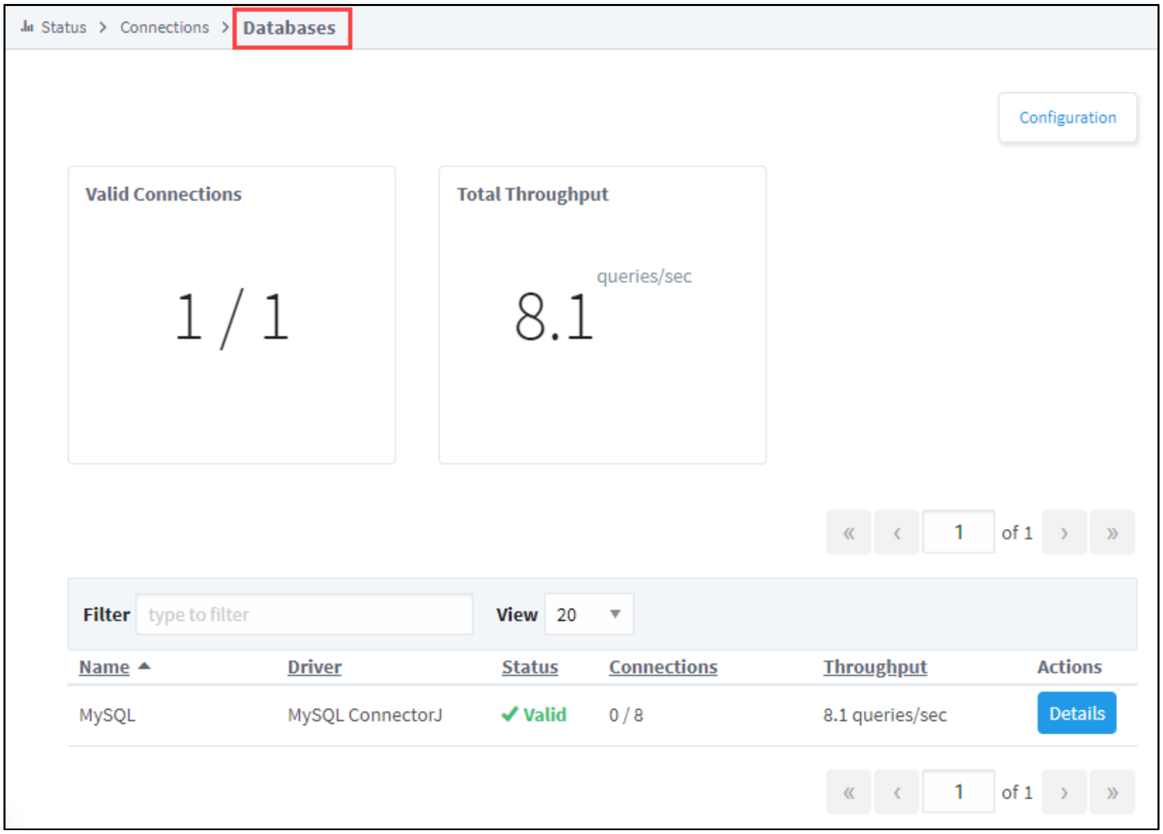
...



Databases Page

The main database page contains some useful information about all of your database connections.

Attributes	Description
Valid Connections	Number of valid connections.
Total Throughput	Number of queries and their status.
Name	Name of the query.
Driver	Name of the driver.
Status	Current status of the database connection.



Valid Connections 1 / 1

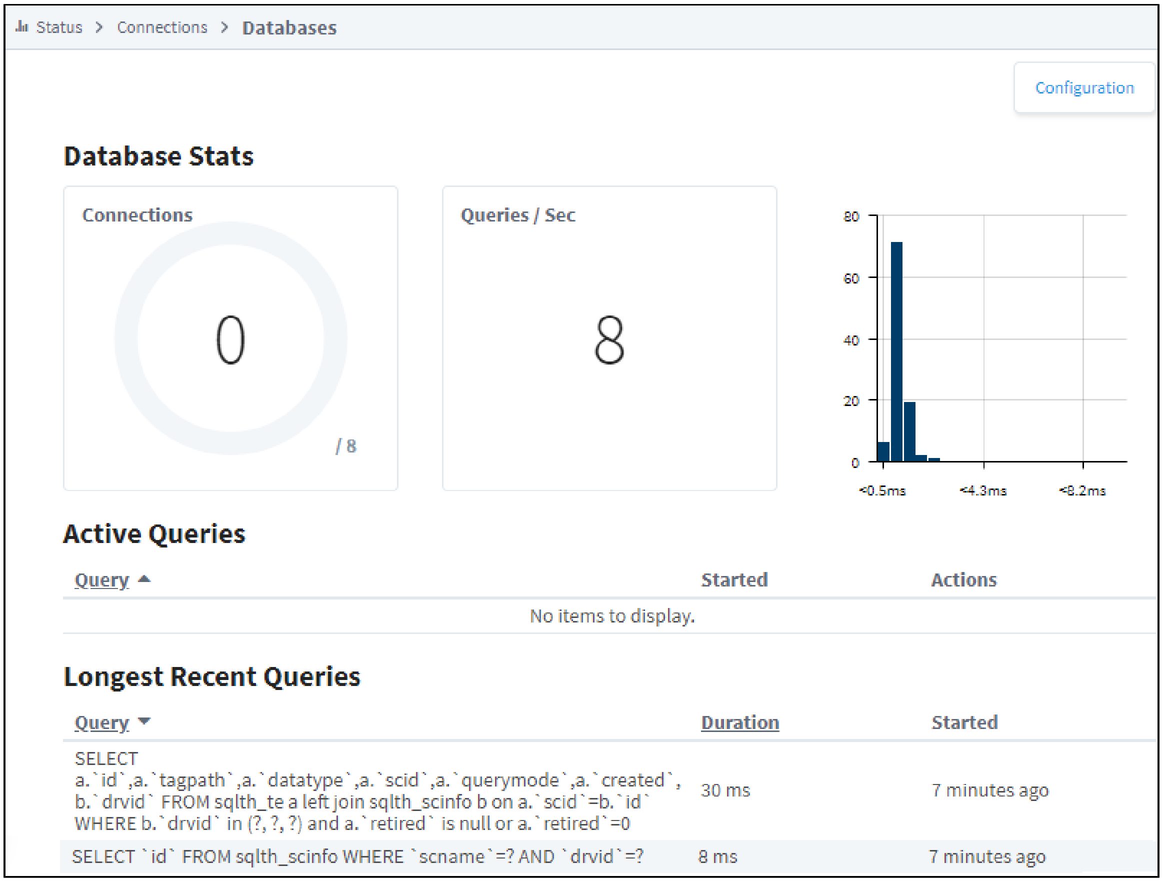
Total Throughput 8.1 queries/sec

Name	Driver	Status	Connections	Throughput	Actions
MySQL	MySQL ConnectorJ	✓ Valid	0 / 8	8.1 queries/sec	Details

Database Connection Details

The **Details** page for an individual connection provides more in depth information for that particular database connection.

Attribute	Description
Database Stats - Stats about the database connection	
Connections	Number of database connections out of configured databases.
Queries / Sec	Number of queries running per second along with a trend showing the percentage of queries that completed in that time.
Active Queries - A list of currently active queries	
Query	Currently running queries.
Started	When the query was started.
Actions	The ability to cancel a query.
Longest Recent Queries - A list of the longest running queries.	
Query	Displays the actual query.
Duration	Amount of time the query ran.
Started	When the query started running.



Database Stats

Active Queries

Query ▲	Started	Actions
No items to display.		

Longest Recent Queries

Query ▼	Duration	Started
SELECT a.`id`,a.`tagpath`,a.`datatype`,a.`scid`,a.`querymode`,a.`created`, b.`drvid` FROM sqlth_te a left join sqlth_scinfo b on a.`scid`=b.`id` WHERE b.`drvid` in (?, ?, ?) and a.`retired` is null or a.`retired`=0	30 ms	7 minutes ago
SELECT `id` FROM sqlth_scinfo WHERE `scname`=? AND `drvid`=?	8 ms	7 minutes ago

Related Topics ...

- Database Connections

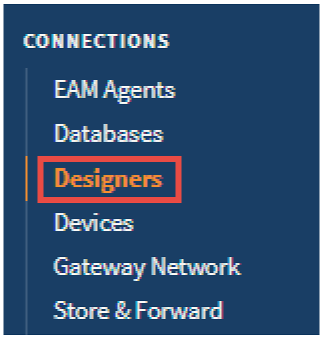
Connections - Designers

Designers

On this page

...

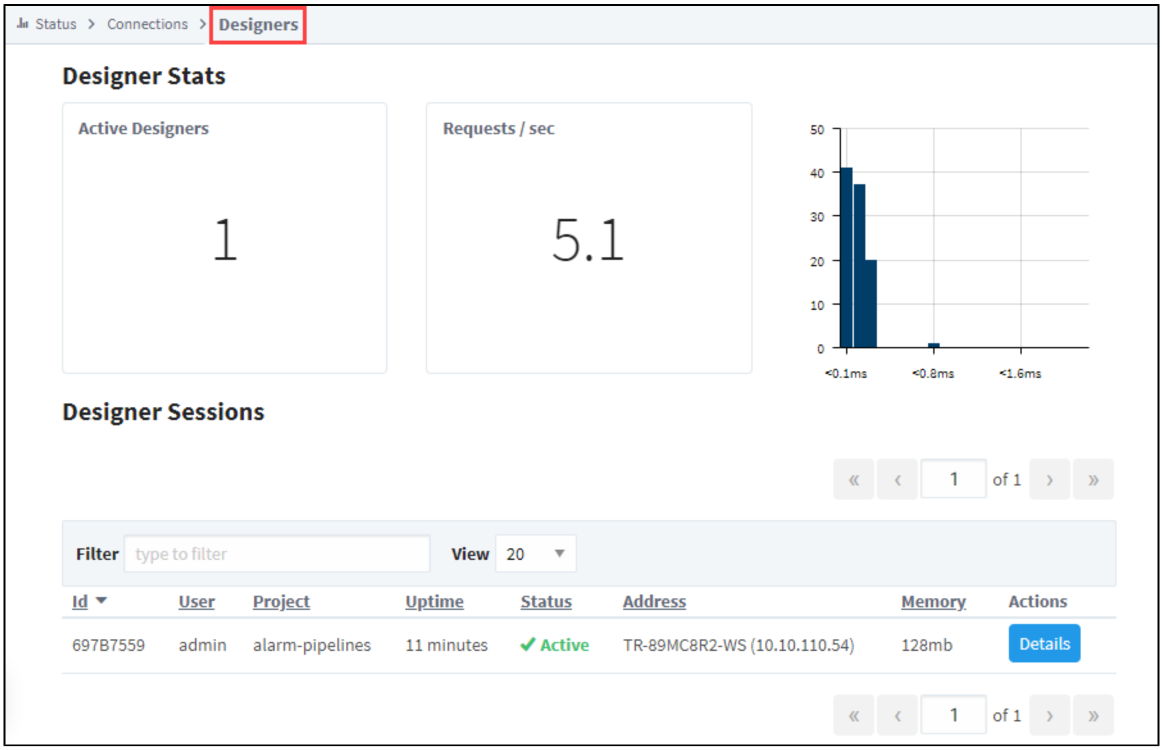
The Designers page displays information on currently running designer sessions. All the open designers are displayed on the page along with some basic information about each session, such as what user is logged into each designer session and the project they are currently working on. Clicking on the **Details** button to the right of a designer session will display more information about that particular designer. On the Details page, you can see session information, as well as what designer locks the session currently has set. Locks are when a designer is working on a particular page or set of pages, the system places a lock on those resources (i.e., window, pipelines, etc.,) to prevent other designers from working on the same resource. There is also a log at the bottom of the page displaying any errors pertaining to that designer session.



Designers Page

The main Designers page has a list of all currently running designer sessions, and some basic information about all of them.

Attribute	Description
Designer Stats	
Active Designers	Number of active Designer connections.
Requests / Sec	Number of requests running per second along with a trend showing the percentage of requests that completed in that time.
Designer Sessions	
Filter	Search criteria to filter for specific designer sessions.
View	The number of designer sessions to preview.
Id	Designer session id number.
User	Name of the user logged into the Designer.
Project	Name of project the user is currently working on.
Uptime	Amount of time the user is logged into the designer session.
Status	If the Designer is actively connected or not.
Address	The IP Address and name of the computer running the designer.
Memory	Current memory usage for the session.



Status > Connections > Designers

Designer Stats

Active Designers: 1

Requests / sec: 5.1

Designer Sessions

Filter: type to filter

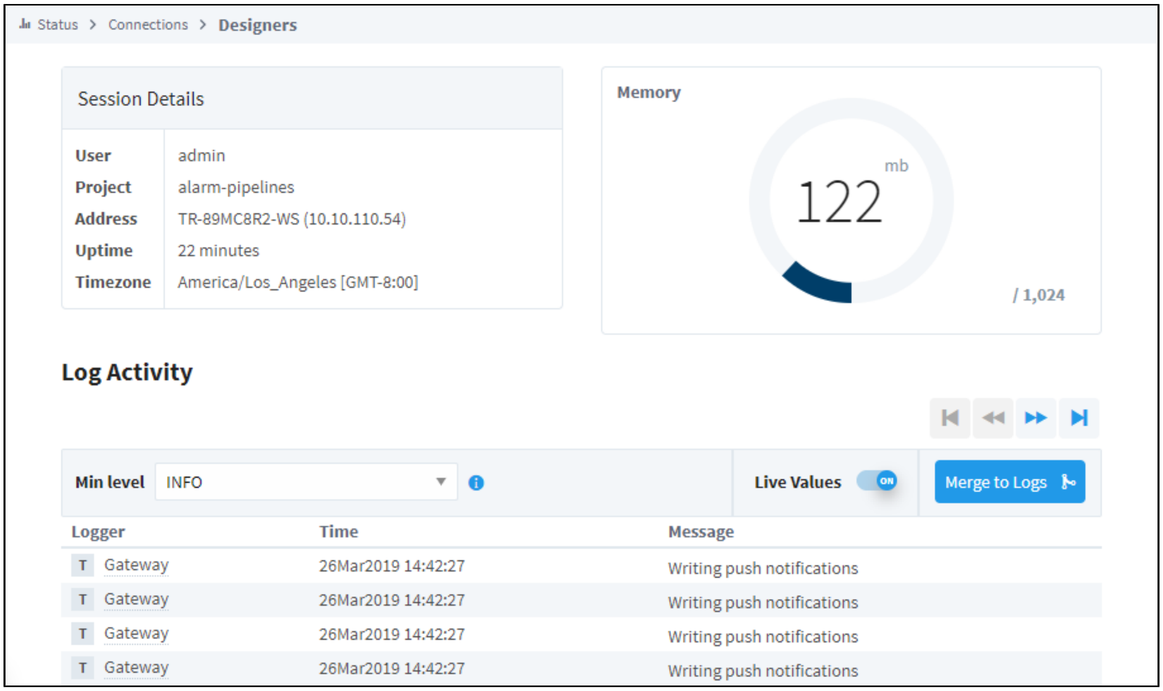
View: 20

Id	User	Project	Uptime	Status	Address	Memory	Actions
697B7559	admin	alarm-pipelines	11 minutes	✓ Active	TR-89MC8R2-WS (10.10.110.54)	128mb	Details

Designer Details

The Designer Details page shows more in depth information about that particular designer session, with the most important being the list of Designer Locks. The locks are project resources that the designer session is using. Because that designer session is using those project resources, no other designers will be able to access those resources, to prevent two designers from working on the same thing.

Attribute	Description
Session Details	
User	Name of the user logged into the Designer.
Project	Name of project the user is currently working on.
Address	The IP Address and name of the computer running the designer.
Uptime	Amount of time the user is logged into the designer session.
Memory	Current memory usage for the session.
Timezone	Local time of the user.
Log Activity	
Min Level	Dropdown menu with options Info, Debug, and Trace.
Live Values	Toggle switch to turn live values on or off.
Logger	Name of the logger that describes the context of the message.
Time	Time of log.
Message	Message for the log.



Status > Connections > **Designers**

Session Details

User	admin
Project	alarm-pipelines
Address	TR-89MC8R2-WS (10.10.110.54)
Uptime	22 minutes
Timezone	America/Los_Angeles [GMT-8:00]

Log Activity

Min level: INFO

Logger	Time	Message
T Gateway	26Mar2019 14:42:27	Writing push notifications
T Gateway	26Mar2019 14:42:27	Writing push notifications
T Gateway	26Mar2019 14:42:27	Writing push notifications
T Gateway	26Mar2019 14:42:27	Writing push notifications

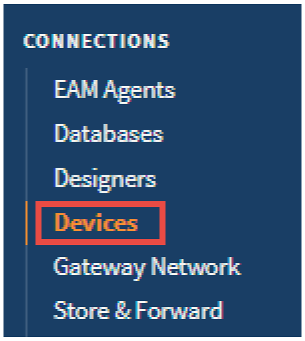
Related Topics ...

- Designer

Connections - Devices

Devices

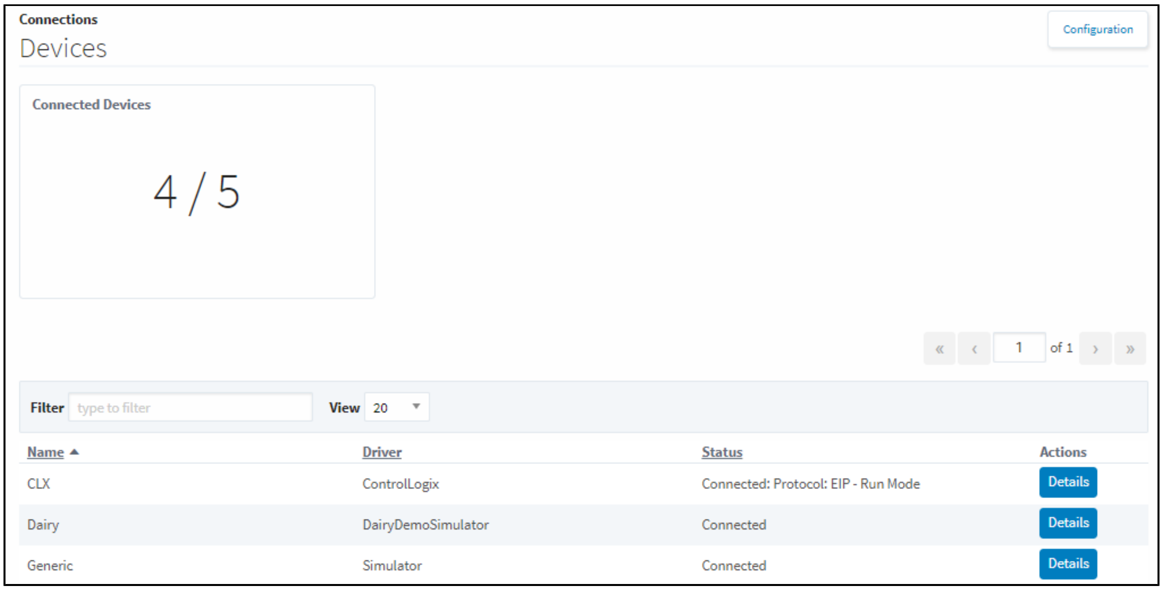
The Devices page lists the currently configured OPC UA devices, and lets you know which are connected and which have a faulty connection. From here, you can drill into a device connection to see how many Tags Ignition is requesting from the device, along with how often it is requesting them. This information can be used to determine if we are overloading the device with too many requests too quickly, or if we can request more from our device.



Devices Page

The main Devices page lists out all OPC UA devices, as well as how many currently have a valid connection. Note: this will only show the devices connected through an Ignition device connection. For information about devices connected through other OPC Servers, see those programs.

Attribue	Description
Connected Devices	Number of devices connected out of configured devices.
Name	Name of the device.
Driver	Name of the device driver.
Status	Current device status.



Connected Devices

4 / 5

Name ▲	Driver	Status	Actions
CLX	ControlLogix	Connected: Protocol: EIP - Run Mode	Details
Dairy	DairyDemoSimulator	Connected	Details
Generic	Simulator	Connected	Details

Device Details

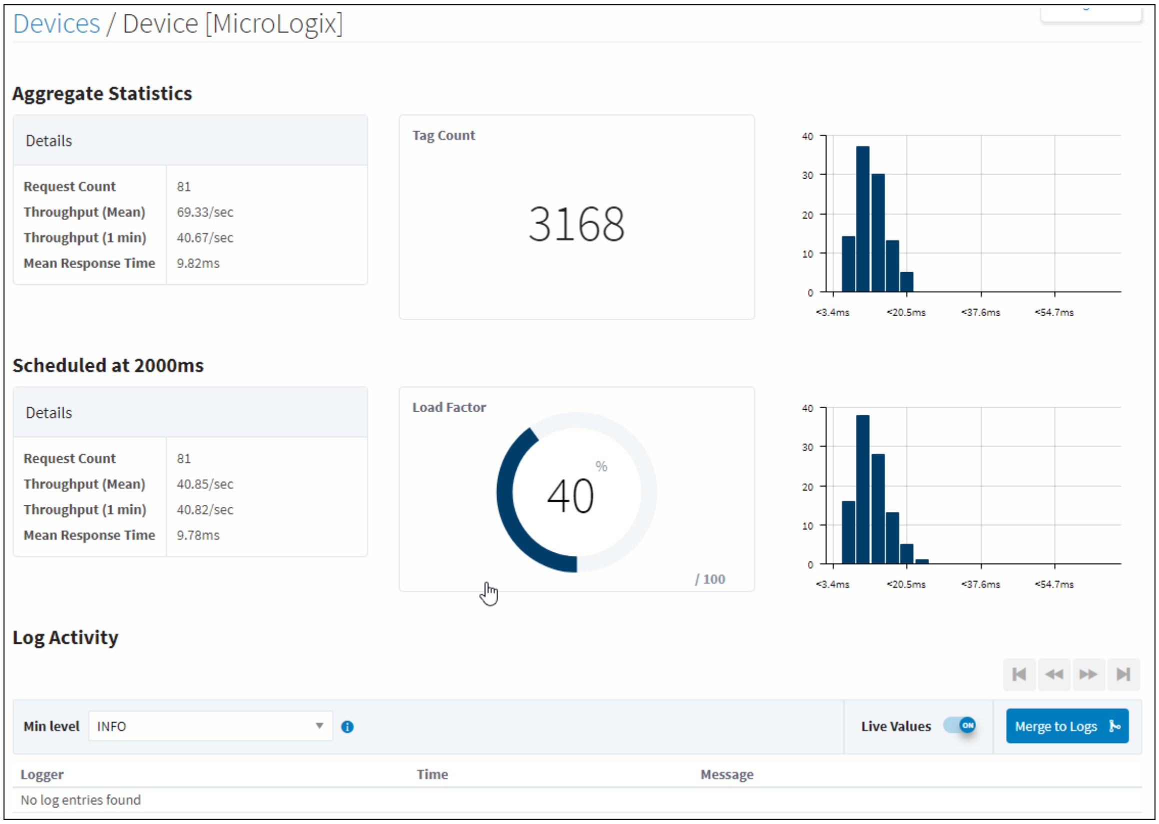
Clicking the **Details** button to the right will display a diagnostics page for that device which provides a lot of useful metrics like a Tag count. The log at the bottom of the page will display recent events that pertain to that particular device.

The Device Details page also lists statistics to help determine if the device is overloaded with requests. There are values for each Tag Group that are configured on Tags from the specified device, as well as aggregate statistics which pull from all Tag Groups to get an average for the device. The four values are listed in the table below.

Attribute	Description
Aggregate Statics	
Request Count	Tracks the number of requests that are coming in from the device, each request consisting of multiple Tags. How many Tags are in a request can vary between devices, which Tags are being requested, and how often they are requested.
Throughput (Mean)	Average amount of requests that come through per second since the device was last started. If the device connection is edited and saved, this will cause the device connection to reinitialize and this value will be reset.
Throughput (1 min)	Average amount of requests that come through per second for the last minute.
Mean Response Time	Average time it takes for Ignition to get a response from the device. This number should be an average of the graph on the right of the page.

So based on the statistics above, we can determine if we are overloading our device by requesting too many Tags too quickly. We simply need to take the throughput and use that to determine how many total requests come in per Tag Group execution. If this is at or above the request count, then our device can keep up with the numbers we are requesting. If it is below the request count, then the number of requests we are making to the device aren't coming in as fast as we want them, which can lead to poor device performance or bad values.

For example, take the image below. Scheduled at 2000ms, we have 81 requests. Our average throughput is about 40.8 requests a second. This means that every two seconds, we get 81.6 requests. Since we are making 81 requests every two seconds, this is exactly what we would expect to be getting from this device.



Devices / Device [MicroLogix]

Aggregate Statistics

Details

Request Count	81
Throughput (Mean)	69.33/sec
Throughput (1 min)	40.67/sec
Mean Response Time	9.82ms

Tag Count

3168

Scheduled at 2000ms

Details

Request Count	81
Throughput (Mean)	40.85/sec
Throughput (1 min)	40.82/sec
Mean Response Time	9.78ms

Load Factor

40 %

/ 100

Log Activity

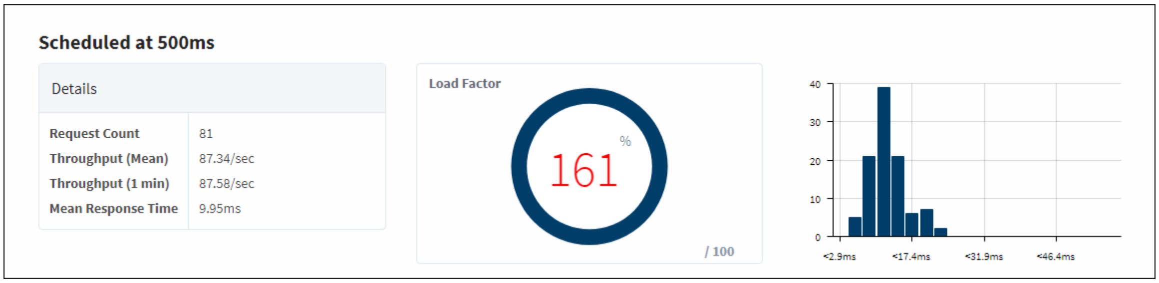
Min level INFO

Live Values ON

Logger	Time	Message
No log entries found		

Overloaded Device

However, if we altered the Tag Group on the same device to 500ms, we see a very different picture. The request count is still 81, but because we are requesting values every 500ms, we should see about 162 requests coming through every second. The throughput values only show us getting about 87.5 requests per second, which is about half of what we actually are asking for. This would indicate that our device is overloaded, which can also be seen in the Load Factor. Our Tags would not be updating as quickly as we would like, which could lead to bad Tag values.



Scheduled at 500ms

Details

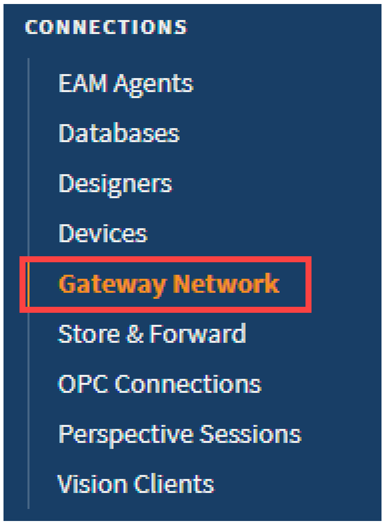
Request Count	81
Throughput (Mean)	87.34/sec
Throughput (1 min)	87.58/sec
Mean Response Time	9.95ms

Related Topics ...

- OPC UA
- Allen Bradley Ethernet
- Modbus
- Siemens

Connections - Gateway Network

The Gateway Network Status page is designed to give a quick overview of the status of all Gateways within the Gateway Network. If a Gateway connection is faulted, the status message in red can be selected to see the error that pertains to why it is faulted. Any Gateway connections with a status of 'Connected' can be drilled into by clicking the **Details** button to the right. On the Details page, metrics for the selected Gateway connection are displayed, giving an idea of the rate of data transfer between the two Gateways, as well as a list of recent connection events.



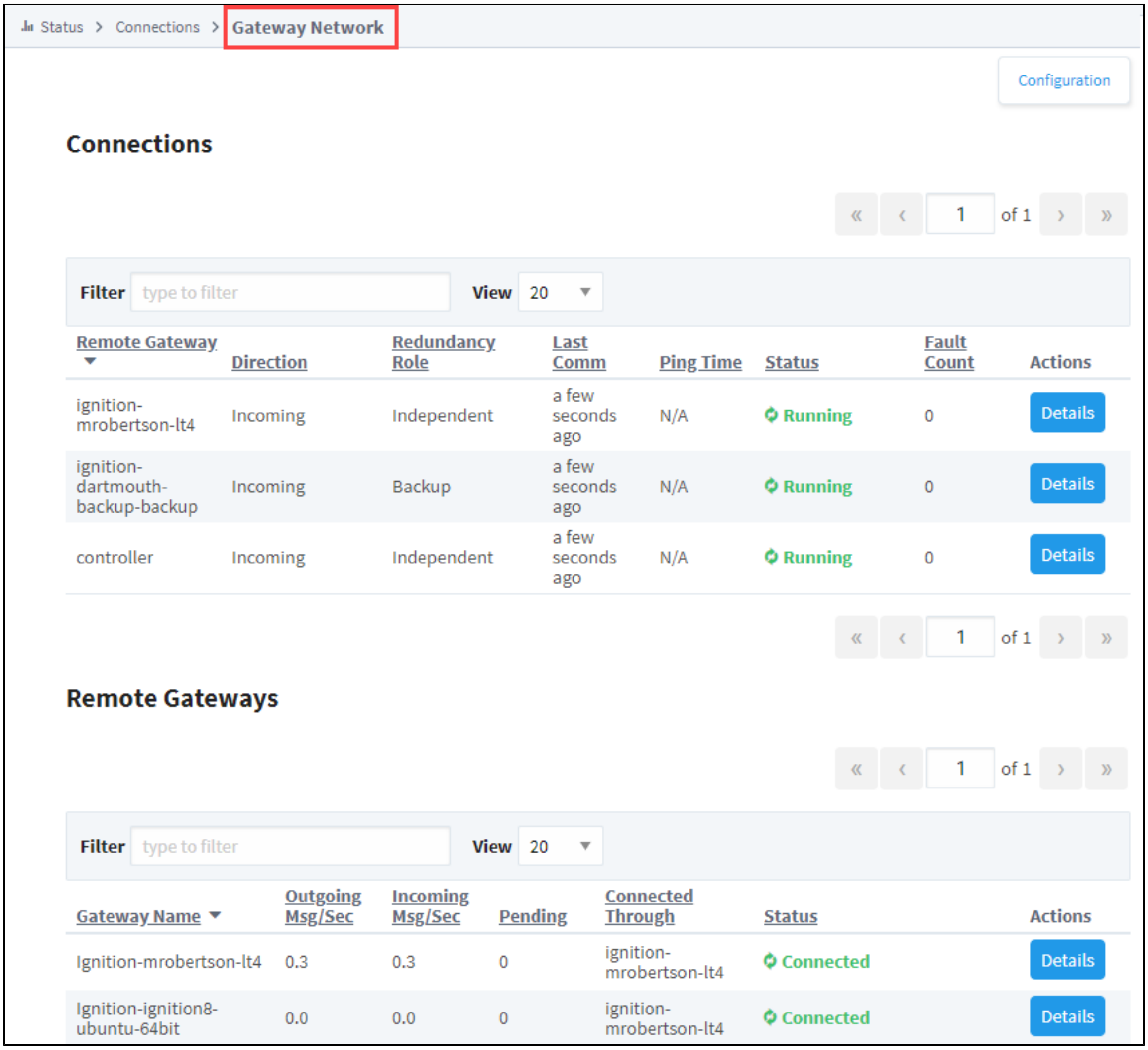
On this page ...

Gateway Network Page

The main Gateway Network page displays a list of all current Gateway Network connections both incoming and outgoing, as well a list of all Remote Gateways that the Local Gateway can see both from its Gateway Network connections and through proxy connections. Each list has some basic information along with the ability to see more details on a specific Gateway Network connection or a Remote Gateway.

Attribute	Description
Gateway Network Connections	
Remote Gateway	Name of the Remote Gateway connection.
Direction	The direction of the Gateway Network connection. Can either be Incoming or Outgoing.
Redundancy Role	The redundancy role of the Gateway. Can either be Independent, Backup, or Master.
Last Comm	The time of the last communication with the Gateway.
Ping Time	Reaction time of Gateway connection. How fast you get a response after you've sent out a request.
Status	Current state of the Gateway Network Connection.
Fault Count	Number of times the connection has faulted since the Gateway has been started.
Remote Gateways	
Gateway Name	Name of the Remote Gateway.
Outgoing Msg/Sec	The number of outgoing messages per second.
Incoming Msg/Sec	The number of incoming messages per second.

Pending	Number of messages pending in a queue that are waiting to be dispatched to the Gateway Network connection.
Connected Through	How the Gateway is connected to the Remote Gateway.
Status	Current state of the Remote Gateway Connection.



Connections

Filter type to filter View 20

Remote Gateway ▼	Direction	Redundancy Role	Last Comm	Ping Time	Status	Fault Count	Actions
ignition-mrobertson-lt4	Incoming	Independent	a few seconds ago	N/A	Running	0	Details
ignition-dartmouth-backup-backup	Incoming	Backup	a few seconds ago	N/A	Running	0	Details
controller	Incoming	Independent	a few seconds ago	N/A	Running	0	Details

Remote Gateways

Filter type to filter View 20

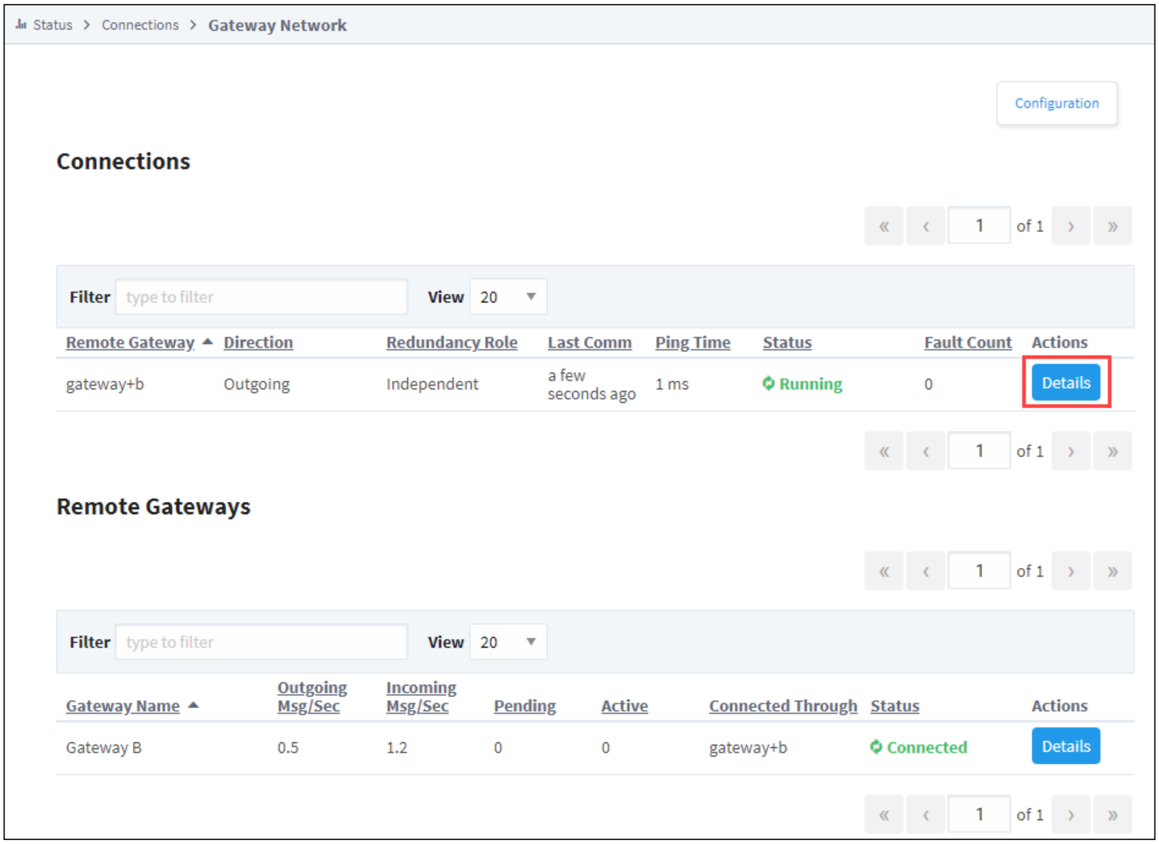
Gateway Name ▼	Outgoing Msg/Sec	Incoming Msg/Sec	Pending	Connected Through	Status	Actions
Ignition-mrobertson-lt4	0.3	0.3	0	ignition-mrobertson-lt4	Connected	Details
Ignition-ignition8-ubuntu-64bit	0.0	0.0	0	ignition-mrobertson-lt4	Connected	Details

Significant updates have been made to the Gateway Network starting in release 8.0.14. Use the links below to view the Gateway Network information for the version of Ignition that you are using.

Gateway Area Network for Ignition 8.0.14 and Earlier

Connection Details 8.0.14 and Earlier

From the Gateway Network Status page, clicking on the Details button to the right of a Gateway Network connection gives more details about that specific connection.



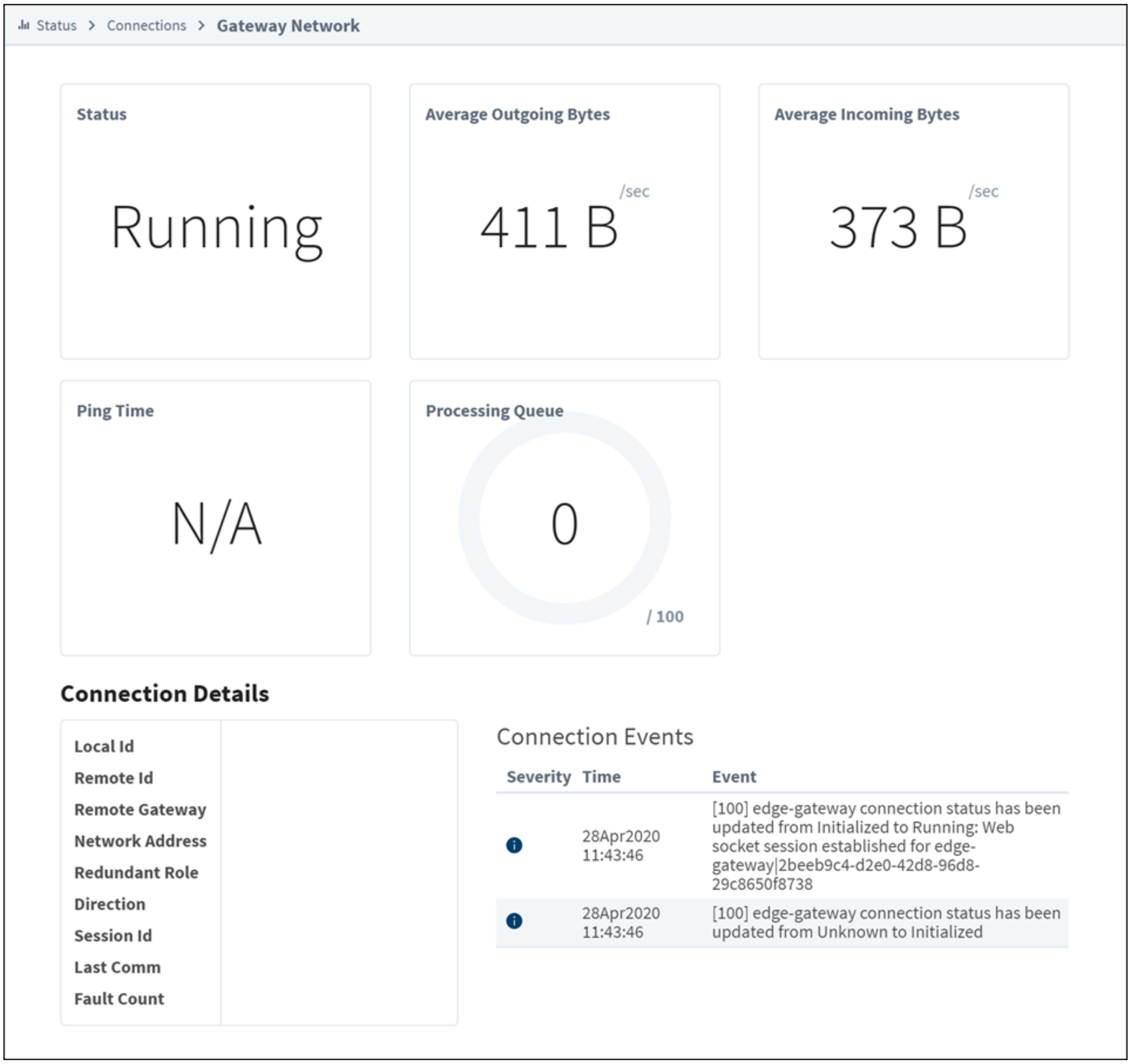
Connections

Remote Gateway ▲	Direction	Redundancy Role	Last Comm	Ping Time	Status	Fault Count	Actions
gateway+b	Outgoing	Independent	a few seconds ago	1 ms	Running	0	Details

Remote Gateways

Gateway Name ▲	Outgoing Msg/Sec	Incoming Msg/Sec	Pending	Active	Connected Through	Status	Actions
Gateway B	0.5	1.2	0	0	gateway+b	Connected	Details

Below, we can see the connection details for a gateway network connection to Gateway B:



Status > Connections > Gateway Network

Status

Running

Average Outgoing Bytes

411 B /sec

Average Incoming Bytes

373 B /sec

Ping Time

N/A

Processing Queue

0 / 100

Connection Details

Local Id	
Remote Id	
Remote Gateway	
Network Address	
Redundant Role	
Direction	
Session Id	
Last Comm	
Fault Count	

Connection Events

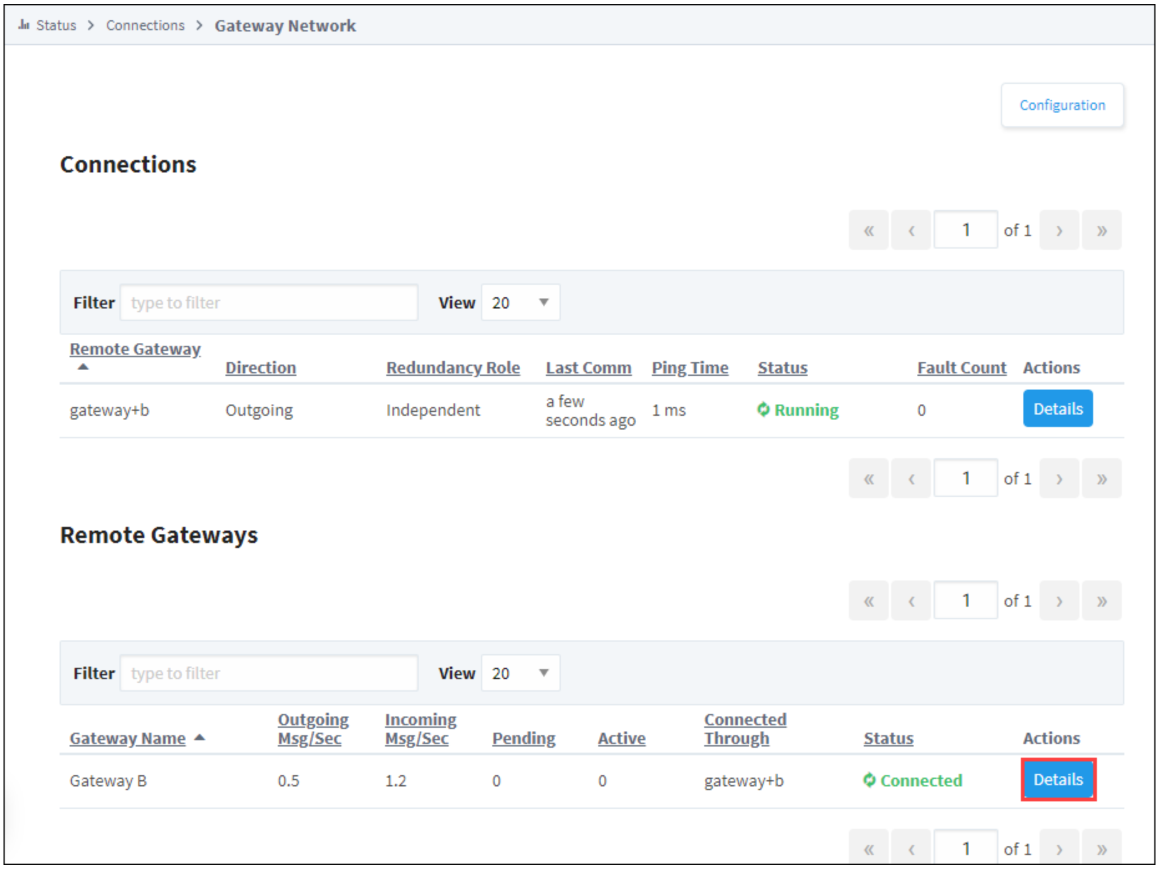
Severity	Time	Event
●	28Apr2020 11:43:46	[100] edge-gateway connection status has been updated from Initialized to Running: Web socket session established for edge-gateway|2beeb9c4-d2e0-42d8-96d8-29c8650f8738
●	28Apr2020 11:43:46	[100] edge-gateway connection status has been updated from Unknown to Initialized

Attribute	Description
Status	Current state of the Gateway Network Connection.
Average Outgoing Bytes	Average bytes of data going from the Local Gateway to the Remote Gateway per second.
Average Incoming Bytes	Average bytes of data going from the Remote Gateway to the Local Gateway per second.
Ping Time	Reaction time of Gateway connection. How fast you get a response after you've sent out a request.
Pending Uploads	This feature was removed from Ignition in version 8.0.13 The number of uploads pending from the Gateway Network connection.
Send Threads	The following feature is new in Ignition version **8.0.13** Click here to check out the other new features A thread that is used by the Gateway Network to process outgoing messages from one Gateway to another.

Pending Downloads	This feature was removed from Ignition in version 8.0.13 The number of downloads pending from the Gateway Network connection.
Receive Threads	The following feature is new in Ignition version **8.0.13** Click here to check out the other new features A thread that is used by the Gateway Network to process incoming messages sent from one Gateway to another.
Local Id	Id of the local Gateway.
Remote Id	Id of the remote Gateway.
Remote Gateway	Name of the Gateway on the Gateway network.
Network Address	Physical address used to communicate with all devices on the Gateway network.
Redundant Role	The redundancy role of the Gateway. Can either be Independent, Backup, or Master.
Direction	The direction of the Gateway Network connection. Can either be Incoming or Outgoing.
Session Id	Gateway connection session Id number.
Last Comm	The time of the last communication with the Gateway.
Fault Count	Number of errors.
Connection Events	Displays a list of recent connection events.

Remote Gateway Details

Users can also see Remote Gateway Details by clicking on the Details button of the Remote Gateway of interest as below:


Status > Connections > Gateway Network
Configuration
« ‹ 1 of 1 › »
Filter type to filter View 20

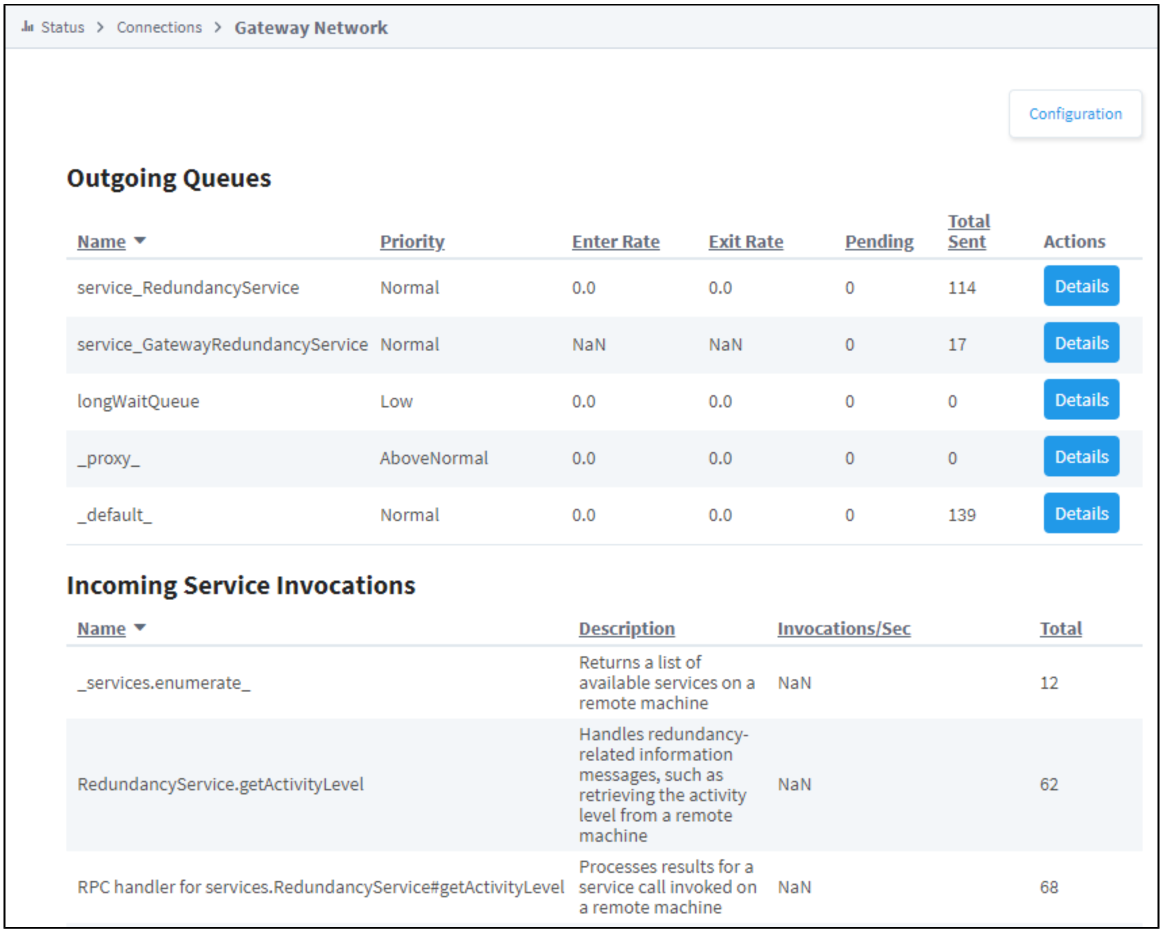
Connections

Remote Gateway ▲	Direction	Redundancy Role	Last Comm	Ping Time	Status	Fault Count	Actions
gateway+b	Outgoing	Independent	a few seconds ago	1 ms	Running	0	Details

Remote Gateways

Gateway Name ▲	Outgoing Msg/Sec	Incoming Msg/Sec	Pending	Active	Connected Through	Status	Actions
Gateway B	0.5	1.2	0	0	gateway+b	Connected	Details

The Remote Gateway Details page offers information regarding outgoing message queues. Ignition's Gateway Network system shares information across Gateways using threads to send and receive information. This information is distributed via messages that can be tracked via the queue statistics below.



Outgoing Queues

Name	Priority	Enter Rate	Exit Rate	Pending	Total Sent	Actions
service_RedundancyService	Normal	0.0	0.0	0	114	Details
service_GatewayRedundancyService	Normal	NaN	NaN	0	17	Details
longWaitQueue	Low	0.0	0.0	0	0	Details
proxy	AboveNormal	0.0	0.0	0	0	Details
default	Normal	0.0	0.0	0	139	Details

Incoming Service Invocations

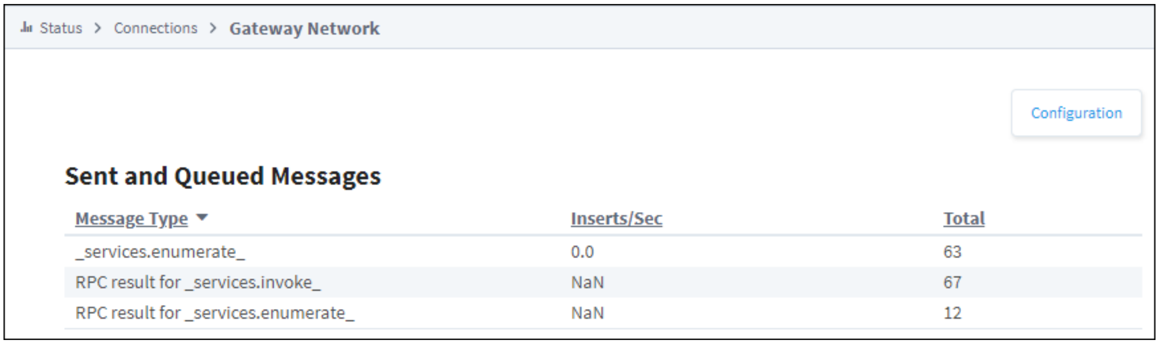
Name	Description	Invocations/Sec	Total
services.enumerate	Returns a list of available services on a remote machine	NaN	12
RedundancyService.getActivityLevel	Handles redundancy-related information messages, such as retrieving the activity level from a remote machine	NaN	62
RPC handler for services.RedundancyService#getActivityLevel	Processes results for a service call invoked on a remote machine	NaN	68

Attributes	Description
Outgoing Queues	
Name	The name of the queue.
Priority	The priority the queue takes among other queues.
Enter Rate	The rate at which messages enter the queue.
Exit Rate	The rate at which messages exit the queue.
Pending	The number of pending messages in that queue.
Total Sent	The number of messages sent out through that queue.
Incoming Service Invocations	
Name	The name of the incoming invocation.
Description	A description of what the invocation is asking of the local Gateway.
Invocations/Sec	The number of that type of invocation per second.
Total	The total number of that type of invocations.

Outgoing Queues Details

Click on the Details button to the right of a Outgoing Queue to bring up details on the messages being sent from that queue. It lists out each message type, as well as the number of each message type per second that is being sent, as well as the total number of that message being sent.

The Sent and Queued Messages screens lists out each message type, the number of each message type per second that is being sent, and the total number of that message being sent.



Sent and Queued Messages

Message Type	Inserts/Sec	Total
services.enumerate	0.0	63
RPC result for _services.invoke_	NaN	67
RPC result for _services.enumerate_	NaN	12

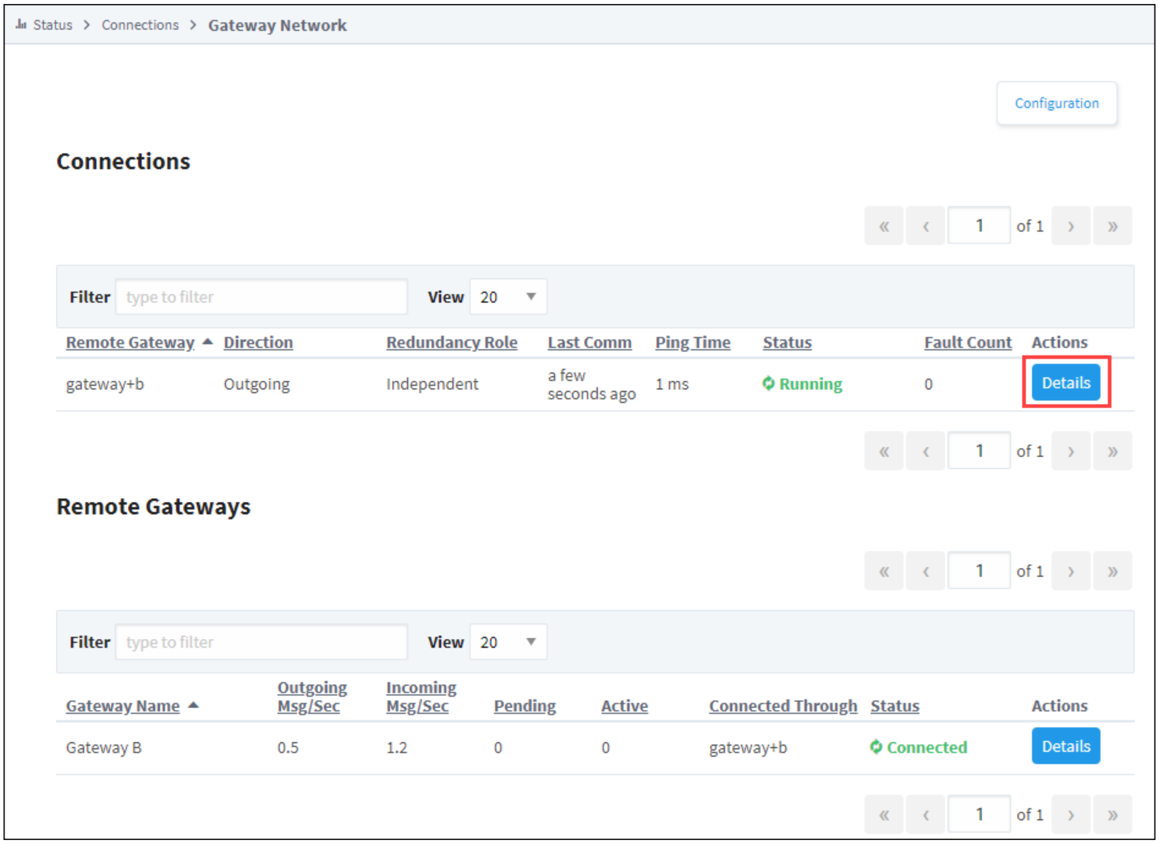
Attributes	Description
Message Type	The name and type of message.
Inserts/Sec	The number of messages per second.
Total	The rate at which messages enter the queue.

Gateway Area Network for Ignition 8.0.15 and Later

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

Ignition's Gateway Network system shares information across Gateways using threads to send and receive information. For example, if you have a Remote Tag Provider configured between Gateway A and Gateway B, messages containing live tag information will be sent between these two Gateways using threads to send and receive live tag data. Similarly, a Remote Historical Tag Provider will send/receive messages with historical tag data between Gateway A and Gateway B using this same set of threads. Each Ignition sub system that uses the Gateway Network will utilize these threads in some way.

Ignition's Gateway Network also has a queue associated with each Ignition subsystem. These queues allow for Ignition to have a way to prioritize which sub system should have access to a send or receive thread. Prioritization here is especially important because if all send threads are in use, messages cannot be sent between two Gateways. As of 8.0.15, the Gateway Network page has been expanded to include additional information that will help better monitor Gateway interactions.


Status > Connections > Gateway Network
Configuration
« ‹ 1 of 1 › »
Filter type to filter View 20
Details

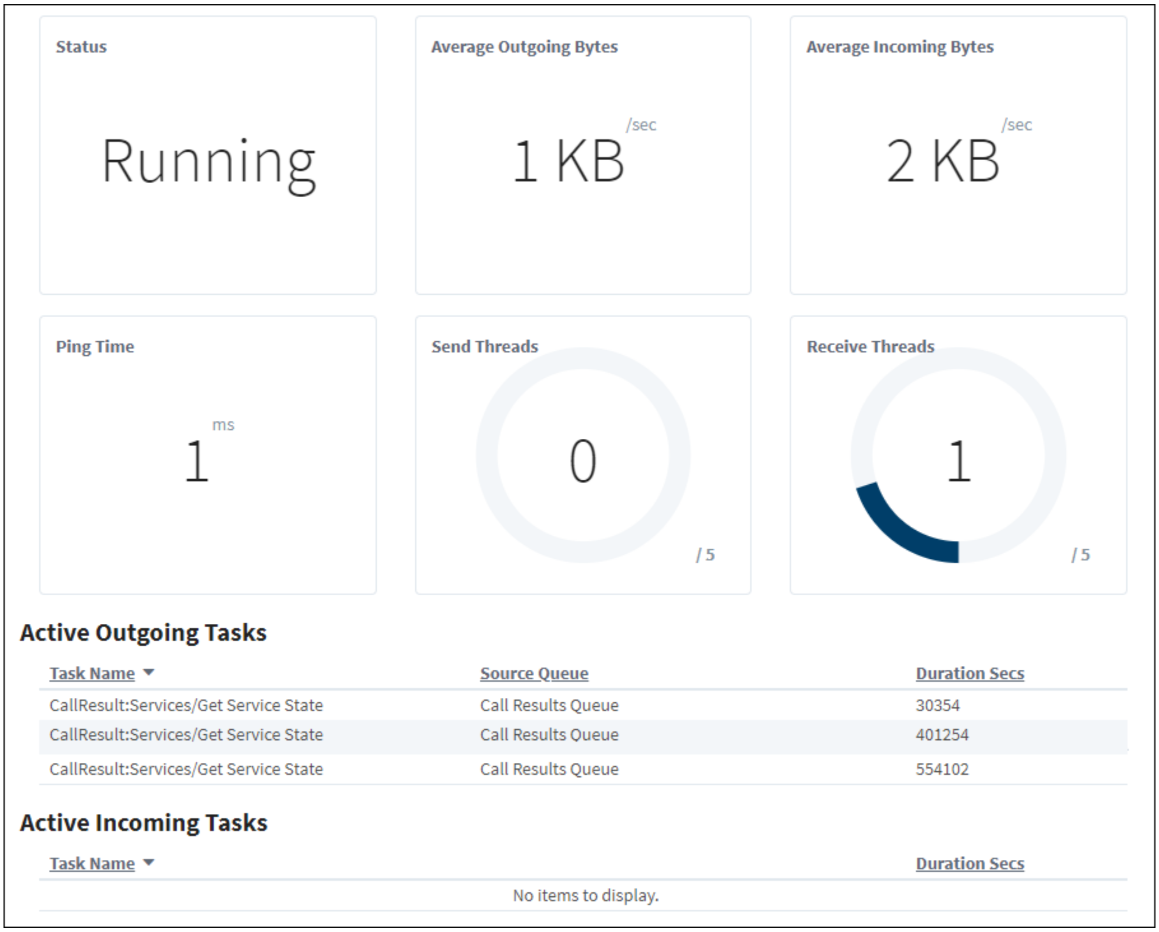
Connections

Remote Gateway ▲	Direction	Redundancy Role	Last Comm	Ping Time	Status	Fault Count	Actions
gateway+b	Outgoing	Independent	a few seconds ago	1 ms	Running	0	Details

Remote Gateways

Gateway Name ▲	Outgoing Msg/Sec	Incoming Msg/Sec	Pending	Active	Connected Through	Status	Actions
Gateway B	0.5	1.2	0	0	gateway+b	Connected	Details

The image above shows the basic Gateway Network Connection Status page where we can see there is an outgoing Gateway Network connection from Gateway A to Gateway B. Pressing the Details button will bring up detailed information about this connection as below:



Status
Running

Average Outgoing Bytes
1 KB /sec

Average Incoming Bytes
2 KB /sec

Ping Time
1 ms

Send Threads
0 / 5

Receive Threads
1 / 5

Active Outgoing Tasks

Task Name	Source Queue	Duration Secs
CallResult:Services/Get Service State	Call Results Queue	30354
CallResult:Services/Get Service State	Call Results Queue	401254
CallResult:Services/Get Service State	Call Results Queue	554102

Active Incoming Tasks

Task Name	Duration Secs
No items to display.	

Attribute	Description
Status	Current state of the Gateway Network Connection.
Average Outgoing Bytes	Average bytes of data going from the Local Gateway to the Remote Gateway per second.
Average Incoming Bytes	Average bytes of data going from the Remote Gateway to the Local Gateway per second.
Ping Time	Reaction time of Gateway connection. How fast you get a response after you've sent out a request.
Active	Number of messages being actively processed by the gateway network connection.
Send Threads	The following feature is new in Ignition version **8.0.13** Click here to check out the other new features A thread that is used by the Gateway Network to upload messages from one gateway to another.
Receive Threads	The following feature is new in Ignition version **8.0.13** Click here to check out the other new features A thread that is used by the Gateway Network to download messages from one gateway to another.
Local Id	Id of the local Gateway.
Remote Id	Id of the remote Gateway.
Remote Gateway	Name of the Gateway on the Gateway network.
Network Address	Physical address used to communicate with all devices on the Gateway network.
Redundant Role	The redundancy role of the Gateway. Can either be Independent, Backup, or Master.
Direction	The direction of the Gateway Network connection. Can either be Incoming or Outgoing.

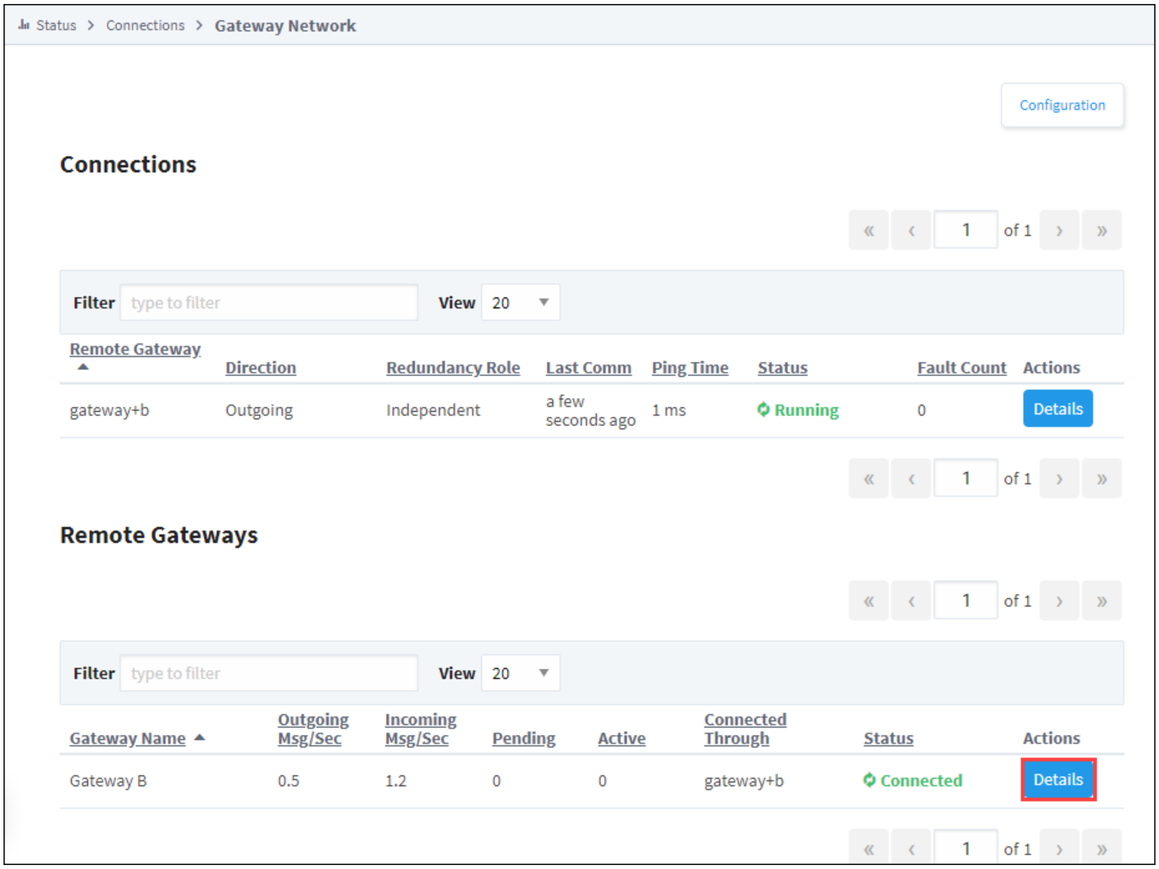
Session Id	Gateway connection session Id number. Connected gateways use the same session Id on both gateways.
Last Comm	The time of the last communication with the Gateway.
Fault Count	Number of times the connection has faulted since the Gateway has been started.
Connection Events	Displays a list of recent connection events.

Active Outgoing and Incoming Tasks List

Attribute	Description
Task Name	Name of the task that is using a thread.
Source Queue	The Ignition sub system queue that dispatched this task.
Duration Secs	This is how long in seconds it takes for a task to be performed.

Gateway Network Statistics

In addition to showing live thread and task information, users can also keep track of Gateway Network Statistics. By clicking on the Details button below, the Gateway Network Statistics Page appears for Gateway B:



Status > Connections > Gateway Network
Configuration

Connections

« ‹ 1 of 1 › »
Filter type to filter View 20

Remote Gateway	Direction	Redundancy Role	Last Comm	Ping Time	Status	Fault Count	Actions
gateway+b	Outgoing	Independent	a few seconds ago	1 ms	Running	0	Details

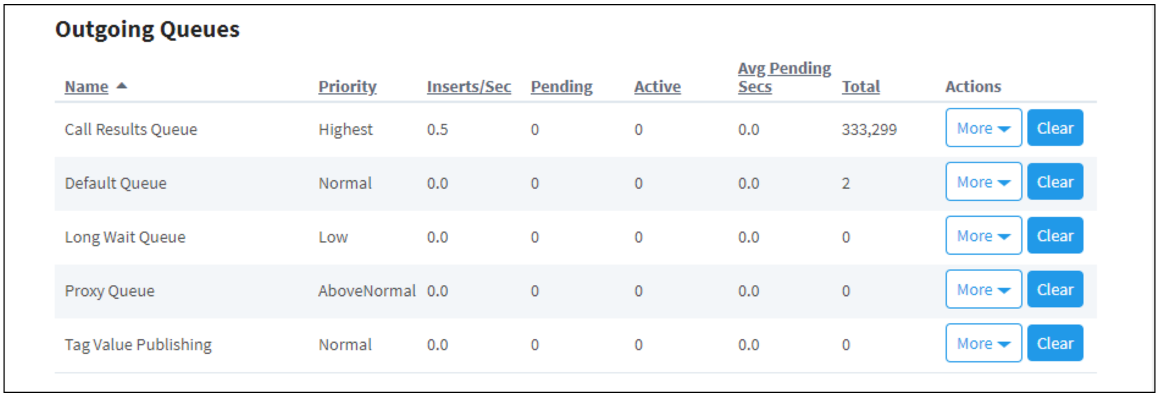
Remote Gateways

« ‹ 1 of 1 › »
Filter type to filter View 20

Gateway Name	Outgoing Msg/Sec	Incoming Msg/Sec	Pending	Active	Connected Through	Status	Actions
Gateway B	0.5	1.2	0	0	gateway+b	Connected	Details

Outgoing Queues

The Gateway Network Statistics page has three sections associated with it. First, the Outgoing Queues section. The Outgoing Queues section shows tasks that are both in a pending and active state. A pending task is a task that has not yet been dispatched to the gateway network thread pool. An active task is a task that is being processed by the gateway network thread pool and should show up under the Gateway Network Connection Status page as either an outgoing or incoming task.



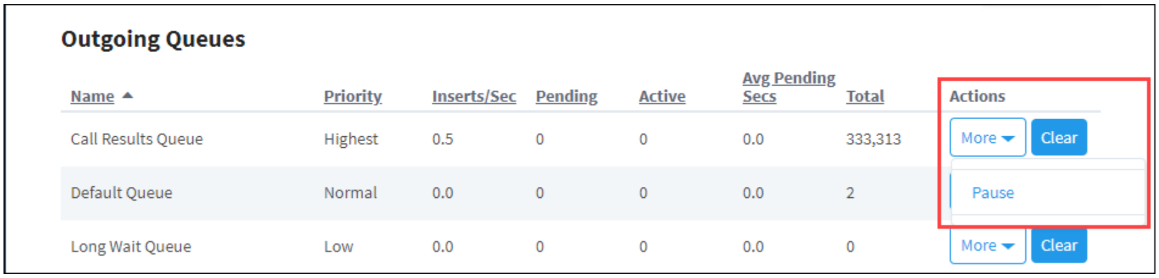
Outgoing Queues

Name ▲	Priority	Inserts/Sec	Pending	Active	Avg Pending Secs	Total	Actions
Call Results Queue	Highest	0.5	0	0	0.0	333,299	More ▾ Clear
Default Queue	Normal	0.0	0	0	0.0	2	More ▾ Clear
Long Wait Queue	Low	0.0	0	0	0.0	0	More ▾ Clear
Proxy Queue	AboveNormal	0.0	0	0	0.0	0	More ▾ Clear
Tag Value Publishing	Normal	0.0	0	0	0.0	0	More ▾ Clear

Attribute	Description
Name	Name of the queue.
Priority	Level of priority for a queue.
Inserts/Sec	Rate of task inserts per second for a queue
Pending	Number of pending tasks in a queue that have not yet been dispatched to a Send/Receive thread.
Active	Number of messages being actively processed by the gateway network connection.
Avg Pending Secs	Average number of seconds that a task has been pending in a queue.
Total	Total number of tasks executed from this queue.
Actions	Set of actions associated with a queue. Users can both Pause and Clear a queue.

Temporary Queue Actions

The Gateway Network Statistics page has controls for pausing and clearing a queue. To **pause** a queue means no new tasks will be allowed to be inserted into the paused queue. To **clear** a queue means that all pending tasks will be purged. These actions are designed to help the user deal with a possibly overloaded gateway network connection due to a specific sub system flooding the queue with more tasks than the connection can handle. Note that neither of these actions will have any effect on active tasks, as they have already been dispatched to the gateway network connection and cannot be cancelled.

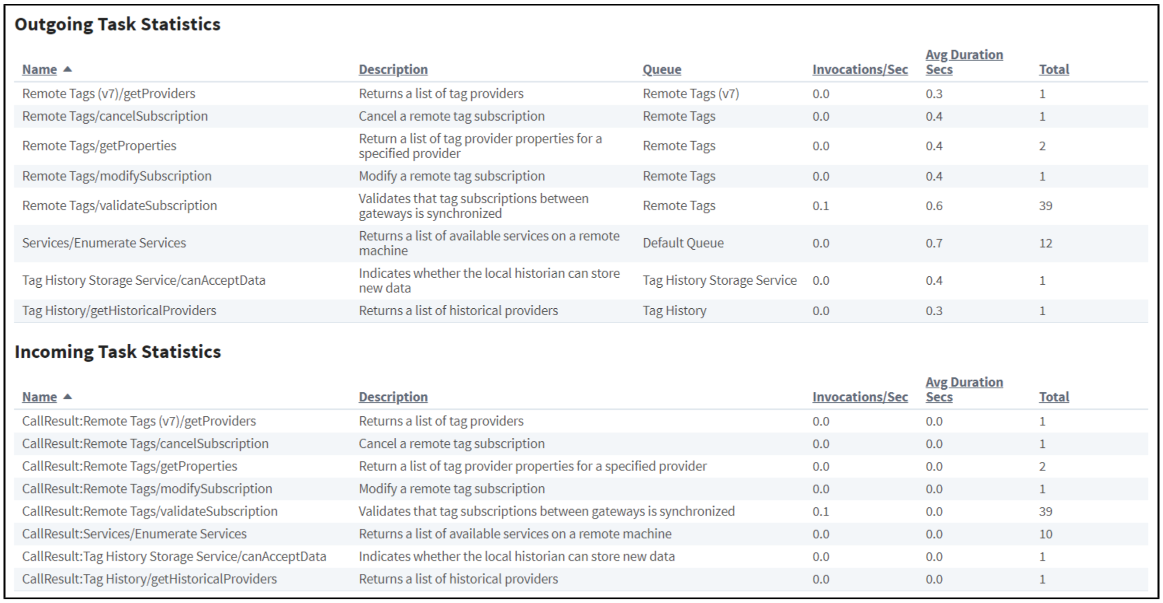


Outgoing Queues

Name ▲	Priority	Inserts/Sec	Pending	Active	Avg Pending Secs	Total	Actions
Call Results Queue	Highest	0.5	0	0	0.0	333,313	More ▾ Clear
Default Queue	Normal	0.0	0	0	0.0	2	Pause
Long Wait Queue	Low	0.0	0	0	0.0	0	More ▾ Clear

Outgoing and Incoming Tasks Statistics

The Outgoing and Incoming Task Statistics section shows individual tasks that have been processed by the Gateway Network Connection.



Outgoing Task Statistics

Name ▲	Description	Queue	Invocations/Sec	Avg Duration Secs	Total
Remote Tags (v7)/getProviders	Returns a list of tag providers	Remote Tags (v7)	0.0	0.3	1
Remote Tags/cancelSubscription	Cancel a remote tag subscription	Remote Tags	0.0	0.4	1
Remote Tags/getProperties	Return a list of tag provider properties for a specified provider	Remote Tags	0.0	0.4	2
Remote Tags/modifySubscription	Modify a remote tag subscription	Remote Tags	0.0	0.4	1
Remote Tags/validateSubscription	Validates that tag subscriptions between gateways is synchronized	Remote Tags	0.1	0.6	39
Services/Enumerate Services	Returns a list of available services on a remote machine	Default Queue	0.0	0.7	12
Tag History Storage Service/canAcceptData	Indicates whether the local historian can store new data	Tag History Storage Service	0.0	0.4	1
Tag History/getHistoricalProviders	Returns a list of historical providers	Tag History	0.0	0.3	1

Incoming Task Statistics

Name ▲	Description	Invocations/Sec	Avg Duration Secs	Total
CallResult:Remote Tags (v7)/getProviders	Returns a list of tag providers	0.0	0.0	1
CallResult:Remote Tags/cancelSubscription	Cancel a remote tag subscription	0.0	0.0	1
CallResult:Remote Tags/getProperties	Return a list of tag provider properties for a specified provider	0.0	0.0	2
CallResult:Remote Tags/modifySubscription	Modify a remote tag subscription	0.0	0.0	1
CallResult:Remote Tags/validateSubscription	Validates that tag subscriptions between gateways is synchronized	0.1	0.0	39
CallResult:Services/Enumerate Services	Returns a list of available services on a remote machine	0.0	0.0	10
CallResult:Tag History Storage Service/canAcceptData	Indicates whether the local historian can store new data	0.0	0.0	1
CallResult:Tag History/getHistoricalProviders	Returns a list of historical providers	0.0	0.0	1

Outgoing and Incoming Tasks Attributes	Description
Name	Name of a task.
Description	Description for a task.
Queue	Queue invoking a task (For Outgoing Tasks only)
Invocation/Sec	Rate at which a task is invoked per second.
Avg Duration Secs	Average duration in seconds of the time it takes for this task to execute.
Total	Number of times the task has been executed in total.

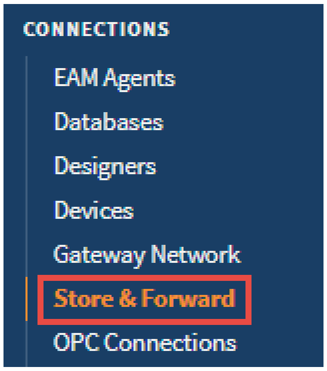
Related Topics ...

- Gateway Status
- Diagnostics - Logs

Connections - Store & Forward

Store & Forward

The Store and Forward page displays a list of the Store and Forward engines, including their status, as well as the number of records currently in each Store and Forward system. If the database connection becomes faulted, the database records wait in the Store and Forward system until the database connection is restored.



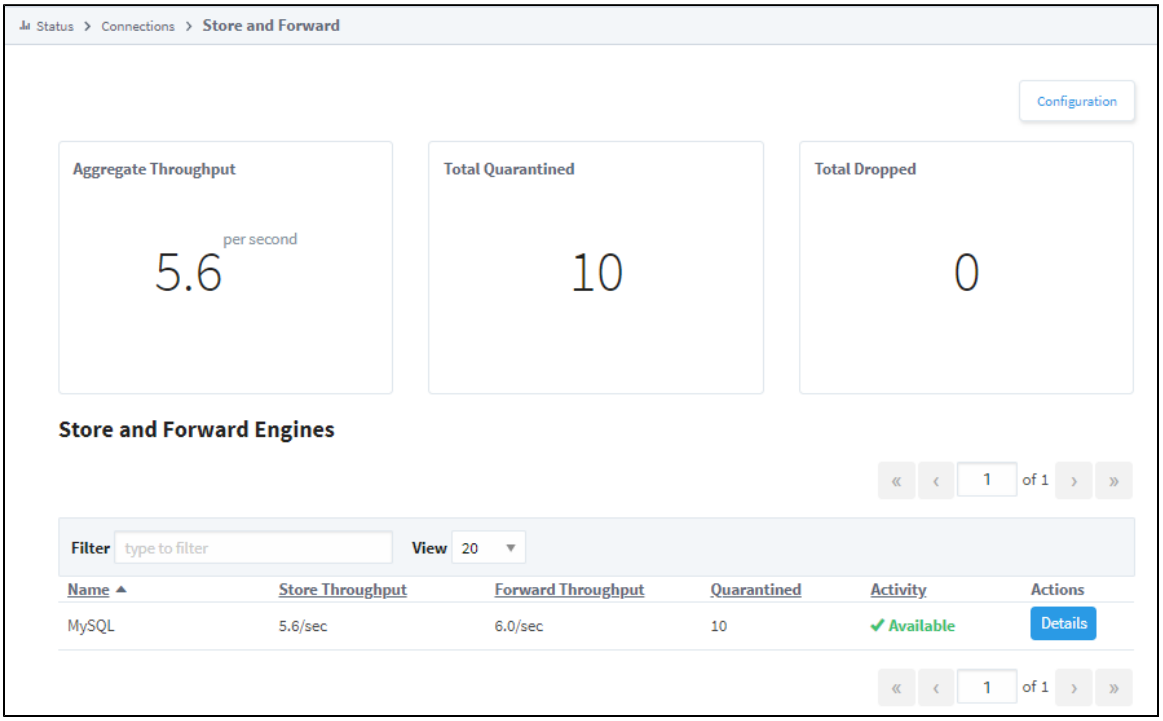
On this page

...

Store & Forward Page

The main Store and Forward page lists out all store and forward engines. Typically, each database connection gets its own store and forward engine, so there should be one engine for each database connection. In addition to displaying some basic stats for each engine, you can also find some totals for all store and forward systems, to get an idea of how much data is being pushed through the system to databases and if any records are being dropped.

Attributes	Description
Store and Forward Connections	
Aggregate Throughput	Aggregate number of records inserted into a database from any Store and Forward engine per second.
Total Quarantine	Number of quarantined items for all Store and Forward engines.
Total Dropped	Number of records dropped from the Store and Forward engines. A record is considered dropped if it can not be added to one of the buffers, (i.e., when a buffer is full and the Store and Forward engine can no longer accept new records).
Store and Forward Engines	
Name	Name of the Store and Forward engine.
Store Throughput	Number of records that go through the Store and Forward Engine per second.
Forward Throughput	Number of records to be forwarded on to the database per second.
Quarantined	Is data that has erred-out multiple times during attempts to forward it, or data that could not be stored because of some configuration issues.
Activity	Current state of the Store and Forward engine.
Actions	By clicking Details, shows additional information about Store and Forward engines.



Aggregate Throughput	Total Quarantined	Total Dropped
5.6 per second	10	0

Store and Forward Engines

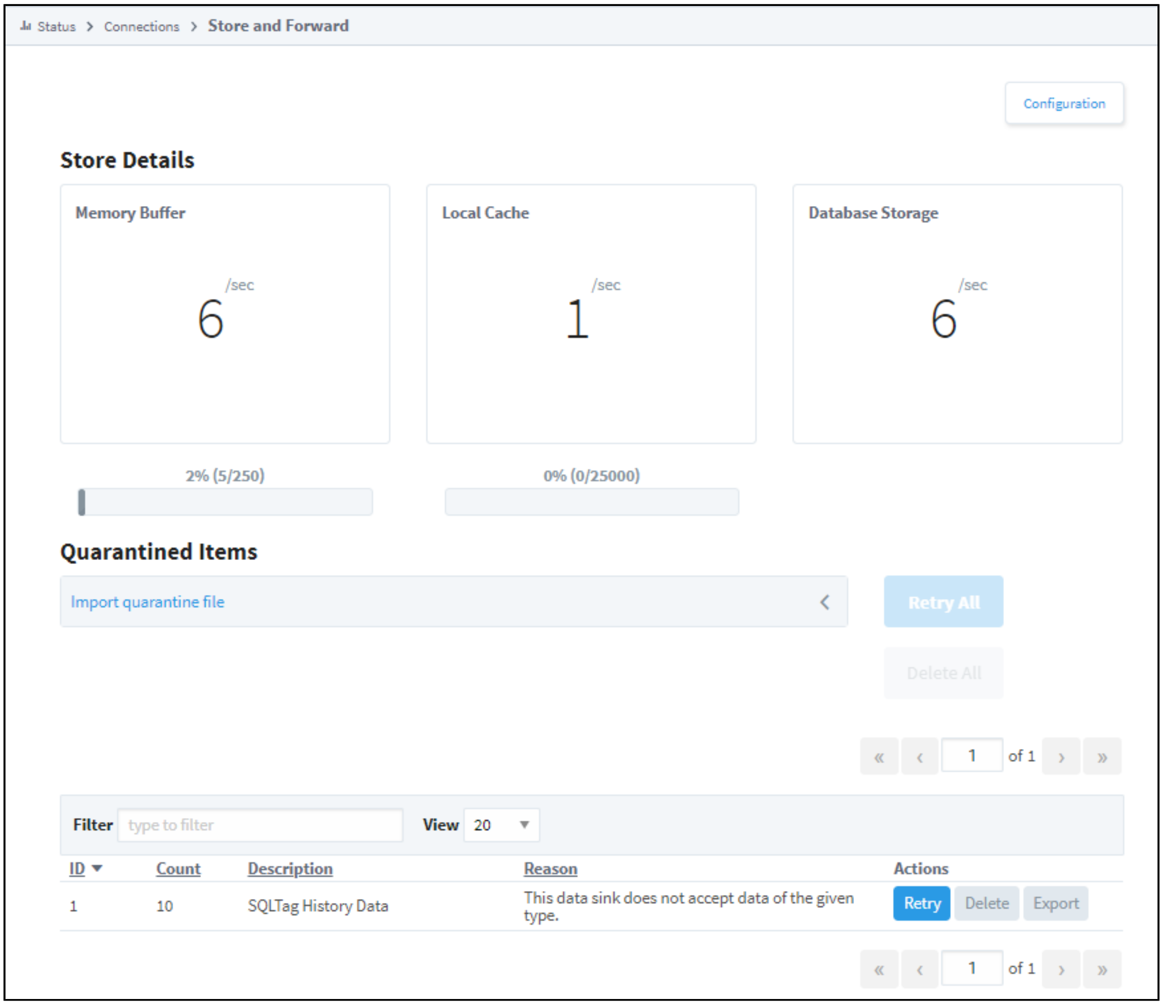
Filter type to filter View 20

Name ▲	Store Throughput	Forward Throughput	Quarantined	Activity	Actions
MySQL	5.6/sec	6.0/sec	10	✓ Available	Details

Store & Forward Details

Clicking the **Details** button brings up a new window that will show even more details about the records in the selected Store and Forward Engine. Here, we can see a count of the number of records in the memory buffer and local cache, as well as the number of quarantined records. The quarantined items at the bottom of the Details page will have some buttons that allow you to control the data that is in the quarantine. The quarantined item can be retried, where it will be thrown back through the Store and Forward system to see if it will go through properly, assuming the original reason why it was quarantined has been fixed. It can also be deleted so that it is no longer taking up space in the Store and Forward system, or exported to your local machine where you can save it to try again later. You can then import the file back from the same page when you resolved the issue that caused the data to be quarantined in the first place.

Attribute	Description
Memory Buffer	Number of records entering the Memory Buffer per second. The progress bar shows the percent of the buffer being utilized, along with the current and max number of records.
Local Cache	Number of records entering the Local Cache per second. An "Idle" state means the engine is able to successfully store all records into the database before the **Write Size** or **Write Time** values have been reached. The progress bar shows the percent of the buffer being utilized, along with the current and max number of records.
Database Storage	Displays the number of records pushed from either buffer to the database per second.
Quarantined Items	Shows a list of quarantine items and allows you to choose the quarantined file and import it.
ID	Identification number of the quarantine item.
Count	Number of occurrences for the quarantined item.
Description	Description of where the quarantine item origninated from.
Reason	Explanation why the record was placed into quarantine.
Actions	Provides and opportunity to retry, delete, or export the items from quarantine.



Store Details

Memory Buffer	Local Cache	Database Storage
6 /sec	1 /sec	6 /sec
2% (5/250)	0% (0/25000)	

Quarantined Items

Filter: type to filter View: 20

ID ▼	Count	Description	Reason	Actions
1	10	SQLTag History Data	This data sink does not accept data of the given type.	Retry Delete Export

Related Topics ...

- Using Store and Forward
- Controlling Quarantine Data

Connections - OPC Connections

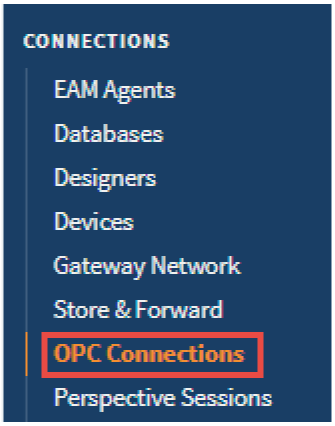
OPC Connections

On this page

...

The OPC Connections page displays all currently configured OPC (both UA and DA) connections.

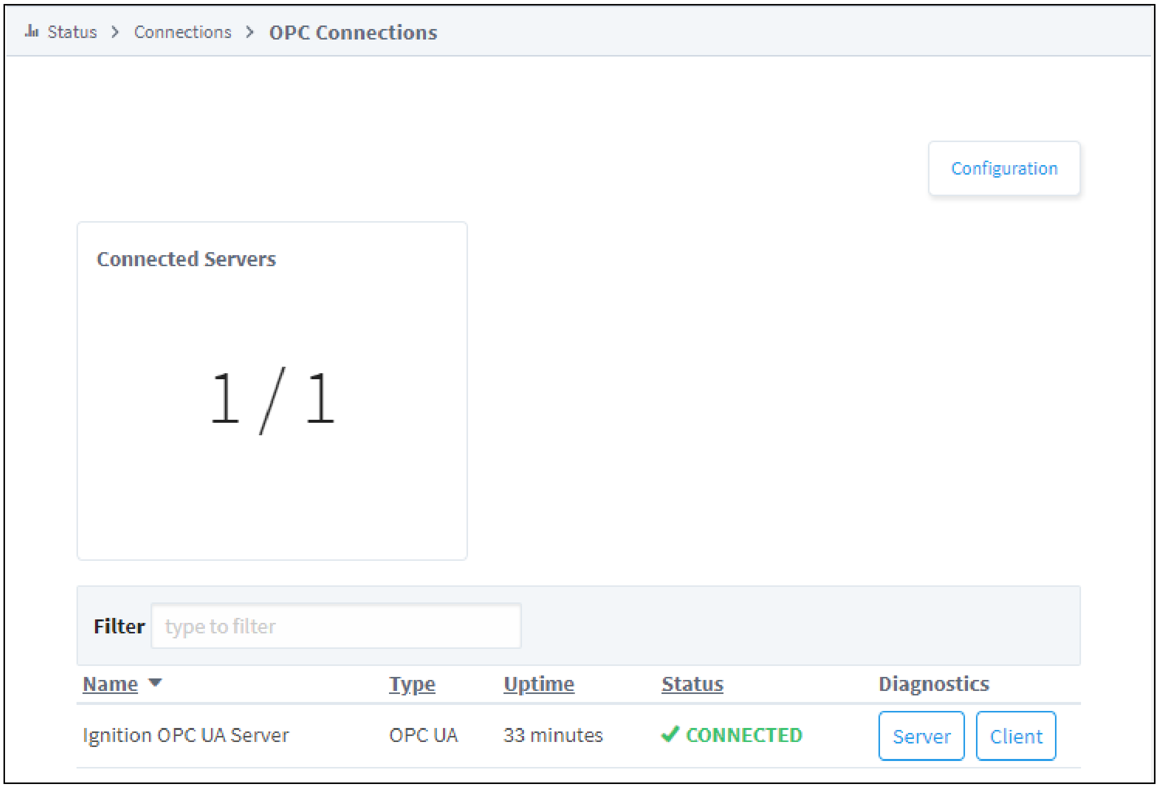
> ⚠ If you are using an OPC UA connection, you should be using the OPC UA Module. If you are using the DA connection, you should be using the OPC COM Module.



OPC Connections Page

Here on the main OPC Connections page, we can see a list of all current OPC connections, as well as their status. If any are faulted, you can click the red faulted status to get an error message popup with a full description of the error.

Attribute	Description
Connected Servers	Displays the list of OPC servers out of configured servers, and their status.
Name	OPC server name.
Filter	Search criteria to filter for specific server names.
Type	OPC server type - UA or DA
Uptime	Total time OPC server is connected.
Status	Current status of OPC server.
Diagnostics	The following feature is new in Ignition version **8.0.15** Click here to check out the other new features Displays diagnostic information for any connected OPC UA server. • Server - Shows server diagnostics. • Client - Shows client connection subscription diagnostics.



Connected Servers

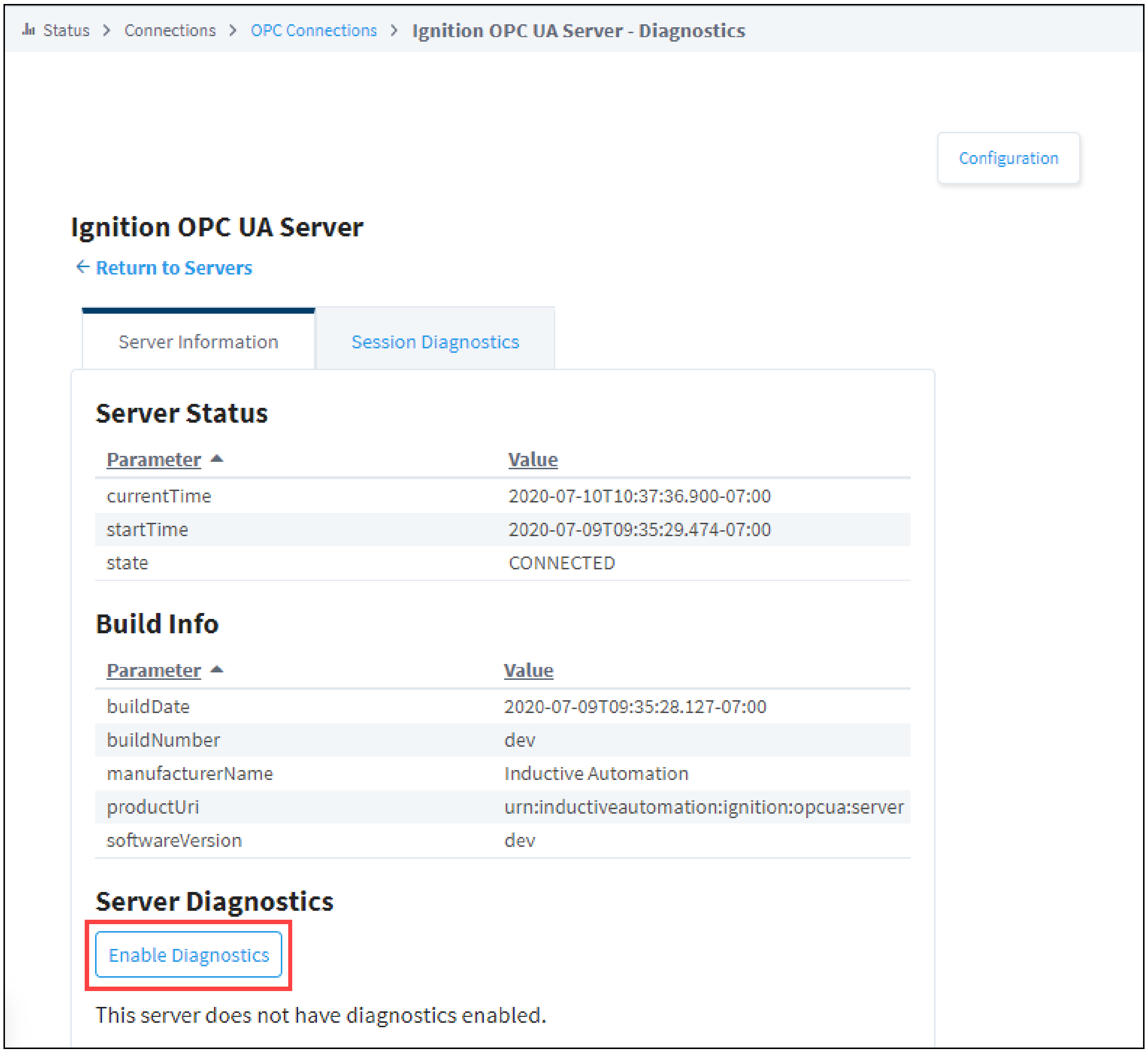
1 / 1

Filter: type to filter

Name	Type	Uptime	Status	Diagnostics
Ignition OPC UA Server	OPC UA	33 minutes	✓ CONNECTED	Server Client

Server

Click the **Server** button to get information on the server. By default, diagnostics are set to off since they can generate a considerable amount of network overhead and impact performance. To turn on diagnostics, click **Enable Diagnostics**. Not all OPC UA servers support diagnostics.



Ignition OPC UA Server

Server Status

Parameter	Value
currentTime	2020-07-10T10:37:36.900-07:00
startTime	2020-07-09T09:35:29.474-07:00
state	CONNECTED

Build Info

Parameter	Value
buildDate	2020-07-09T09:35:28.127-07:00
buildNumber	dev
manufacturerName	Inductive Automation
productUri	urn:inductiveautomation:ignition:opcua:server
softwareVersion	dev

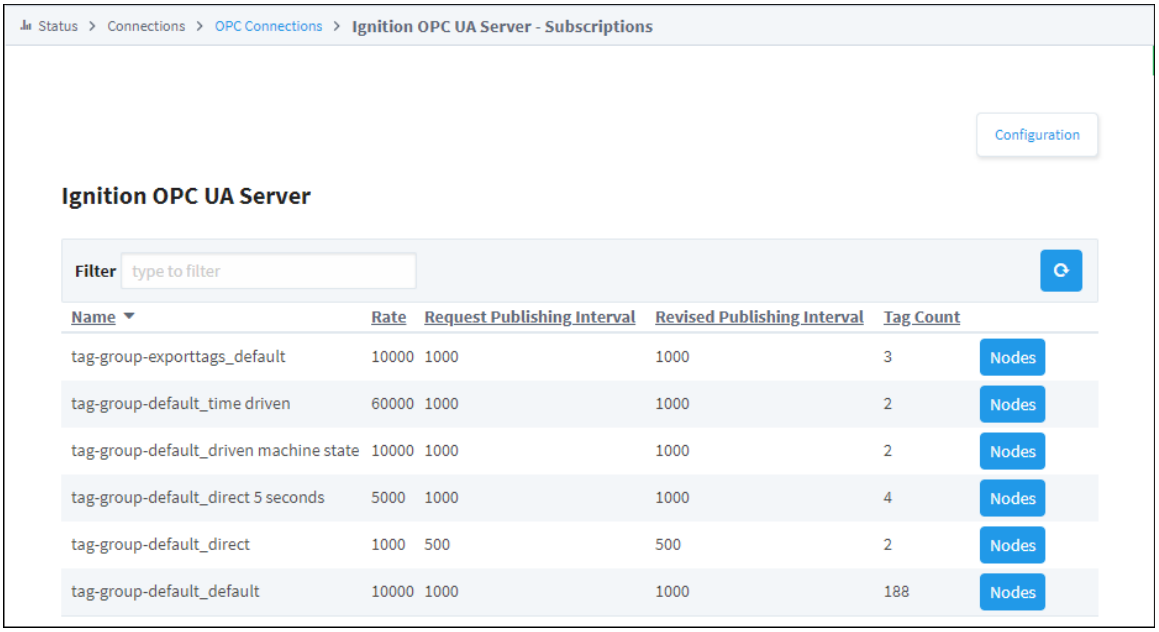
Server Diagnostics

This server does not have diagnostics enabled.

Clients

By clicking on the **Client** button, it brings up the subscription information for that particular server. It will list out all of the subscriptions to that server with the publishing rates, as well as the number of items within that subscription.

Attribute	Description
Filter	Search criteria to filter for specific subscriptions.
Refresh	Refreshes subscription data.
Name	Displays the subscription name.
Rate	The rate defined in the Tag Group, used as the requested Sampling Interval for Monitored Items belonging to the corresponding Subscription.
Request Publishing Interval	The rate a subscription will report accumulated change notifications at. The interval is derived from Rate, unless explicitly defined.
Revised Publishing Interval	The server's revised rate for accumulated change notifications.
Tag Count	The number of Tags currently subscribed to that Tag Group.


Status > Connections > OPC Connections > Ignition OPC UA Server - Subscriptions
Configuration
Filter
type to filter

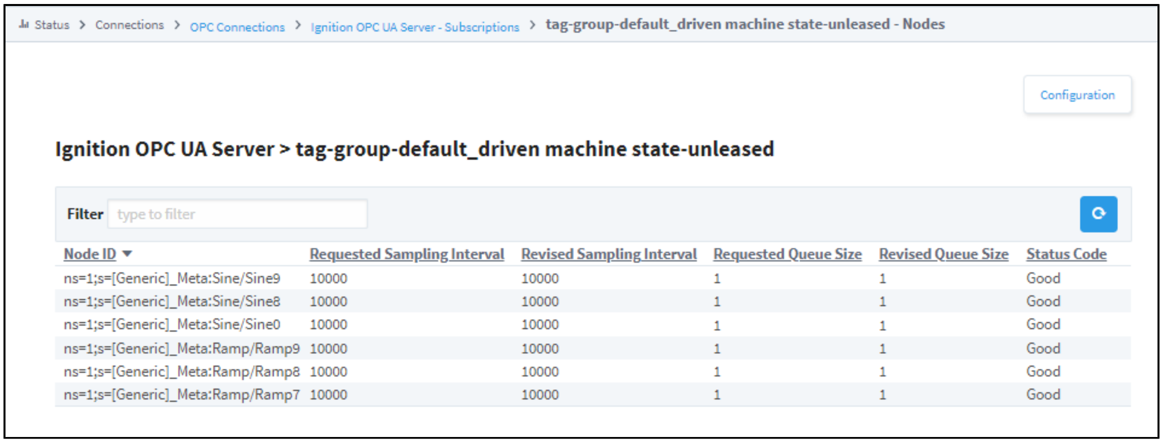
Ignition OPC UA Server

Name	Rate	Request Publishing Interval	Revised Publishing Interval	Tag Count	
tag-group-exporttags_default	10000	1000	1000	3	Nodes
tag-group-default_time driven	60000	1000	1000	2	Nodes
tag-group-default_driven machine state	10000	1000	1000	2	Nodes
tag-group-default_direct 5 seconds	5000	1000	1000	4	Nodes
tag-group-default_direct	1000	500	500	2	Nodes
tag-group-default_default	10000	1000	1000	188	Nodes

Nodes

Clicking the **Nodes** button for one of the subscriptions will bring up the list of subscribed OPC items.

Attribute	Description
Filter	Search criteria to filter for a specific node.
Refresh	Refreshes node data.
Node ID	The OPC item path.
Requested Sampling Interval	The rate the underlying tag/node will be polled at.
Revised Sampling Interval	The revised rate at which the underlying tag/node will be polled at.
Requested Queue Size	Determines how many data points can be stored and transferred to the client once the sampling rate elapses when the sampling rate is slower that the publishing rate.
Status Code	Status of the node.


Status > Connections > OPC Connections > Ignition OPC UA Server - Subscriptions > tag-group-default_driven machine state-unleased - Nodes
Configuration
Filter
type to filter

Ignition OPC UA Server > tag-group-default_driven machine state-unleased

Node ID	Requested Sampling Interval	Revised Sampling Interval	Requested Queue Size	Revised Queue Size	Status Code
ns=1;s=[Generic]_Meta:Sine/Sine9	10000	10000	1	1	Good
ns=1;s=[Generic]_Meta:Sine/Sine8	10000	10000	1	1	Good
ns=1;s=[Generic]_Meta:Sine/Sine0	10000	10000	1	1	Good
ns=1;s=[Generic]_Meta:Ramp/Ramp9	10000	10000	1	1	Good
ns=1;s=[Generic]_Meta:Ramp/Ramp8	10000	10000	1	1	Good
ns=1;s=[Generic]_Meta:Ramp/Ramp7	10000	10000	1	1	Good

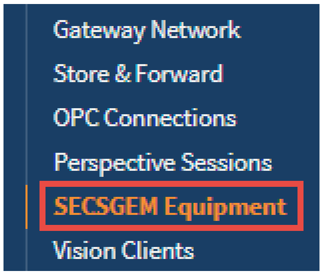
Related Topics ...

- OPC UA
- OPC UA Client Connection Settings

Connections - SECS/GEM Equipment

SECS/GEM Equipment

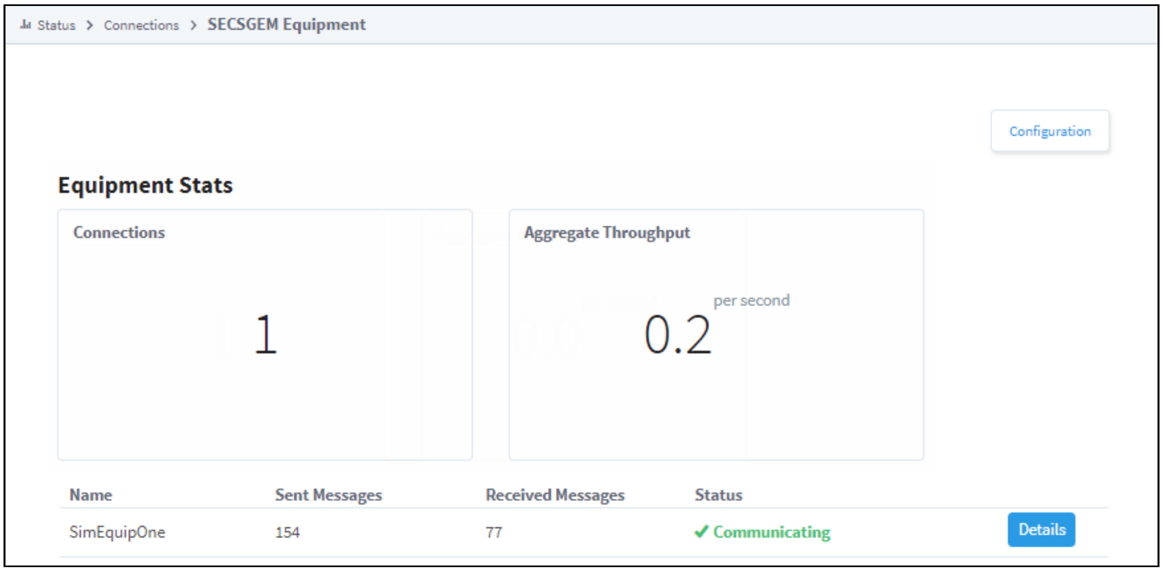
The SECSGEM Equipment page displays a list of all equipment connections, their status, as well as the number of sent requests and received messages. If any piece of equipment becomes faulted, it will show a status of "Not Connected." If you click the **Details** button, it opens a new page with some Connection Stats. Note: the SECSGEM Module is not standard and will be missing for most installs of Ignition. The SECSGEM Equipment Connection will only be displayed in the Status section of the Gateway webpage under Connections when the module is installed.



SECS/GEM Equipment Page

The main SECS/GEM Equipment page lists out all equipment connections and displays the number of messages sent between them and Ignition.

Attribute	Description
Connections	The total number of active SECS/GEM Equipment connections.
Aggregate Throughput	The messages per second the system is sending and receiving from all devices at that time.
Name	The name of the equipment connection in the list.
Sent Messages	The number of messages sent to the equipment.
Received Messages	The number of messages received from the equipment.
Status	The status of the equipment.



Equipment Stats

Connections: 1

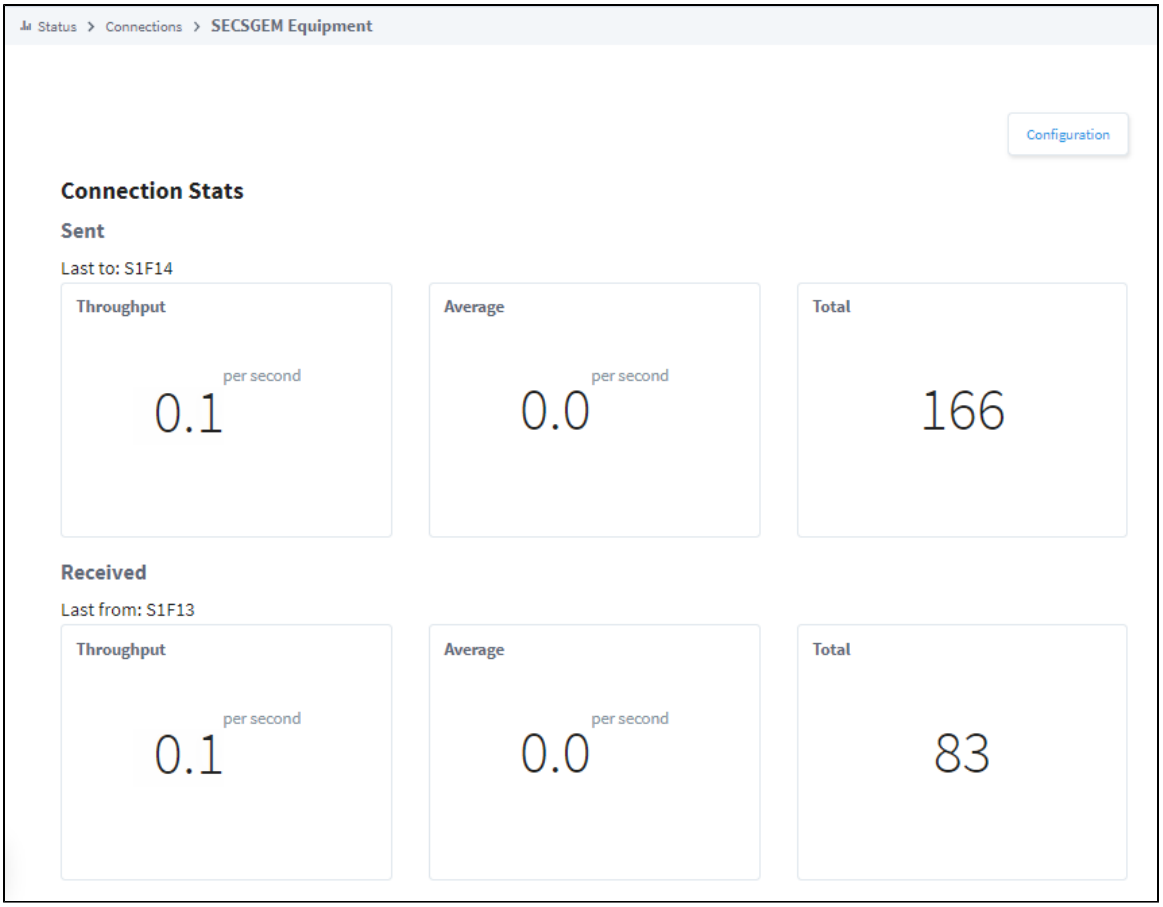
Aggregate Throughput: 0.2 per second

Name	Sent Messages	Received Messages	Status	
SimEquipOne	154	77	✓ Communicating	Details

SECS/GEM Equipment Details

Clicking the **Details** button to the right of a piece of equipment will take you to a page that shows more detailed information for that particular piece of equipment.

Attribute	Description
Sent Messages	
Throughput	The messages per second the system is currently sending.
Average	The average messages sent per second from when the equipment was first enabled.
Total	The total messages sent.
Received Messages	
Throughput	The messages per second the system is currently receiving.
Average	The average messages received per second from when the equipment was first enabled.
Total	The total messages received.



Connection Stats

Sent

Last to: S1F14

Throughput	Average	Total
0.1 per second	0.0 per second	166

Received

Last from: S1F13

Throughput	Average	Total
0.1 per second	0.0 per second	83

Related Topics ...

- SECS/GEM

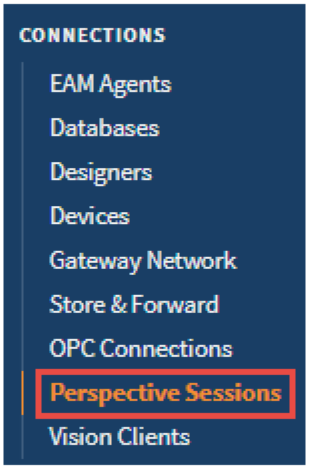
Connections - Perspective Sessions

On this page

...

Perspective Sessions

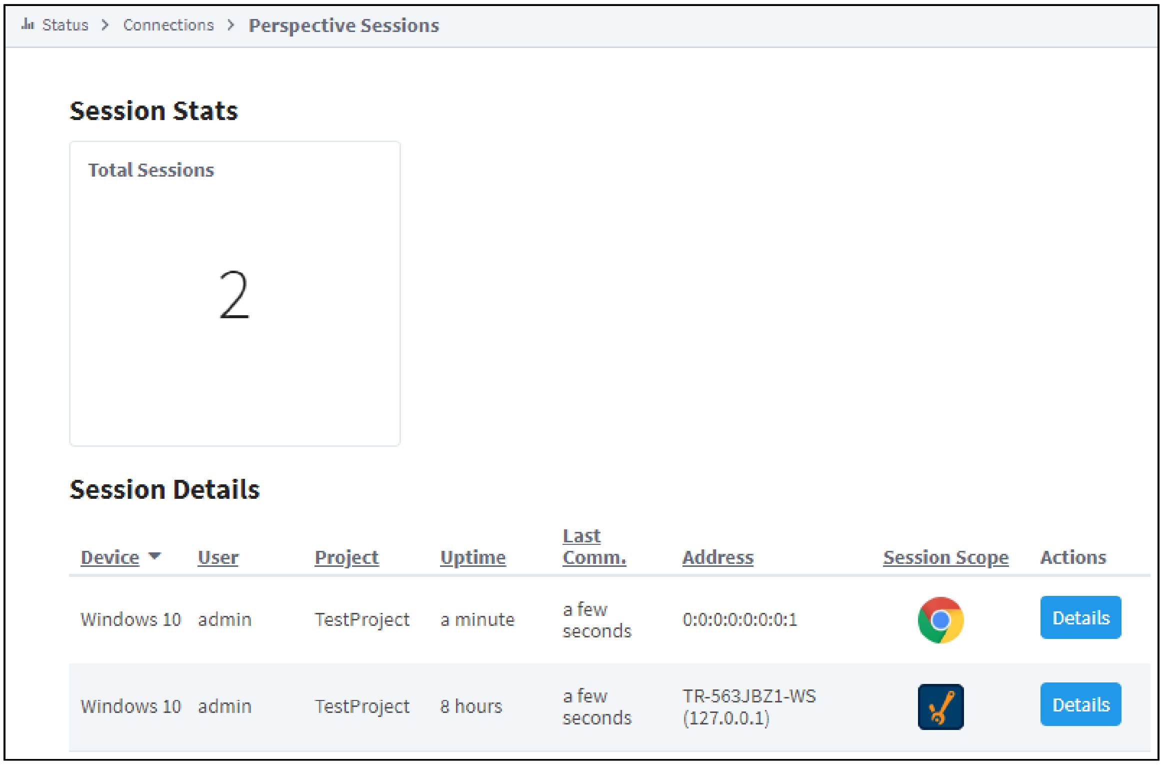
Much like the Designers page, the Perspective Sessions page shows a ton of information regarding currently open Sessions. All the open Perspective Sessions are displayed on the page along with some basic information about each session, such as what user is logged into each session and the project they are currently working on along with their IP address. Each session has a Details button that allows you navigate all the elements of a session: pages, view instances, and components.



Perspective Sessions Page

The main Perspective Sessions page displays a list of all currently running Designer and Client Sessions to show how many are open, and the name of the users that are currently using them. By clicking on the **Details** button to the right of a session will display even more information about that particular session, including any Log Activity pertaining to errors recorded while the session was active.

Attributes	Description
Sessions Details	
Total Sessions	The number of currently active Designer and Perspective Sessions.
Device	Type of device running the Session.
User	The user logged into the Session.
Project	The name of the project open in the Session.
Uptime	The total uptime of the Session.
Last Comm.	Date and time recorded for the last communication with the Session.
Address	The IP Address and computer name where the Session is launched.
Session Scope	Indicates what browser opened the Session, and the Perspective icon indicates the Designer is open.
Actions	Displays more details about the session.



Status > Connections > Perspective Sessions

Session Stats

Total Sessions

2

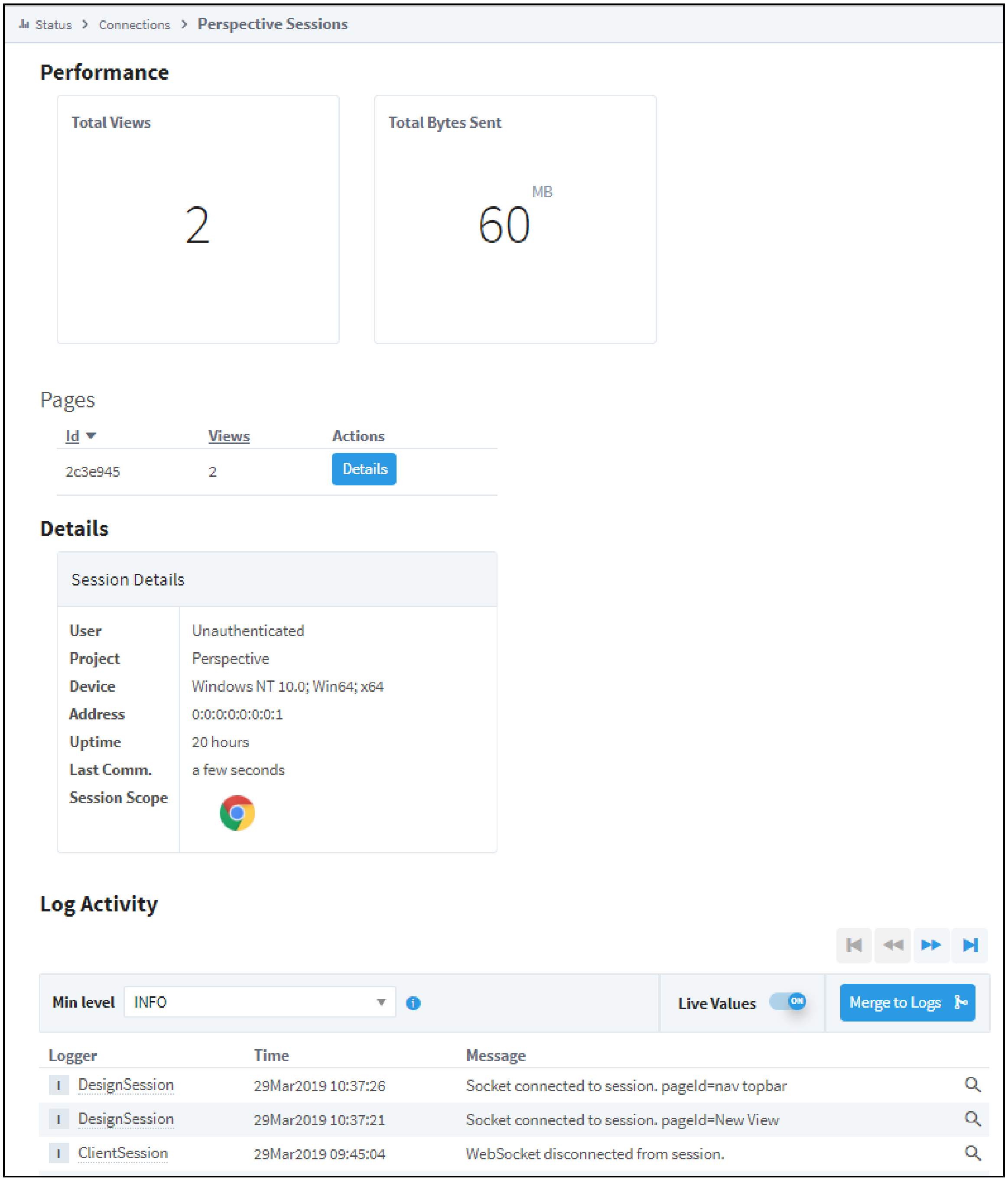
Session Details

Device	User	Project	Uptime	Last Comm.	Address	Session Scope	Actions
Windows 10	admin	TestProject	a minute	a few seconds	0:0:0:0:0:0:0:1		Details
Windows 10	admin	TestProject	8 hours	a few seconds	TR-563JBZ1-WS (127.0.0.1)		Details

Perspective Session Details

The Performance Page within the session displays more in depth information such as the Page Id and how many views are on each Page. Hit the **Details** button to get more detailed information about each page in the sesson.

Attributes	Description
Performance	
Total Views	The number of currently active pages.
Total Bytes Sent	Displays total number of bytes sent out.
Pages	
Id	Page Id
Views	Displays total number of views on a page.
Actions	Displays the details of the view instances in the session.
Log Activity	
Min level	Dropdown menu with options Info, Debug, and Trace.
Live Values	Toggle switch to turn live values on or off.
Merge to Logs	Merge the settings of the current view with the main Diagnostics Log Viewer.
Logger	Name of the logger that describes the context of the message.
Time	Time of log.
Message	Message for the log.
🔍 (Log Properties)	Displys log properties of the logged event.



Performance

Total Views	Total Bytes Sent
2	60 MB

Pages

Id	Views	Actions
2c3e945	2	Details

Details

Session Details

User	Unauthenticated
Project	Perspective
Device	Windows NT 10.0; Win64; x64
Address	0:0:0:0:0:0:0:1
Uptime	20 hours
Last Comm.	a few seconds
Session Scope	

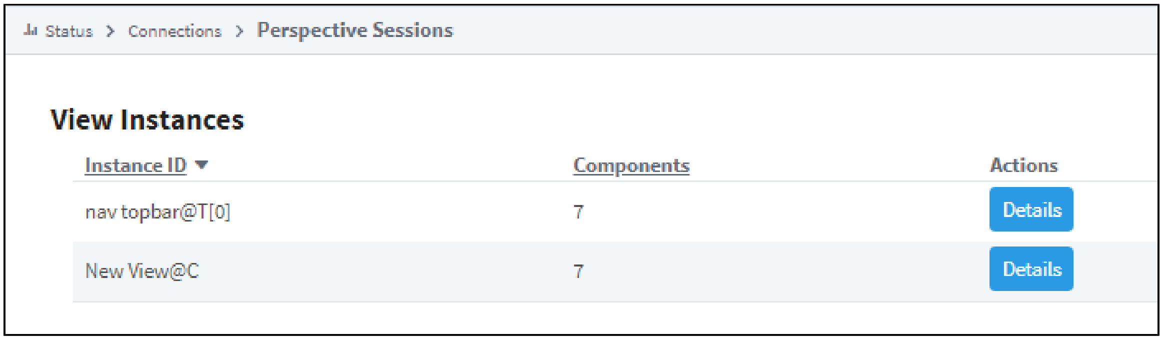
Log Activity

Logger	Time	Message
I DesignSession	29Mar2019 10:37:26	Socket connected to session. pageId=nav topbar
I DesignSession	29Mar2019 10:37:21	Socket connected to session. pageId=New View
I ClientSession	29Mar2019 09:45:04	WebSocket disconnected from session.

View Instances

The View Instances page shows the number of View Instances on a Page and the number of Components in each view instance. Press the **D etails** button next to each View Instance to get even more information about the type of components used in each view.

Attributes	Description
Sessions Details	
InstanceID	Displays View Instances for eac view.
Components	Total number of components used in the specified view.
Actions	By clicking on the Details button shows mor information for each view instance.



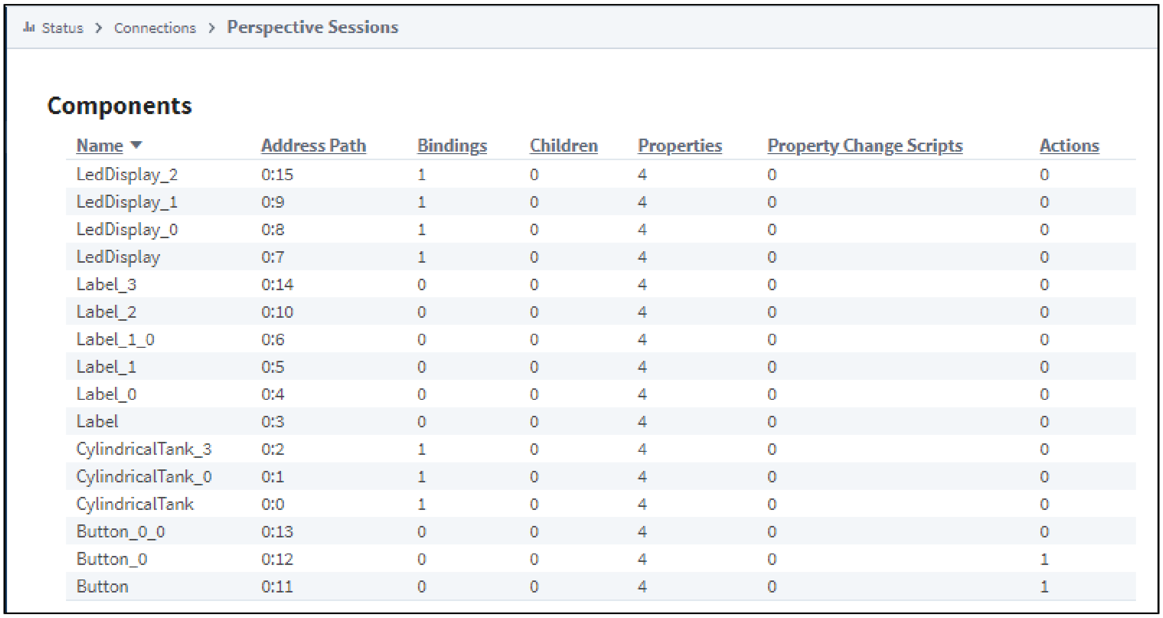
View Instances

Instance ID ▼	Components	Actions
nav topbar@T[0]	7	Details
New View@C	7	Details

Components

The Components page lists all the components used in a View Instance of Page along with the following information about each component.

Attributes	Description
Name	Name of the component.
AddressPath	Address of the component.
Bindings	Displays the number of bindings for each component.
Children	Displays the number of children for each component.
Properties	Displays the number of properties for each component.
PropertyChangeScripts	Displays the number of property change scripts for each component
Actions	Displays the number of actions for each component.



Components

Name ▼	Address Path	Bindings	Children	Properties	Property Change Scripts	Actions
LedDisplay_2	0:15	1	0	4	0	0
LedDisplay_1	0:9	1	0	4	0	0
LedDisplay_0	0:8	1	0	4	0	0
LedDisplay	0:7	1	0	4	0	0
Label_3	0:14	0	0	4	0	0
Label_2	0:10	0	0	4	0	0
Label_1_0	0:6	0	0	4	0	0
Label_1	0:5	0	0	4	0	0
Label_0	0:4	0	0	4	0	0
Label	0:3	0	0	4	0	0
CylindricalTank_3	0:2	1	0	4	0	0
CylindricalTank_0	0:1	1	0	4	0	0
CylindricalTank	0:0	1	0	4	0	0
Button_0_0	0:13	0	0	4	0	0
Button_0	0:12	0	0	4	0	1
Button	0:11	0	0	4	0	1

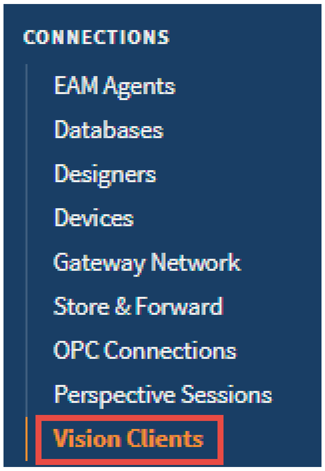
Related Topics ...

- Launching a Perspective Session

Connections - Vision Clients

Vision Clients

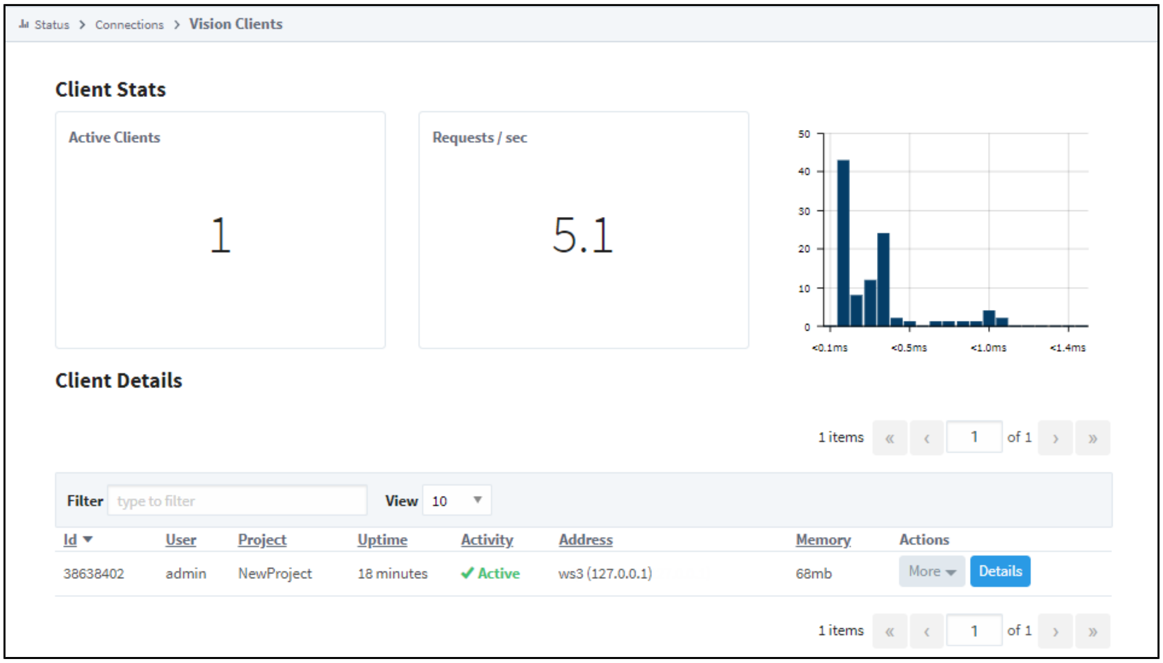
Much like the Designers page, the Vision Clients page shows information regarding currently opened Clients. The Clients are listed and show some basic information such as the address of the Client. From here, the Client session can be terminated by selecting the **More** button and hitting **Terminate**, or select the **Details** option to see more details about the Client session such as the number of Tags that the session is currently subscribed to, as well as a log of errors that may have happened with that Client.



Vision Clients Page

The main Vision Clients page displays a list of all currently running Clients which show how many clients are open and what users are currently using them.

Attributes	Description
Clients Stats and Details	
Active Clients	The number of currently active Clients.
Requests/sec	The number of requests per second coming from all Clients.
Id	The Client Id.
User	The user logged in on the Client session.
Project	The name of the project open in the Client session.
Uptime	The total uptime of the Client session.
Activity	Denotes whether the Client session is currently connected and active or not.
Address	The IP Address and computer name where the Client is launched.
Memory	The Client's current memory usage.
Actions	The option to terminate the Client session, or see more details about the session.



Status > Connections > Vision Clients

Client Stats

Active Clients

1

Requests / sec

5.1

Client Details

1 items « ‹ 1 of 1 › »

Filter type to filter View 10

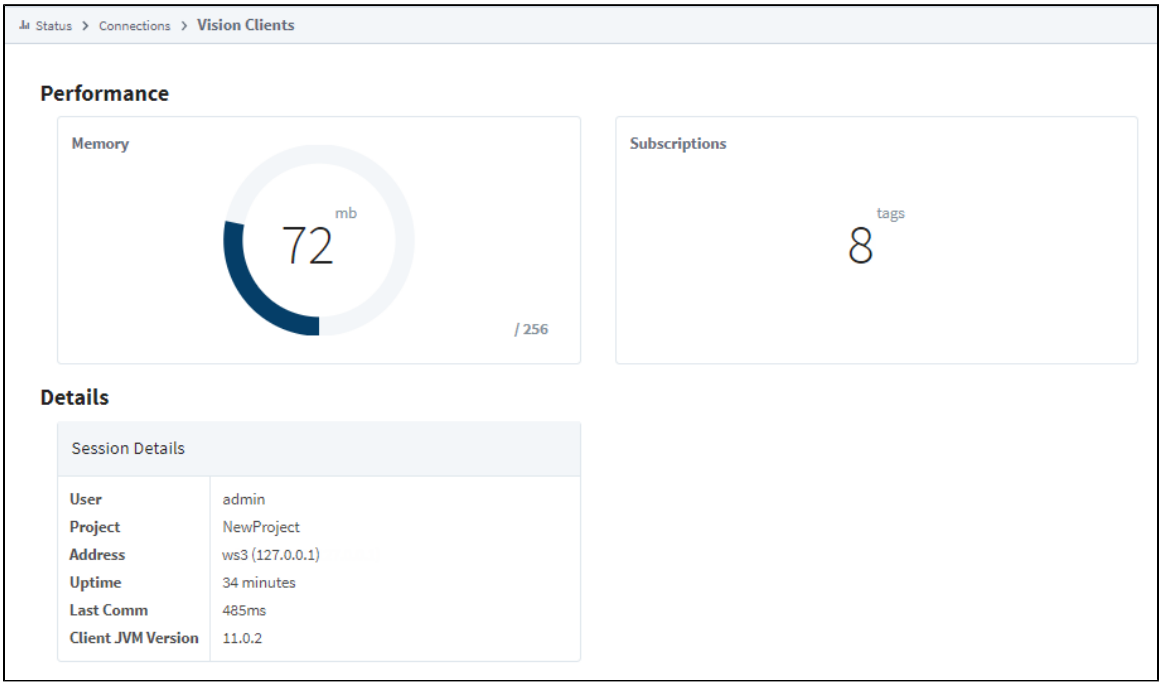
Id	User	Project	Uptime	Activity	Address	Memory	Actions
38638402	admin	NewProject	18 minutes	✓ Active	ws3 (127.0.0.1)	68mb	More Details

1 items « ‹ 1 of 1 › »

Client Session Details

Clicking on the **Details** button for a Client session will take you to a page that displays more in depth information for that particular Client session. It also gives a logger at the bottom where errors coming from that particular Client can be seen.

Attributes	Description
Client Performance and Details	
Memory	The Client's current memory use.
Subscriptions	The number of Tags the Client is currently subscribed to.
User	The user logged in on the Client session
Project	The name of the project open in the Client session.
Address	The IP Address and computer name where the Client is launched.
Uptime	The total uptime of the Client session.
Last Comm	The last time the Gateway communicated with the Client.
Client JVM Version	The Java version that the Client is currently running on.



Session Details

User	admin
Project	NewProject
Address	ws3 (127.0.0.1)
Uptime	34 minutes
Last Comm	485ms
Client JVM Version	11.0.2

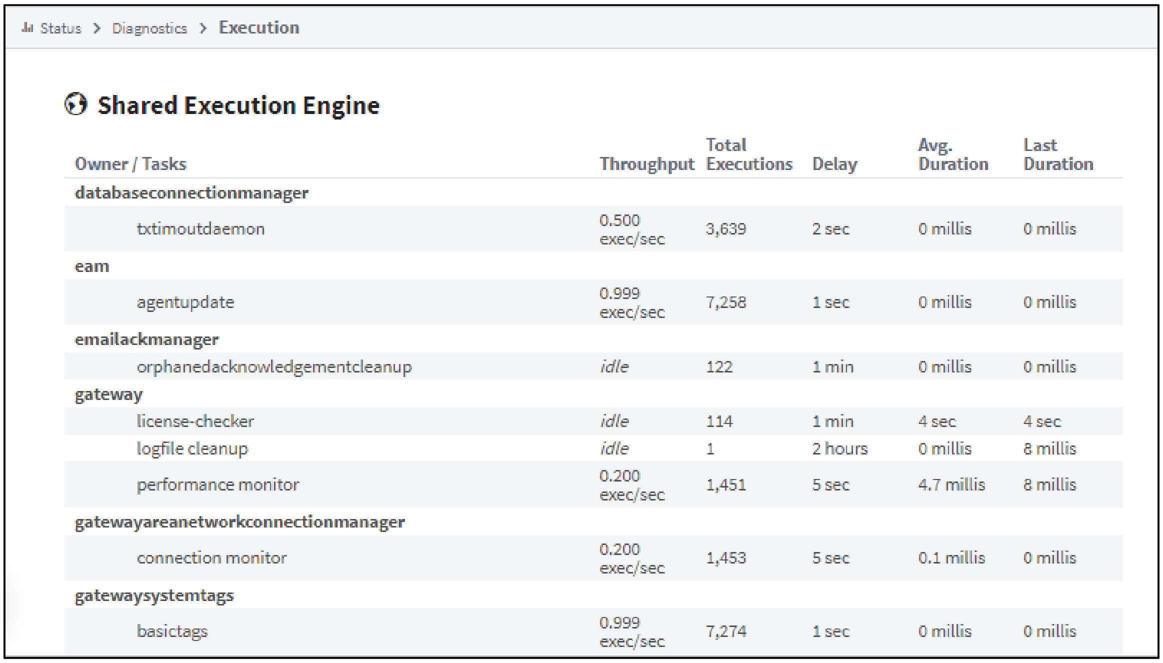
Related Topics ...

- Vision Client Launcher

Diagnostics - Execution

The Execution page reports the status of all tasks that your Gateway runs on a schedule. Here you can find helpful information such as the duration and execution time of an alarm journal update or the average time it takes your Gateway to execute a Tag Group.

Attributes	Description
Throughput	Number of executions that come through per second.
Total Executions	Total number of times the task executed.
Delay	Amount of time waiting to execute the task.
Avg. Duration	Average time to run the task.
Last Duration	Amount of time to run the last task.



Status > Diagnostics > Execution

Shared Execution Engine

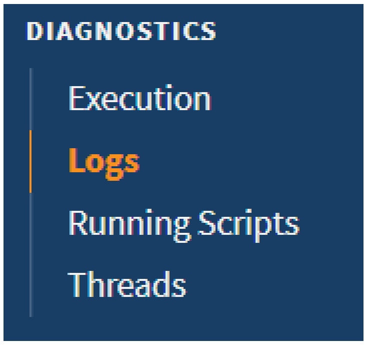
Owner / Tasks	Throughput	Total Executions	Delay	Avg. Duration	Last Duration
databaseconnectionmanager					
txtimoutdaemon	0.500 exec/sec	3,639	2 sec	0 millis	0 millis
eam					
agentupdate	0.999 exec/sec	7,258	1 sec	0 millis	0 millis
emailackmanager					
orphanedacknowledgementcleanup	*idle*	122	1 min	0 millis	0 millis
gateway					
license-checker	*idle*	114	1 min	4 sec	4 sec
logfile cleanup	*idle*	1	2 hours	0 millis	8 millis
performance monitor	0.200 exec/sec	1,451	5 sec	4.7 millis	8 millis
gatewayareanetworkconnectionmanager					
connection monitor	0.200 exec/sec	1,453	5 sec	0.1 millis	0 millis
gatewaysystemtags					
basictags	0.999 exec/sec	7,274	1 sec	0 millis	0 millis

Related Topics ...

- Diagnostics - Logs

Diagnostics - Logs

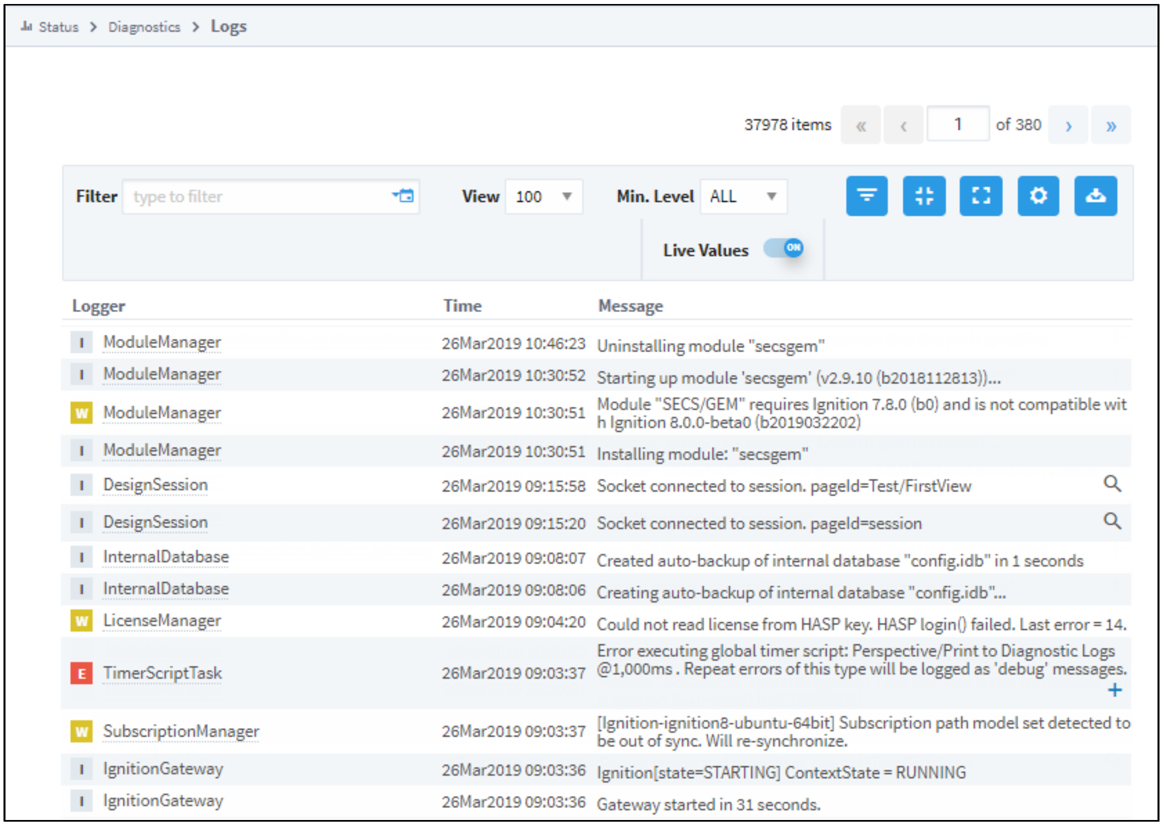
One of the most important troubleshooting tools of the Gateway Webpage is the Logs page. This console shows errors caused by Gateway events including things like Database or Device connections, Authentication profiles, Alarm Journals and Pipelines, and anything else that is Gateway scoped. You can find logs in the Status tab of the **Gateway Webpage** under **Diagnostics > Logs**.



On this page ...

The Gateway Logs also provide a wealth of information about the running state of the Gateway. To learn more, refer to the section on Troubleshooting the Gateway. The Logs page is where the Gateway Console is located that allows you to see a live flow of log events in the system.

Attributes	Description
Logger	Name of the logger that describes the context of the message.
Time	Time of log.
Message	Message for the log.
Filter	Search criteria to filter for specific tasks or events. Click the **Calendar** 📅 icon to set a filter date and time range. Options are Last Month, Last Week, Last 24 Hours, or Custom.
Min Level	Dropdown menu with options: All, Trace, Debug, Info, Warn and Error.
Live Values	Toggle switch to turn live values on or off.



Filter

View 100

Min. Level ALL

Live Values ON

	Logger	Time	Message
I	ModuleManager	26Mar2019 10:46:23	Uninstalling module "secsgem"
I	ModuleManager	26Mar2019 10:30:52	Starting up module 'secsgem' (v2.9.10 (b2018112813))...
W	ModuleManager	26Mar2019 10:30:51	Module "SECS/GEM" requires Ignition 7.8.0 (b0) and is not compatible wit h Ignition 8.0.0-beta0 (b2019032202)
I	ModuleManager	26Mar2019 10:30:51	Installing module: "secsgem"
I	DesignSession	26Mar2019 09:15:58	Socket connected to session. pageId=Test/FirstView
I	DesignSession	26Mar2019 09:15:20	Socket connected to session. pageId=session
I	InternalDatabase	26Mar2019 09:08:07	Created auto-backup of internal database "config.idb" in 1 seconds
I	InternalDatabase	26Mar2019 09:08:06	Creating auto-backup of internal database "config.idb"...
W	LicenseManager	26Mar2019 09:04:20	Could not read license from HASP key. HASP login() failed. Last error = 14.
E	TimerScriptTask	26Mar2019 09:03:37	Error executing global timer script: Perspective/Print to Diagnostic Logs @1,000ms . Repeat errors of this type will be logged as 'debug' messages. +
W	SubscriptionManager	26Mar2019 09:03:37	[Ignition-ignition8-ubuntu-64bit] Subscription path model set detected to be out of sync. Will re-synchronize.
I	IgnitionGateway	26Mar2019 09:03:36	Ignition[state=STARTING] ContextState = RUNNING
I	IgnitionGateway	26Mar2019 09:03:36	Gateway started in 31 seconds.

Changing Logging Levels

The logs can be filtered by using a search term or by date.

1. To change logging levels, go to the Gateway's **Status > Diagnostics > Logs** page and click on the **Settings** ⚙ icon.
2. On the Log Configuration popup that will open, search for the logger name in the Filter box, and select the desired logging level from the dropdown to the right of the logger name

Downloading the Logs

Gateway logs can also be exported using the **Download Logs** ⤓ icon the right.

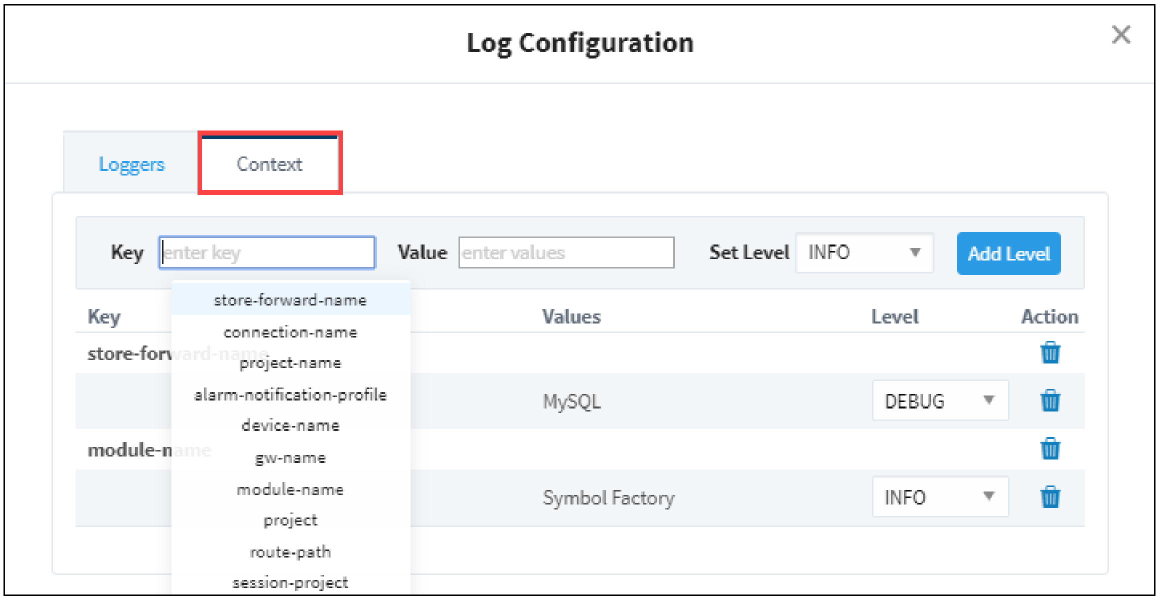
Printing to the Logs

You can print to Gateway logs by using the system.util.getLogger() function. Below is a simple example, you can look in the appendix for more options.

```
logger = system.util.getLogger("My Logger Name")
logger.warn("My Warning Message")
```

Mapped Diagnostic Context Keys

Mapped Diagnostic Context Keys (MDC Keys) allow you to specify a specific context, such as a particular project, and then set a logging level for it. This will set all loggers that pertain to the specified project to the logging level. This is useful to help diagnose an issue with a specific system within the Gateway. To use MDC Keys, click the **Settings** ⚙ icon on the Logs page and navigate to the Context tab. Here, you can choose specific Key-Value pairs that match a particular system. Selecting the Key text field will bring up a list of possible keys in your system. Once a Key has been selected, selecting the Value text field will bring up a list of possible values that relate to the selected Key.

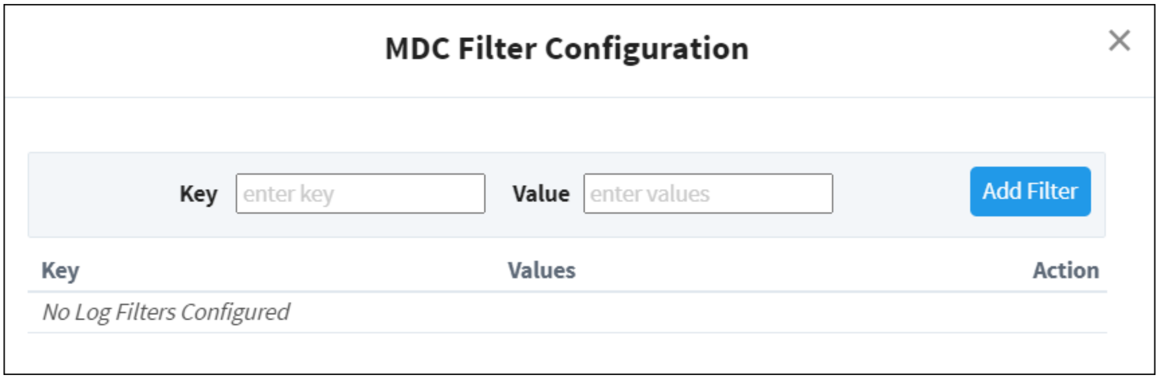


Log Configuration

Key	Values	Level	Action
store-forward-name			
	MySQL	DEBUG	
module-name			
	Symbol Factory	INFO	

Filter by Mapped Diagnostic Context Key

The Logs can also be filtered to show only logs that pertain to an MDC Key. This can be useful when altering the logging level of one or more MDC Keys. To set an MDC filter on the logs, simply click the **Add Mapped Diagnostic Context Filter** icon on the Logs page to open a window where MDC Key filters can be set. The filter can be specified for all values in a Key by not specifying a Value.

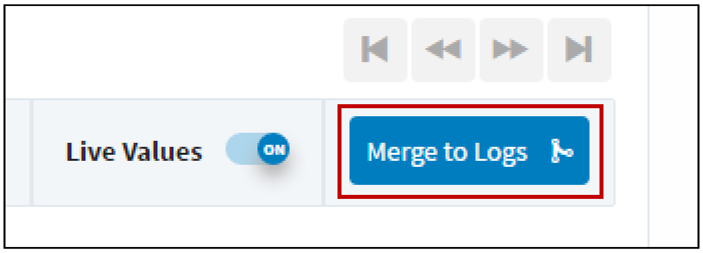


MDC Filter Configuration

Key	Values	Action
No Log Filters Configured		

Merging to Logs

Many of the other pages in the Status tab also contain a logger that will just show logs that are relevant for that particular section. These log views all contain a button on the right side called **Merge to Logs**. This button allows the current filter to be applied to the main logs page, allowing you to still see only the logs for that section but use all of the tools available.



Related Topics ...

- Diagnostics - Execution

- Diagnostics - Running Scripts
- Diagnostics - Threads

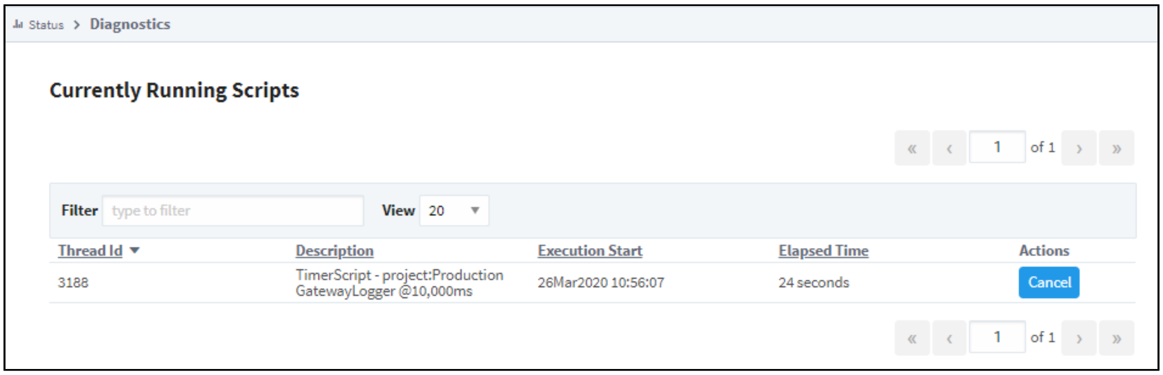
In This Section ...

Diagnostics - Running Scripts

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

The Running Scripts page shows all actively running Gateway scripts, as well as providing a way to terminate any running script. In addition, the Vision client and Designer consoles have a Running Scripts tab, which also lists running scripts and provides a way to terminate them.

Attributes	Description
Thread Id	Thread Id number.
Description	Name and description of the script.
Execution Start	Time script started running.
Elapsed Time	Amount of time the script has been running.
Actions	Ability to Cancel the running script.



Thread Id	Description	Execution Start	Elapsed Time	Actions
3188	TimerScript - project:Production GatewayLogger @10,000ms	26Mar2020 10:56:07	24 seconds	Cancel

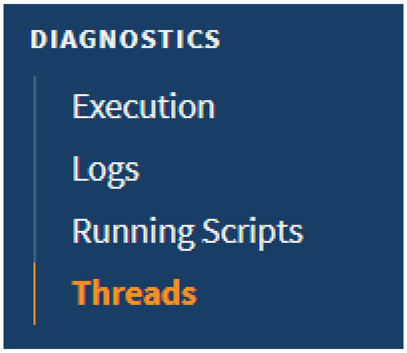
Related Topics ...

- Diagnostics - Execution
- Diagnostics - Logs
- Diagnostics - Running Threads

Diagnostics - Threads

The Threads page displays a snapshot of information about Gateway's execution threads. This information may be important when troubleshooting certain types of issues. If you are working with Support, you may be asked for a thread dump.

Their state and CPU usages are displayed to easily find problem threads, as well as a chart of what systems are using the threads and the ability to filter the threads based on a keyword. Each thread can be expanded to give details on what it is currently doing, and that individual thread process can be copied to the clipboard.



On this page ...

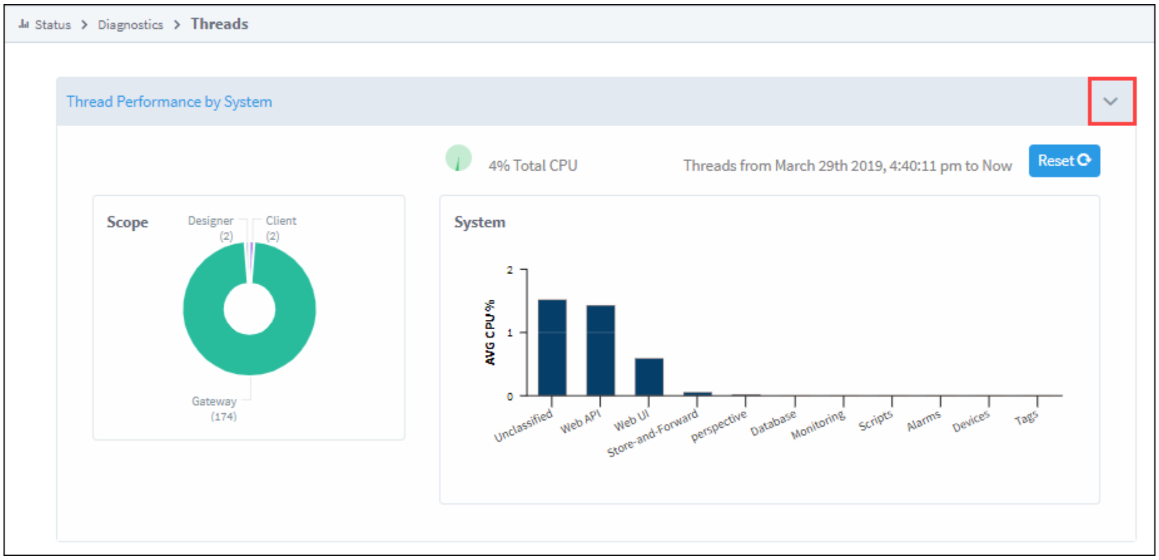
- Thread Performance by System
- Individual Threads

Thread Performance by System

The upper section of the Diagnostics - Threads page depicts graphical representation of the current threads including percentage of total CPU used, overall scope, and average CPU usage per system.

Click the **Expand** ‹ icon to see the display.

Click the **Reset** Reset ⟳ icon to reset the information.

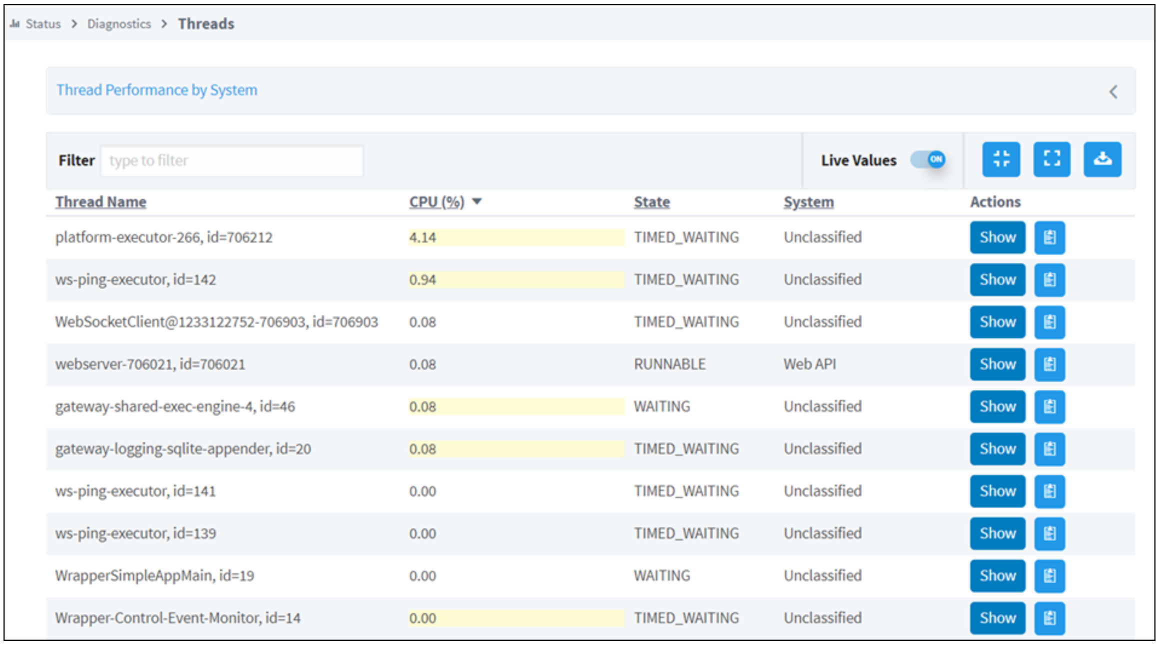


Individual Threads

The lower portion of the Diagnostics - Threads page displays statistics for individual threads.

To download a thread dump, click the **Download** icon on the right of the page. In order to be useful, a thread dump must be taken at the time the issue is observed. Unlike logs, which maintain a record of past messages, a thread dump is a time-specific snapshot and will not contain a record of past states.

Attributes	Description
Thread Name	Name of the thread.
Filter	Search criteria to filter for specific threads.
CPU(%)	Percentage of CPU usage.
State	Current state / status of the system.
System	Search criteria to filter for specific tasks or events.
Actions	View thread details.
Live Values	Toggle switch to turn live values on or off.



Status > Diagnostics > Threads

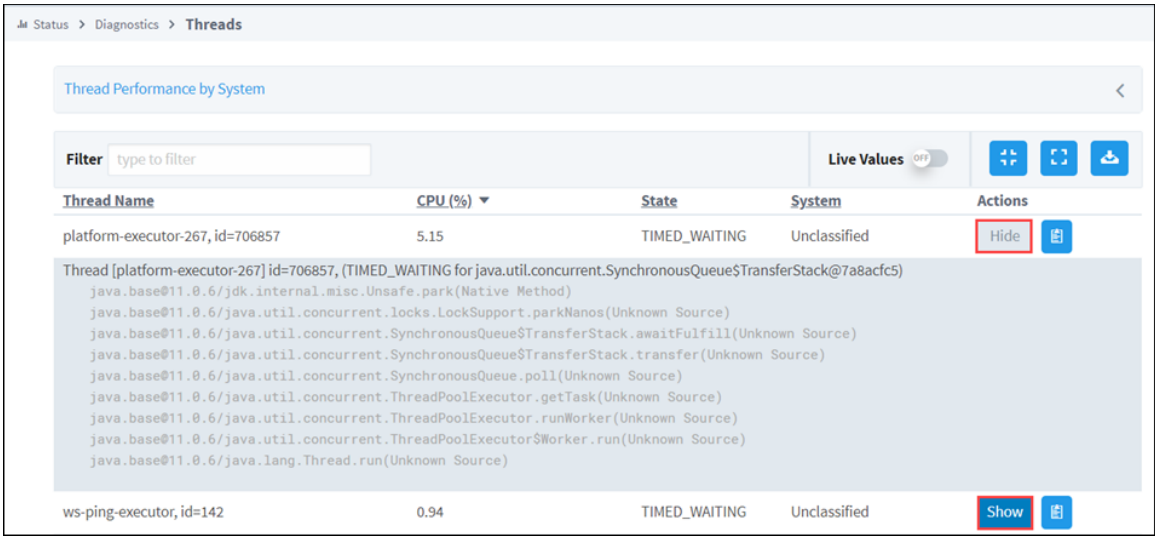
Thread Performance by System

Filter: type to filter

Live Values: ON

Thread Name	CPU (%) ▾	State	System	Actions
platform-executor-266, id=706212	4.14	TIMED_WAITING	Unclassified	Show 📋
ws-ping-executor, id=142	0.94	TIMED_WAITING	Unclassified	Show 📋
WebSocketClient@1233122752-706903, id=706903	0.08	TIMED_WAITING	Unclassified	Show 📋
webserver-706021, id=706021	0.08	RUNNABLE	Web API	Show 📋
gateway-shared-exec-engine-4, id=46	0.08	WAITING	Unclassified	Show 📋
gateway-logging-sqlite-appender, id=20	0.08	TIMED_WAITING	Unclassified	Show 📋
ws-ping-executor, id=141	0.00	TIMED_WAITING	Unclassified	Show 📋
ws-ping-executor, id=139	0.00	TIMED_WAITING	Unclassified	Show 📋
WrapperSimpleAppMain, id=19	0.00	WAITING	Unclassified	Show 📋
Wrapper-Control-Event-Monitor, id=14	0.00	TIMED_WAITING	Unclassified	Show 📋

Click the **Show** Show icon to display additional details for a single thread. You can click the **Clipboard** 📋 icon to copy that thread to the clipboard.



Thread Name	CPU (%)	State	System	Actions
platform-executor-267, id=706857	5.15	TIMED_WAITING	Unclassified	Hide
ws-ping-executor, id=142	0.94	TIMED_WAITING	Unclassified	Show

```
Thread [platform-executor-267] id=706857, (TIMED_WAITING for java.util.concurrent.SynchronousQueue$TransferStack@7a8acfc5)
    java.base@11.0.6/jdk.internal.misc.Unsafe.park(Native Method)
    java.base@11.0.6/java.util.concurrent.locks.LockSupport.parkNanos(Unknown Source)
    java.base@11.0.6/java.util.concurrent.SynchronousQueue$TransferStack.awaitFulfill(Unknown Source)
    java.base@11.0.6/java.util.concurrent.SynchronousQueue$TransferStack.transfer(Unknown Source)
    java.base@11.0.6/java.util.concurrent.SynchronousQueue.poll(Unknown Source)
    java.base@11.0.6/java.util.concurrent.ThreadPoolExecutor.getTask(Unknown Source)
    java.base@11.0.6/java.util.concurrent.ThreadPoolExecutor.runWorker(Unknown Source)
    java.base@11.0.6/java.util.concurrent.ThreadPoolExecutor$Worker.run(Unknown Source)
    java.base@11.0.6/java.lang.Thread.run(Unknown Source)
```

Related Topics ...

- Diagnostics - Execution
- Diagnostics - Logs
- Diagnostics - Running Scripts

Config

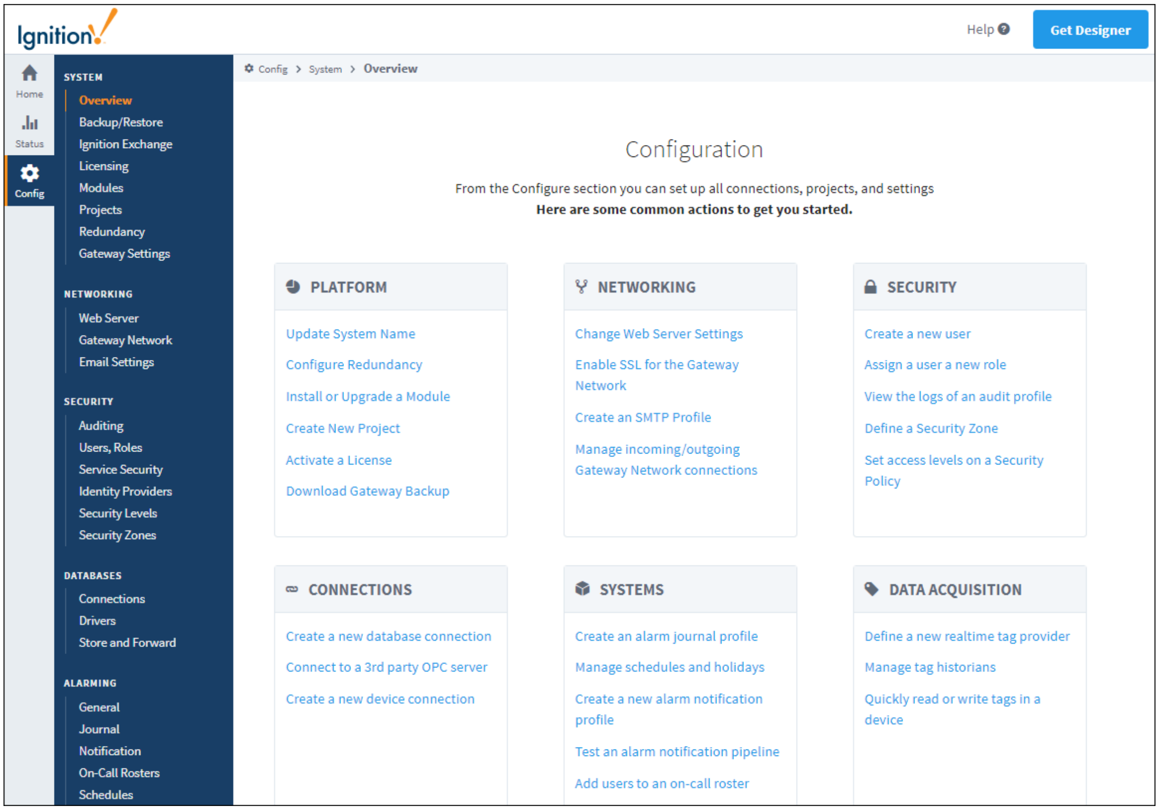
Configuring the Gateway

On this page

...

The Config tab provides access to configuration options for Gateway settings. This is where most of the settings that affect the whole Gateway are set up. We can add database and device connections, users and roles, adjust alarm settings, set up security, and create a schedule for a Gateway backup to be taken automatically at specific times.

The list of Config options on the left menu change based on what modules are installed on your Gateway. Third-party modules have settings that are not discussed on this page.



From the Configure section you can set up all connections, projects, and settings

Here are some common actions to get you started.

System

The System section is a sort of catch all section that can do a lot of different things. The first window is the Overview page, which is what will first show up when navigating to the Configure tab. The Overview page only has links to other sections of the Config tab, but it is useful if you aren't exactly sure where the setting you are looking for is located, because the Overview page lists the most common configuration changes.

Page	Description
Backu	The Backup/Restore page is where you can manually take a backup, and restore a previous backup. It also can upgrade a

p/Restore	legacy backup.
Ignition Exchange	The following feature is new in Ignition version **8.0.5** Click here to check out the other new features On the Ignition Exchange page you can access the Ignition Exchange to browse for resources. You can also import an Ignition Exchange Package that you've downloaded.
Licensing	The Licensing page allows you to control any and all licenses currently activated on the Gateway. It will show all the modules that the license is currently good for, and have some options for activating a new or additional license.
Modules	The Modules page displays all currently installed modules and if they are active or not. This is where you can install a new module or upgrade an existing module.
Projects	The Projects page can easily manage all of the projects currently configured in the Gateway, with settings that change the Name and Title, or the default database and authentication profile of the project.
Redundancy	The Redundancy page is where all of Ignition's redundancy settings are configured. This is where the master and backup nodes are configured as well as the network settings to make sure the two nodes can properly communicate.
Gateway Settings	The Gateway Settings page is where all miscellaneous settings are located. Autobackups can be configured here to happen at certain times. SSL encryption can be turned on here as well as changing roles required to access certain parts of the Gateway for increased security.

Networking

The Networking section deals with setup and management of the Gateway Network.

Page	Description
Web Server	The following feature is new in Ignition version **8.0.3** Click here to check out the other new features The Web Server page is for configuring the http and https ports, setting up the SSL / TLS certificate, redirecting traffic through a known address, and whether or not all http traffic should be forcefully redirecting to https. If you are allowing users to access your Gateway from outside your network (through the internet), you will need to configure the Public HTTP Address settings. You can find out more about SSL in Using SSL.
Gateway Network	The Gateway Network allows you to connect multiple Gateways together over a wide area network. The Gateway's connection settings can also be changed to only allow certain connections. The Gateway Settings set the basic rules for the system.
Email Settings	The Email Settings section allows you to create an SMTP server connection in the Gateway that can be used by several different resources such as Alarm Notification and Report Schedules.

Security

Ignition provides serveral Security options to safeguard data and applications in Ignition which fall into the following categories, and are set up in the Config section.

Page	Description
Auditing	Here you can set up an Audit Profile to record details about specific events that occurred.
Users	This is where you set up users and roles. Security policies are defined in terms of the roles set up in the system.

and Roles	
Service Security	A security policy can be defined for each Security Zone and is set up in Service Security. The Security Policy has four sections: Alarm Notification, Alarm Status, History Provider Access, and Tag Access. They work together to define how the local Gateway gives access to incoming Gateway connections.
Identity Providers	Identify Providers (IdP) provides a way for users to log in to Ignition using credentials stored outside of Ignition. An IdP creates, maintains, and manages identity (login) information while providing authentication services to Ignition. This provides a secure login that allows Ignition to use SSL and two-factor authentication (2FA).
Security Levels	Security Levels define a hierarchy for access inside a Perspective Session. This authorization system provides a way to map roles from an Identity Provider (IdP) to Ignition roles.
Security Zones	A Security Zone is a list of Gateways, Computers, or IP addresses that are defined and grouped together. This group now becomes a zone on the Gateway Network, which can have additional policies and restrictions placed on it. Security Zones provide this functionality to the Gateway Network, limiting locations instead of people to be read-only for specific actions. This allows for greater control over the type of information that is passing over the network, improving security and helping to keep different areas of the business separate, while still allowing them to interconnect.

Databases

The Databases section is where database connections are set up. Databases are used in historical data logging, reporting, storing alarm logs, and Tag storage.

Page	Description
Connections	Many of the advanced features of Ignition, such as the Transaction Groups and Tags Historian require a connection to an external database, and most databases require special permissions for each computer that wants to connect. Ignition takes care of all of this. You create a connection to your database once, and every system in Ignition will use that central connection. From here, you can create new database connections and edit existing connections.
Drivers	JDBC drivers used in database connections are imported and configured in Drivers of the Database section.
Store and Forward	The Store-and-Forward system provides a reliable way for Ignition to store data to the database. The Store-and-Forward system settings offer a good deal of flexibility in tuning. Different types of situations and goals will likely require different configurations.

Alarming

The Alarming section provides general alarm configuration settings to provide up-to-date status of alarms, store alarm history, build the logic for how, why, and when alarm notifications are delivered, manage alarm notifications for user groups, and send Email, SMS, or Voice notifications. With all these features and functions in Alarming, you can easily create alarms, and design and manage your alarm notifications any way you choose.

Page	Description
General	This General setting provides of some basic alarm configuration settings.
Journal	Alarm Journals are configured in the Alarming section to store basic historical information in a database about alarms that occurred, such as their source and timestamp, associated data on the alarm, and the values of the alarm's properties at the time the event occurred.
Notification	Notification Profiles are configured in the Alarming section to allow for Email, SMS, or Voice notifications to be sent out when an alarm event occurs.
	The On-Call Roster is where you create user groups to be notified when an alarm occurs. When an alarm is triggered, it is sent to a designated On-Call Roster where it evaluates the users schedules, and only notifies those users that have an active schedule. Users that are off-schedule will not be notified.

On-Call Rosters	
Schedules	Defines the times of users on-call availability and unavailability by configuring Schedules.

Tags

The Tags section is where both Realtime Tag Providers as well as Historical Tag Providers are configured. Note that this is not where individual Tags are set up.

Page	Description
History	Configure the settings for each of the configured Historical Tag Providers, or create new remote or split providers.
Realtime	Configure the settings for each of the configured Realtime Tag Providers,or create new standard or remote providers.

OPC Client

The OPC Client section is where connections from Ignition's internal OPC UA server to other OPC servers are located.

Page	Description
OPC Connections	Configure OPC Connections to Ignition's built in OPC Server.
OPC Quick Client	The OPC Quick Client allows for quick and simple testing of any OPC Connections connected to the OPC server.

OPC UA

The OPC UA Server is where Ignition's internal OPC UA server is configured.

Page	Descriptions
Device Connections	This is where all device connections to our internal OPC UA server are configured.
Security	Upload and trust client and server OPC UA certificates.
Server Settings	Configure Ignition's internal OPC UA Server's settings.

Enterprise Administration

The Enterprise Administration section controls the majority of EAM functions. Setting the Gateway to be a Controller or Agent as well as creating Agent Tasks and managing the various Agents from the Controller Gateway can all be done in this section.

Sequential Function Charts

A Sequential Function Chart (SFC) is a series of scripts that are defined in a single location and then called in sequential order.

Page	Description
Settings	This is where the SFC Settings are configured.

Related Topics ...

- Gateway Settings
- Security
- Database Connections
- Tags
- Alarming
- OPC UA
- Enterprise Administration
- Sequential Function Charts

In This Section ...

Gateway Settings

On this page ...

Gateway Configuration

Once you have the Gateway up and running, you start by configuring some or all of the general services in Ignition. You make the configuration changes from the **Config** section of the Gateway Webpage. The different broad categories of what you can configure are as follows:

- **System** (Overview, Backup/Restore, Licensing, Modules, Projects, Redundancy and Gateway Settings)
- **Networking** (Web Server, Gateway Network and Email Settings)
- **Security** (Auditing ,Users, Roles, Service Security, Identity Providers, Security Levels, and Security Zones)
- **Databases** (Connections, Drivers, and Store and Forward)
- **Alarming** (General, Journal, Notification, On-Call Rosters, and Schedules)
- **Tags** (History and Realtime)
- **OPC Client** (OPC Connections and OPC Quick Client)
- **OPC UA** (Device Connections, Security, and Server Settings)
- **Enterprise Administration** (Event Thresholds, Controller Settings, Agent Management, License Management, and Agent Tasks)
- **Sequential Function Charts** (Settings)

Depending on what modules you have installed, some categories may be missing.

Important Gateway Settings

The Gateway is at the heart of the Ignition software. It runs as a web server and you can access it through a web browser. Once running, you can get various status information about the Gateway and access important functions. You can customize the Gateway's Homepage to fit your needs. Here's a description of all the settings and list of all the their associated properties that you will need to setup based on your Gateway configuration and preferences.

Required Roles

There are several properties that are required to access certain sections of the Gateway.

Property	Description
Gateway Config Role (s)	Allows you to enter roles to control access to the Gateway Config section.
Status Page Role(s)	Allows you to enter roles to control access to the Gateway Status section.
Home Page Role(s)	Allows you to enter roles to control access to the Gateway Home section.
Designer Role(s)	Allows you to enter roles to control access to the Designer.

INDUCTIVE UNIVERSITY

Locking the Gateway

Watch the Video

Local Client Fallback Settings

Ignition provides a Local Vision Client Fallback mechanism that lets you use a Gateway running on the local machine. If the Gateway is lost, the Client can automatically retarget to a project that you specify in the local in the local Gateway.

Scheduled Backups

The **Scheduled Backup Settings** control the Gateway's scheduled backup system. This system is capable of automatically making a Gateway backup and storing it to a folder path, which can be a network path.

When you enable this system, you must specify a destination folder. This can be a local folder, for example `C:\backups` or `/var/backups` , or a network path such as `\\fileserver\backups`.

The scheduled backup system works on a schedule that is specified using **UNIX Crontab** syntax. T his is a standard format for specifying a basic schedule. The format consists of five space-separated fields, one for minute, hour, day-of-month, month, and day-of-week. The special Gateway Configuration 72 character ***** means **all**. Slashes can be used to indicate that values should be stepped, for example, */5 in the minutes field means "every 5 minutes", or 0:00, 0; 05, 0:10, etc.

To learn more on how to schedule Gateway Backups, refer the page on Gateway Scheduled Backups.

Gateway Audit Profile

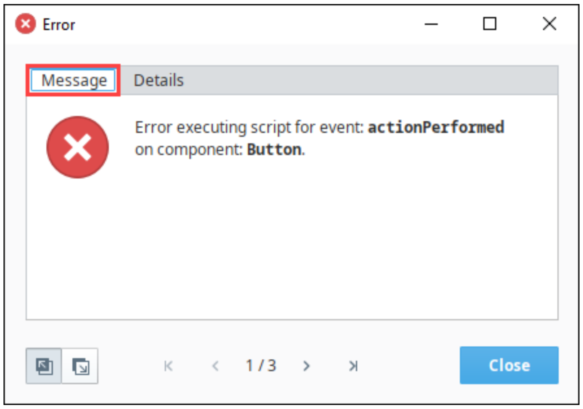
The auditing system in Ignition records actions originating from the Gateway, Perspective and Vision projects. For a list of actions that are logged by the auditing system, see Auditing Actions Reference.

Gateway Scripting Project

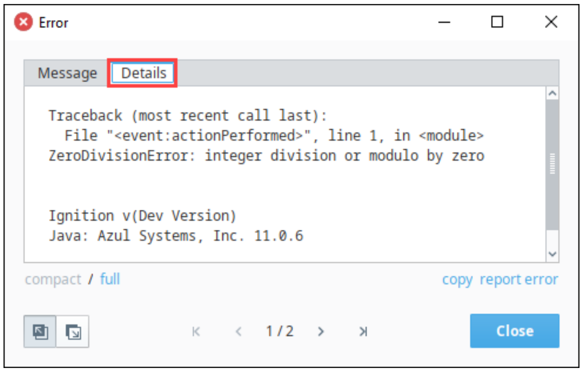
The Gateway Scripting Project is a Project in which Gateway-scoped scripts with no project affiliation can access user script libraries.

Error Reporting Settings

When an error occurs in the Client or Designer, the users can click a link on the Details tab to report the error via email.



Error executing script for event: **actionPerformed**
on component: **Button**.



```
Traceback (most recent call last):
  File "<event:actionPerformed>", line 1, in <module>
ZeroDivisionError: integer division or modulo by zero


Ignition v(Dev Version)
Java: Azul Systems, Inc. 11.0.6
```

These settings define how the errors are reported.

Multicast Settings

These properties allow the Gateway to broadcast information about itself via multicast UDP packets. This allows the Gateway to be discoverable by any components that are also listening to the same multicast address. For example, native client launchers listen on a multicast address to provide a list of available Gateways on the network. Verify that the send ports and receive ports are open on the Gateway machine in order to be able to broadcast multicast message.

Gateway Property Reference

After you launch the Gateway, you can define the high-level settings that apply to the entire Gateway by going to the **Config > Gateway Settings** on the Gateway Webpage. From this page, you can use the default values or define a new setting.

Gateway Settings	
System Name	Is a unique name for this Ignition installation. It is used to distinguish this server from others on the network when working with multiple Ignition installations. **Caution:** It is not recommended to change the System Name on a Gateway that is in production. Doing so can cause subsystems that reference the name, such as the Tag Historian system, to malfunction.
System User Source	Is the user source used to secure access to the Gateway, as well as to the Designer.
Gateway Config Role (s)	Is a comma-separated list of roles, one of which is required to log into the Gateway's configuration section. These roles must be defined in the System User Source.
Status Page Role (s)	The required roles to access the Gateway's **Status** section. Leave blank to remove security restrictions for this section.
Home Page Role (s)	The required roles to access the Gateway's **Home** section. Leave blank to remove security restrictions for this section. Note that this is only used to limit the access to the homepage itself, each project has its own user source for limiting access to the runtimes.
Designer Role(s)	Users must belong to one of these roles in order to have access for logging into the Designer. Multiple roles can be specified by separating them with commas.
Create Project Role(s)	Users must belong to one of these roles in order to create a new Designer project. Multiple roles can be specified by separating them with commas.
Allow User Admin	Allows the administration of the Gateway's system user source from the Designer and Client. Unless this is enabled, anything outside of the Gateway Configure page is prevented from altering the Gateway's system user source.
Allow Designer SSO	Allows single-sign-on authentication for logging into the Designer if the Gateway user source supports it.
Persist Alarms	Whether or not alarm properties such as acknowledgment should persist across Gateway restarts.
Homepage Redirect URL	The URL this gateway will redirect to when http://ip:port/ is visited. Can either be a relative path (e.g., /web/home), or fully qualified (e.g., https://inductiveautomation.com).
Use SSL	This feature was removed from Ignition in version 8.0.3

	As of version 8.0.3 this setting has been moved to the **Networking > Web Server** page. Forces the clients to use SSL encrypted communication when talking to the Gateway. It is highly recommended that you purchase and install a genuine SSL certificate if you use this option. See Installing a Genuine SSL Certificate.
Gateway Scripting Project	The name of the Project in which Gateway-scoped scripts with no project affiliation can access user script libraries.
Gateway Audit Profile	The following feature is new in Ignition version **8.0.8** Click here to check out the other new features The name of the audit profile that Gateway-scoped actions will log to. See Auditing Actions Reference.

Launch Settings

Designer Memory	The maximum amount of memory that the Designer has access to.
Disable Direct3D	Disables the Direct3D rendering pipeline launched clients. Direct3D can cause performance problems with XOR painting. (affects clients on Windows only)
Disable DirectDraw	Disables the DirectDraw system for launched clients. Can be useful for some video cards that don't support DirectDraw well. (affects clients on Windows only)

Local Vision Client Fallback

Enable Local Fallback	Enables a client to fall back to a project in a local Gateway if communication is lost to the central Gateway. Note that port 6501 must be open on the local machine.
Seconds Before Failover	The number of seconds to wait before switching to the local Gateway project after communication loss.
Fallback Project	The local project to use during fallback.

Scheduled Backup Settings

Enable Scheduled Backups	Enables the scheduled backup system which automatically makes backups at a scheduled time.
Backup Folder	A path to a folder in which to put the scheduled backups.
Backup Schedule	A UNIX 'crontab' format scheduling string representing when to make the backups.
Retention Count	The number of backups to keep in the backup folder.

Error Reporting

SMTP Server	When not blank, user-reported errors are emailed using this SMTP server.
To Address	The email address(es) that will receive the error notification. Separate multiple email addresses with a semicolon (;)
From Address	The email address that the error notification is from.
SMTP Username	A username for the SMTP server, if required.
Change Password?	Check this box to change the existing password.

Password	A password for the SMTP server, if required.
Password	Re-type password for verification.
Multicast Settings	
Enable Multicast	Allows this Gateway to be discoverable on your local network.
Multicast IP Address	Gateway messages are broadcast on this address.
Send Port	This port must be open on this machine to send multicast messages.
Receive Port	This port must be open on any machine that will receive multicast messages.
Message Interval	The interval in milliseconds at which multicast messages will be sent.

Related Topics ...

- Config
- Status
- Gateway Security
- Gateway Command-line Utility

Email Settings

Gateway-Wide SMTP Server

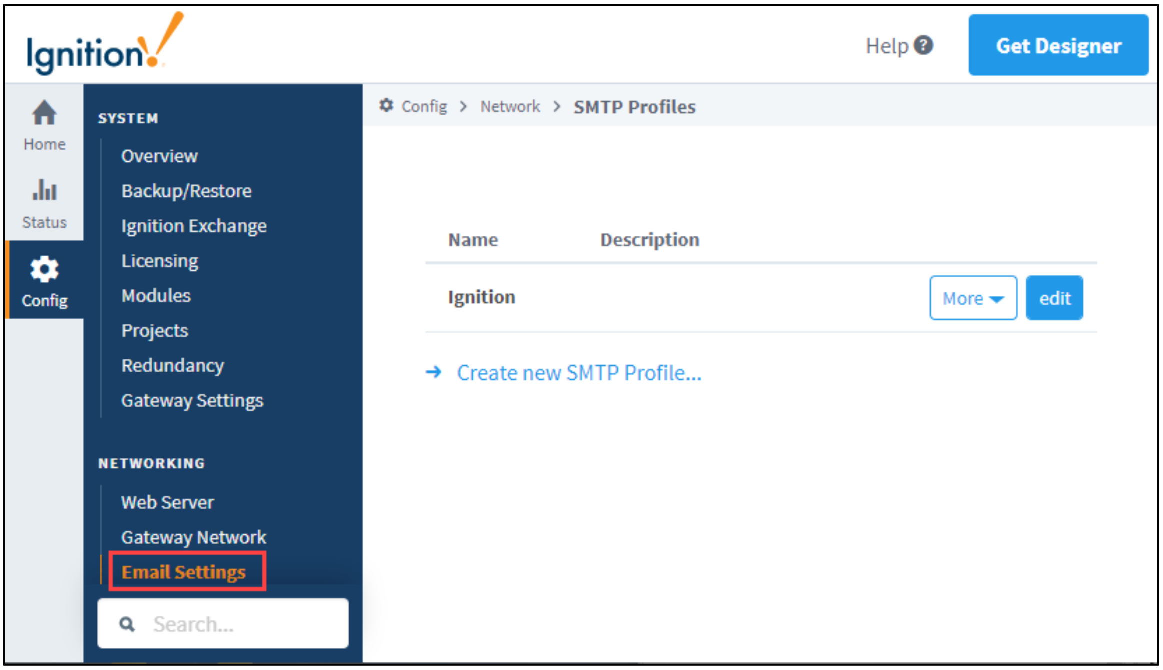
In Ignition, there are several places that you might want to send an email from. Instead of setting up a new email server connection at each one, you can add SMTP server connections in the Gateway Config section and reference them in other places. For example, you can use a pre-configured connection in any of these places:

- Alarm Notification - While the alarm notification system allows you to set up an Email Notification Profile that is separate from the SMTP Profile, you can instead choose to use the settings configured in the SMTP Profile.
- Report Schedules - When Scheduling an Email Action in a report, the Action requires a configured SMTP Profile before the report can be emailed out.
- system.net.sendEmail() - Instead of manually entering in values for the SMTP server inside each of your scripts, this function can instead use the settings from an existing SMTP Profile.

Once your SMTP Profile is set up, you only need to reference the name of the connection and Ignition will take care of the rest.

Setting up an SMTP Server

The Email Settings page allows you to configure an SMTP server connection that can be used by several different resources in the Gateway. This means that instead of setting up an SMTP server connection within each resource (i.e., Alarm Notification Profile, Reports, etc.), you can configure them once here, and have those resources all use the same SMTP server. You can find the Email Settings in the **Config** section of the Gateway under the Networking heading. Here, you can create a new SMTP Profile, or manage your existing profiles.



Name	Description
Ignition	

SMTP Profile Settings

Below is a list of properties available on an SMTP server connection.

Property Name	Property Description
Name	The name of the SMTP profile.

Description	A description of the SMTP profile.
Hostname	Hostname of the SMTP server to send email through.
Port	Port SMTP service is running on. Default is 25.
Use SSL /TLS	Connect using dedicated SSL/TLS. Default is false.
Use STARTTLS	Enable use of the STARTTLS command, allowing the connection to be upgraded to an SSL or TLS connection if supported by the server. This is not necessary for connections that are already SSL/TLS. Default is false.
Username	The username the Gateway will use when authenticating against the mailserver. This is only required if the SMTP server expects authentication.
Password	The password the Gateway will use when authenticating against the mailserver. This is only required if the SMTP server expects authentication.
Password	Retype password for verification.
Advanced Properties	
SMTP Timeout	Timeout (in milliseconds) to use when connecting to, reading from, and writing to the SMTP server. Default is 10,000.
Debug Mode Enabled	Enable email session debugging. Information is printed to standard output (wrapper.log). Default is false.
SSL Protocols	The following feature is new in Ignition version **8.0.6** Click here to check out the other new features A comma separated list of protocols that will be allowed if connecting via SSL/TLS. Default is TLSv1.2.

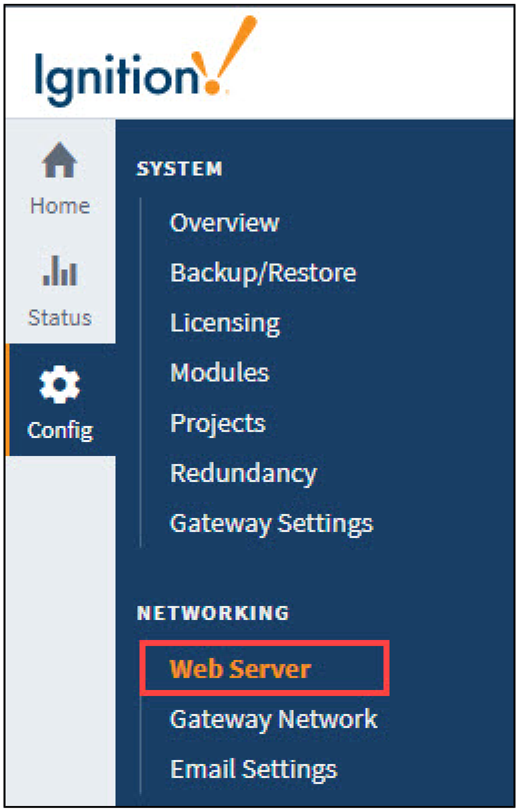
Web Server Settings

The following feature is new in Ignition version **8.0.3**
Click here to check out the other new features

On this page ...

The Web Server page is for configuring the HTTP and HTTPS ports, setting up the SSL / TLS certificate, redirecting traffic through a known address, and whether or not all HTTP traffic should be forcefully redirecting to HTTPS.

If you are allowing users to access your Gateway from outside your network (through the internet), you will need to configure the Public HTTP Address settings.

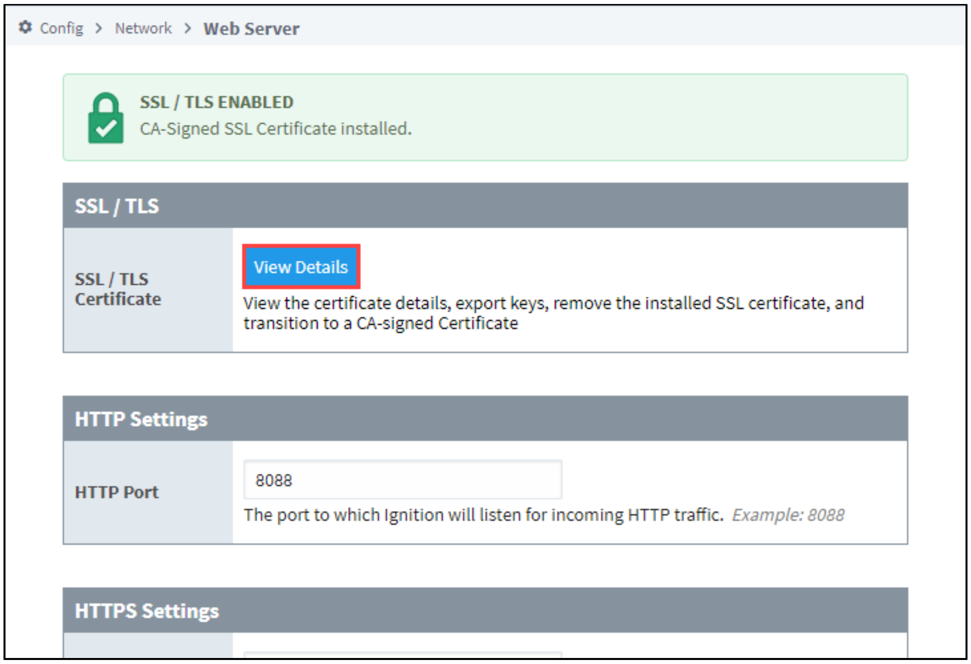


SSL/TLS Settings

From the Gateway webpage, click on **Config > Networking > Web Server**.

On the Web Server screen you can view details of an SSL certificate details, export keys, remove the installed SSL certificate, and transition to a CA-signed Certificate.

1. To view the active SSL / TLS certificate, click on the **View Details** button.



Config > Network > **Web Server**

SSL / TLS ENABLED
CA-Signed SSL Certificate installed.

SSL / TLS

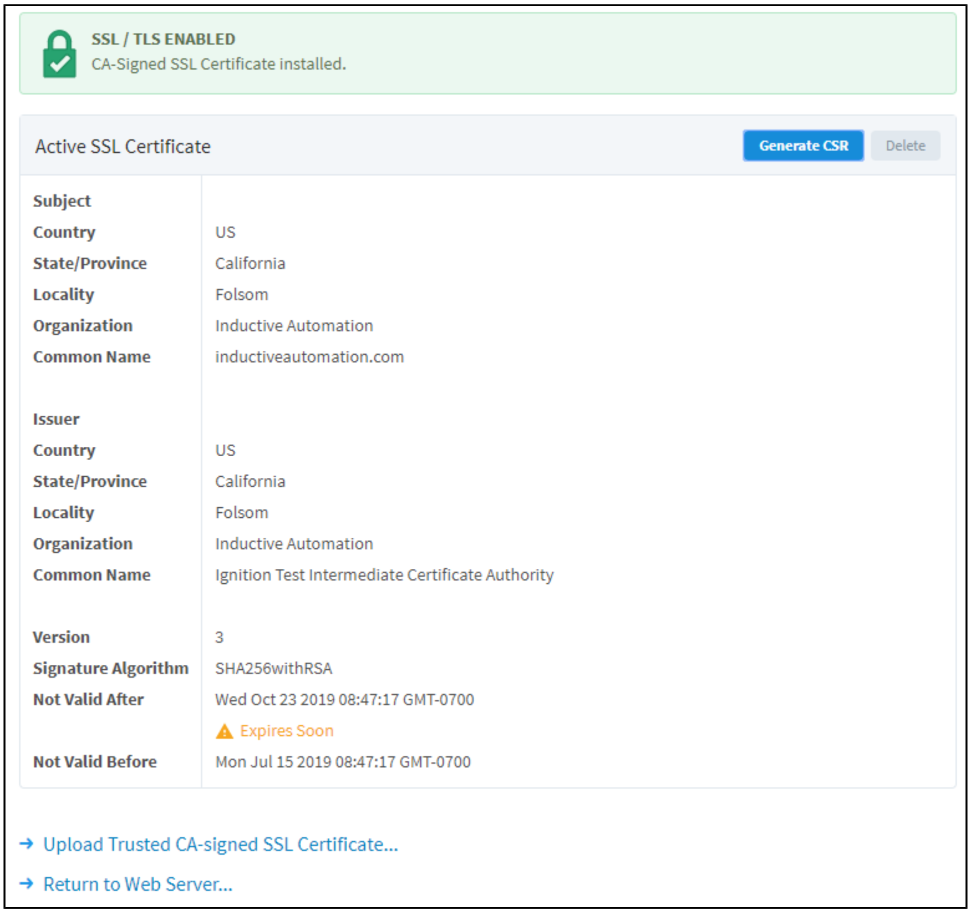
SSL / TLS Certificate	View Details View the certificate details, export keys, remove the installed SSL certificate, and transition to a CA-signed Certificate

HTTP Settings

HTTP Port	8088 The port to which Ignition will listen for incoming HTTP traffic. *Example: 8088*

HTTPS Settings

2. The Certificate Details are shown. To generate a Certificate Signing Request (CSR), click the Generate CSR button in the upper right.



SSL / TLS ENABLED
CA-Signed SSL Certificate installed.

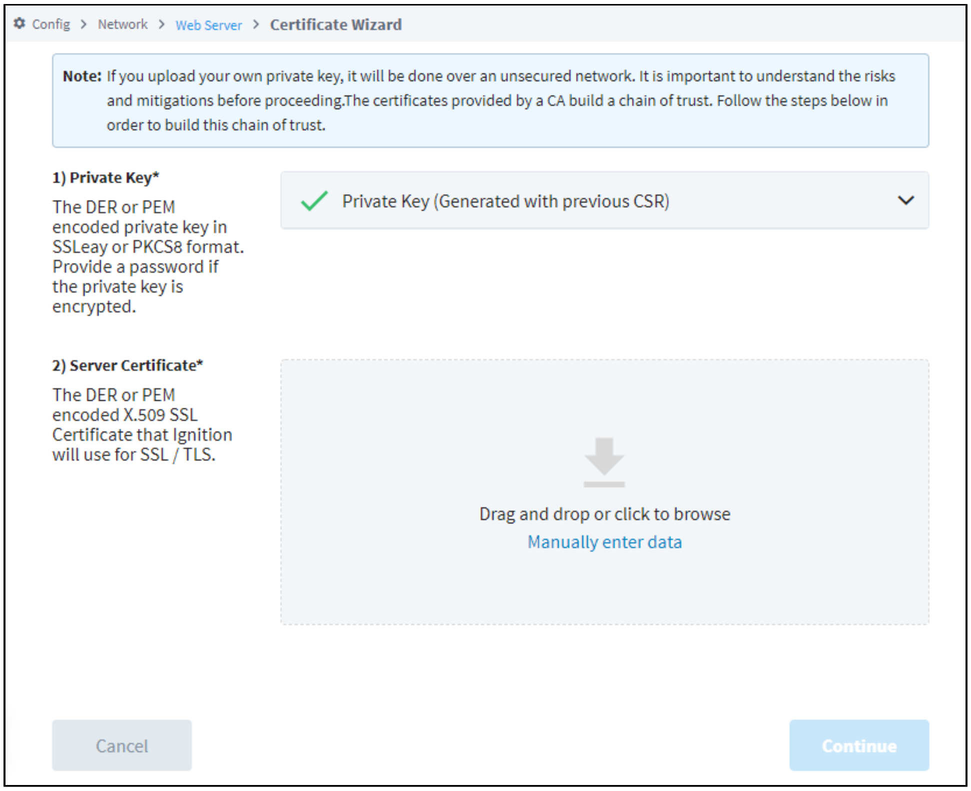
Active SSL Certificate

Subject	
Country	US
State/Province	California
Locality	Folsom
Organization	Inductive Automation
Common Name	inductiveautomation.com
Issuer	
Country	US
State/Province	California
Locality	Folsom
Organization	Inductive Automation
Common Name	Ignition Test Intermediate Certificate Authority
Version	3
Signature Algorithm	SHA256withRSA
Not Valid After	Wed Oct 23 2019 08:47:17 GMT-0700 ⚠ Expires Soon
Not Valid Before	Mon Jul 15 2019 08:47:17 GMT-0700

→ Upload Trusted CA-signed SSL Certificate...

→ Return to Web Server...

3. Ignition downloads the CSR. If you want to upload a new certificate, click the **Upload Trusted CA-signed SSL Certificate...** link. This takes you to the Certificate Wizard.


Config > Network > Web Server > Certificate Wizard
Drag and drop or click to browse
Manually enter data
Cancel
Continue

Note: If you upload your own private key, it will be done over an unsecured network. It is important to understand the risks and mitigations before proceeding.The certificates provided by a CA build a chain of trust. Follow the steps below in order to build this chain of trust.

1) Private Key*

The DER or PEM encoded private key in SSLeay or PKCS8 format. Provide a password if the private key is encrypted.

✓ Private Key (Generated with previous CSR)

2) Server Certificate*

The DER or PEM encoded X.509 SSL Certificate that Ignition will use for SSL / TLS.

4. The next step is to import the certificate chain. This gives you the Intermediate CA Certificate. Drag and drop the certificate file or bundle, browse for it, or manually enter the data.
   You'll see a message that the Intermediate CA Certificate was successfully uploaded. For more information, see Using SSL.

HTTP and HTTPS Settings

HTTP Settings	
HTTP Port	The port to which Ignition will listen for incoming HTTP traffic. Example: 8088
HTTPS Settings	
HTTPS Port	The port to which Ignition will listen for incoming HTTPS traffic. Example: 8043
Force Secure Redirect	When enabled, and if SSL / TLS is enabled, all http traffic will be redirected to its https counterpart. (Default: disabled)
Included Cipher Suites	Whitelist of included cipher suites for clients connecting to Ignition using SSL/TLS.
Excluded Cipher Suites	Blacklist of excluded cipher suites for clients connecting to Ignition using SSL/TLS. Takes precedence over allowed cipher suites.

HTTP and HTTPS Connectors Restart

Certain actions will cause the HTTP port and/or the HTTPS port to restart. Refer to the following table for details.

Configuration Change	HTTP Port Restarted?	HTTPS Port Restarted?
HTTPS Port	Yes	Yes
HTTPS Port	Yes	Yes
Force Secure Redirect	No	Yes

User Included Cipher Suites	No	Yes
User Excluded Cipher Suites	No	Yes
SSL/TLS Setup	No	Yes

Public HTTP Address settings

If you are allowing users to access your Gateway from outside your network (through the internet), you will need to configure the Public HTTP Address settings.

Public HTTP Address	
Auto Detect HTTP Address	To specify an explicit HTTP address that Vision Clients and Perspective Sessions will use, turn this off. Most users will leave autodetect on. (Default: enabled)
Public Address	The public facing address that Vision Clients and Perspective Sessions must use to connect. If Force Secure Redirect is enabled, redirected connections will use this address. Example: yourcompany.com
Public HTTP Port	The public facing HTTP port that Vision Clients and Perspective Sessions must use to connect .Example: 80
Public HTTPS Port	The public facing HTTPS port that Vision Clients and Perspective Sessions must use to connect. If Force Secure Redirect is enabled, redirected connections will use this port. Example: 443

Cipher Support

Below is a list of supported ciphers.

- TLS_AES_128_GCM_SHA256
- TLS_AES_256_GCM_SHA384
- TLS_CHACHA20_POLY1305_SHA256
- TLS_DHE_DSS_WITH_AES_128_CBC_SHA
- TLS_DHE_DSS_WITH_AES_128_CBC_SHA256
- TLS_DHE_DSS_WITH_AES_128_GCM_SHA256
- TLS_DHE_DSS_WITH_AES_256_CBC_SHA
- TLS_DHE_DSS_WITH_AES_256_CBC_SHA256
- TLS_DHE_DSS_WITH_AES_256_GCM_SHA384
- TLS_DHE_RSA_WITH_AES_128_CBC_SHA
- TLS_DHE_RSA_WITH_AES_128_CBC_SHA256
- TLS_DHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_DHE_RSA_WITH_AES_256_CBC_SHA
- TLS_DHE_RSA_WITH_AES_256_CBC_SHA256
- TLS_DHE_RSA_WITH_AES_256_GCM_SHA384
- TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256
- TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA
- TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256
- TLS_ECDH_ECDSA_WITH_AES_128_GCM_SHA256
- TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA
- TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA384
- TLS_ECDH_ECDSA_WITH_AES_256_GCM_SHA384
- TLS_ECDH_RSA_WITH_AES_128_CBC_SHA
- TLS_ECDH_RSA_WITH_AES_128_CBC_SHA256
- TLS_ECDH_RSA_WITH_AES_128_GCM_SHA256

- TLS_ECDH_RSA_WITH_AES_256_CBC_SHA
- TLS_ECDH_RSA_WITH_AES_256_CBC_SHA384
- TLS_ECDH_RSA_WITH_AES_256_GCM_SHA384
- TLS_EMPTY_RENEGOTIATION_INFO_SCSV
- TLS_RSA_WITH_AES_128_CBC_SHA
- TLS_RSA_WITH_AES_128_CBC_SHA256
- TLS_RSA_WITH_AES_128_GCM_SHA256
- TLS_RSA_WITH_AES_256_CBC_SHA
- TLS_RSA_WITH_AES_256_CBC_SHA256
- TLS_RSA_WITH_AES_256_GCM_SHA384

Gateway Backup and Restore

On this page

...

Gateway Backups vs Project Exports

It is a good idea to create backups anytime you make changes to Ignition. There are two main types of backups available in Ignition: Gateway backups and Project exports. You can choose to backup up everything, or be selective about what you want to backup and restore. Ignition is designed to be architecture agnostic or compatible across all platforms. This is particularly helpful when you want to restore your Gateway on a new machine or platform.

Gateway backups are all inclusive and Project exports are simply a backup of individual projects. Whether it is a Gateway backup or Project export, it's always good practice to create backups regularly when making changes to Ignition. This page discusses making Gateway backups and restoring from a Gateway backup. To learn more about project backups and restoring projects, refer to the Project Export and Import page.

Gateway Backup and Restore

Creating Gateway backups and restoring from a Gateway backup are super easy in Ignition. Gateway backups are all inclusive, and typically takes less than a minute to run. It includes everything you find in the Ignition Gateway Webpage. Everything gets backed up - all your projects, Gateway settings, authentication profiles, Tags, database connections, OPC and device connections, alarm pipelines, scripts, sequential function charts, reports, and Image Managment Library (i.e., png, jpg, and jpeg files). The only data that is not included in a Gateway backup is data stored in other programs such as SQL databases, PLC programs, other files you manually added to the install directory, and any additional files you may be using. This information needs to be backed up separately.

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

As of Ignition 8.0.14, Perspective custom assets such as fonts, icons, and themes are now included in a Gateway backup. In addition, any custom assets in a backup file will be restored when performing a Gateway restore.

You have the option of creating a Gateway backup and restoring a backup from the Gateway Webpage or Gateway Command-line Utility. We recommend using Gateway Webpage since it's much easier. Even better, is to set up scheduled backups to run on a regular schedule.

Project Backup vs. Ignition Gateway Backup

Watch the Video

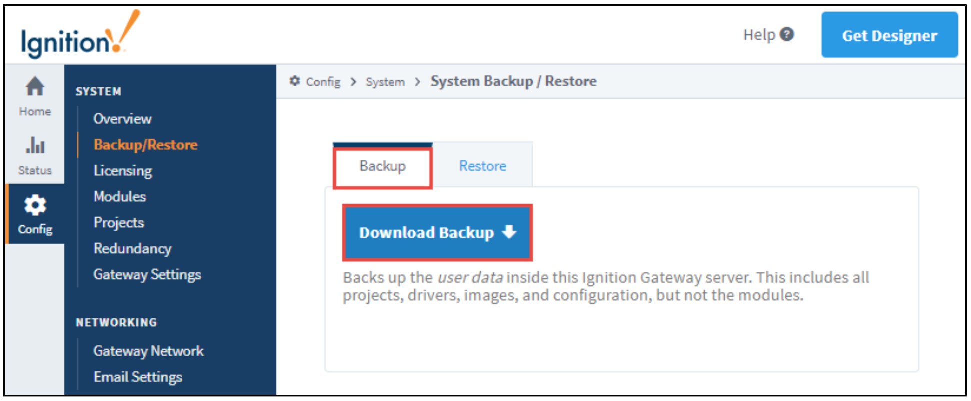
Making Ignition Gateway Backups

Watch the Video

Gateway Backup

The easiest way to create a backup of the Gateway is using the Gateway Webpage.

1. Go to the **Config** tab of the Gateway Webpage, and click on **System > Backup/Restore**.
2. The System Backup/Restore page will be displayed. Make sure the **Backup** tab is selected, then click **Download Backup.**



Backs up the *user data* inside this Ignition Gateway server. This includes all projects, drivers, images, and configuration, but not the modules.

3. By default, this downloads a `.gwbk` file extension to your local file system in your **Downloads** folder.

> The Gateway Backup default filename will look like **GatewayName_Ignition-backup-YYYYMMDD-HHMM.gwbk** where **YYYYMMDD-HHMM** is the timestamp of when it was created.

Command-line Utility

In Windows and Linux, you can use the command-line utility to create a Gateway backup. To run the Command-line Utility, open a shell and enter the command below.

> If you want to add a timestamp, you need to enter the date and time in the filename. Refer to the Command-line Utility page for a complete list of 'gwcmd' options.

```
gwcmd -b C:\Backups\Ignition\IgnitionBackup.gwbk
```

Gateway Restore

Restoring a Gateway backup is just as easy as backing it up and can also be done from the Gateway Webpage.

> When you perform a Gateway Restore, **ALL** of the server's current configuration will be permanently lost! Restoring a Gateway backup overwrites all of the existing settings including your projects. There is no merge option for a Gateway backup. We recommend you always make a backup of the existing server immediately before performing a Gateway Restore.

INDUCTIVE UNIVERSITY

Restoring Ignition Gateway Backups

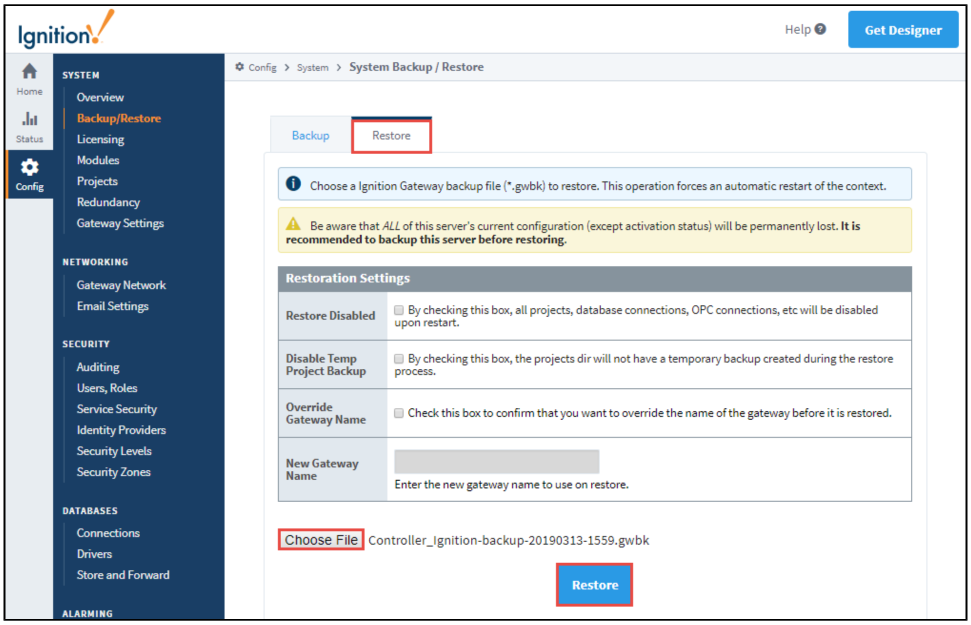
Watch the Video

1. Go to the **Config** tab and click on **System > Backup/Restore**.
2. The System Backup/Restore screen will be displayed. Click on the **Restore** tab. Read carefully through the restoration settings, and check the ones you wish to enable.

> **Restore Disabled Checkbox**
>
> If you check the **Restore Disabled** box, all projects, project resources, notifications, alarm journals, database connections, OPC connections, devices, scripts and third party modules will be disabled upon restart.

3. Click **Choose File**, then navigate to your Gateway backup file (***.gwbk**). By default, all your Gateway backup files are saved in your Downloads folder unless you select another folder location to choose an exisiting Gateway Backup file (**`*.gwbk`**) to restore.

4. Choose your Ignition backup file (**.gwbk**), and click **Open**.
5. Click **Restore** at the bottom of the System Backup/Restore screen. The Gateway stops while restoring the backup file. When restoring is complete, the Gateway restarts itself to apply the restored settings. The Gateway Webpage is refreshed and your projects are loaded and your whole Gateway is restored in less than a minute.



Choose a Ignition Gateway backup file (*.gwbk) to restore. This operation forces an automatic restart of the context.

Be aware that *ALL* of this server's current configuration (except activation status) will be permanently lost. **It is recommended to backup this server before restoring.**

Restoration Settings	
Restore Disabled	□ By checking this box, all projects, database connections, OPC connections, etc will be disabled upon restart.
Disable Temp Project Backup	□ By checking this box, the projects dir will not have a temporary backup created during the restore process.
Override Gateway Name	□ Check this box to confirm that you want to override the name of the gateway before it is restored.
New Gateway Name	Enter the new gateway name to use on restore.

Controller_Ignition-backup-20190313-1559.gwbk

The following table describes different Gateway restoration settings.

Gateway Restore	
Restore Disabled	Checkbox for toggling if the Gateway Backup file's contents such as projects, database connections, OPC connections, etc will be disabled upon restore.
Disable Temp Project Backup	Checkbox for toggling if the projects directory will have a temporary backup created during the restore process.
Override Gateway Name	Checkbox for toggling if the name of the Gateway should be overridden before the Gateway is restored. **Caution:** It is not recommended to change the System Name on a Gateway that is in production. Doing so can cause subsystems that reference the name, such as the Tag Historian system, to malfunction.
New Gateway Name	The name of the new Gateway. Requires "Override Gateway Name" to be enabled.

Command-line Utility

In Windows and Linux, you can use the command-line utility to restore a Gateway backup. To run the Command-line Utility, open a shell and enter the command below. Refer to the Command-line Utility page for a complete list of **'gwcmd'** options.

```
gwcmd -s C:\Backups\Ignition\IgnitionBackup.gwbk
```

Related Topics ...

- Gateway Scheduled Backups
- Project Export and Import
- Exporting and Importing Tags

In This Section ...

Gateway Scheduled Backups

You never know when a disaster is about to happen, and scheduled backups can get you out of a bad situation. Ignition will automatically create Gateway backups based on whatever schedule you want, and save them wherever you need, even on another server. Just fill in a few settings and Ignition will do the rest.

Once you have regular scheduled backups setup, make sure to secure them too! If you have your IT department regularly save files to an off-site location, it's easy to add these scheduled Gateway backups to the list.

On this page ...

Setting up Automatic Gateway Backups

Watch the Video

UNIX Crontab

The scheduled backup system works on a schedule that is specified using **UNIX Crontab** syntax. This is a standard format for specifying a basic schedule. The format consists of five space-separated fields, one for minute, hour, day-of-month, month, and day-of-week as shown below. The special Gateway Configuration 72 character ***** means **all**. Slashes can be used to indicate that values should be stepped, for example, */5 in the minutes field means "every 5 minutes", or 0:00, 0:05, 0:10, etc.

Crontab Format and Ranges

- Minutes - (0 - 59)
- Hour - (0 - 23)
- Day of the Month - (1 - 31)
- Month - (1 - 12)
- Day of the Week - (0 - 6) (0 to 6 is Sunday to Saturday

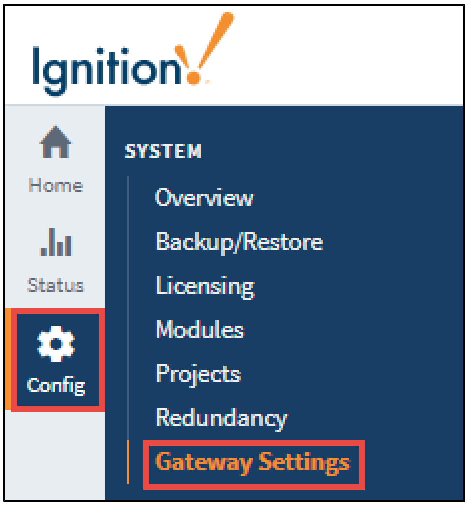
Here are some examples using Crontab formatting:

Examples	
0 1 5 * *	Once a month, on the 5th day at 1am
5 * * * *	Once an hour, on the :05 minute. 0:05, 1:05, 2:05, etc.
30 5 * * 1	Every Monday at 5:30am
* 6-14 * * *	Every minute, but only between 6am and 2pm
*/5 8-17 * * 1-5	Every 5 minutes between 8am and 5pm but only during the week (1-5). 0=Sunday, 1=Monday, etc.
*/15 * * * *	Every 15 minutes, on the quarter-hour. 0:15, 0:30, 0:45; 1:00, 1:15, etc.

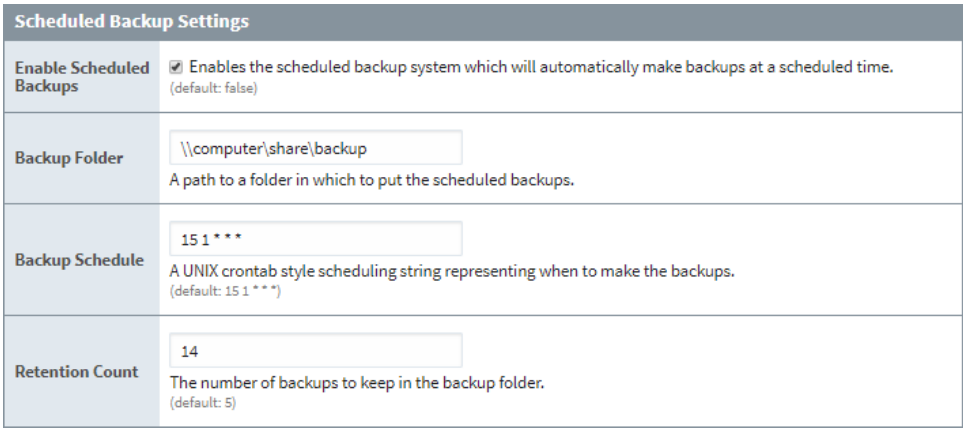
To Setup Scheduled Backups

Gateway backups can be created automatically in Ignition. You can set up a schedule to perform scheduled Gateway backups in the Gateway Webpage. The following is an example of setting up a daily Gateway backup schedule.

1. Go to the **Config** tab of Gateway Webpage, and under System select **Gateway Settings**.



2. On the **Gateway Settings** page, scroll down to the middle of the page to the **Scheduled Backup Settings** section.
   The properties in this section allow you to enable and set the backup schedule.



Scheduled Backup Settings

Enable Scheduled Backups	☑ Enables the scheduled backup system which will automatically make backups at a scheduled time. (default: false)
Backup Folder	\\computer\share\backup A path to a folder in which to put the scheduled backups.
Backup Schedule	15 1 * * * A UNIX crontab style scheduling string representing when to make the backups. (default: 15 1 * * *)
Retention Count	14 The number of backups to keep in the backup folder. (default: 5)

3. Select the checkbox for **Enable Scheduled Backups**.
   This enables the scheduled backups to execute at a scheduled time.

4. In **Backup Folder**, type a path to a folder where the backups will be saved. For example, you can type: `\\computer\share\backup`

   The path to a folder where the backups are to be saved can be a local folder (`C:\backups`) or a network path (`\\computer\share\backup`) as shown in the image above.

5. In **Backup Schedule**, type a UNIX crontab scheduling string to specify when you want to make the backups.
   For example, the code below makes a backup at 15 minutes and 1 hour after midnight, every day, every month, and every day of the week (1:15am every day).

```
15 1 * * *
```

6. In **Retention Count**, specify the number of most recent backups kept in the backup folder. For example, you can specify 14 if you want to only keep the last 14 backups. After 14 backups, the oldest backup will be deleted and the new one will be saved.

7. Click **Save Changes** at the bottom of the page.
   Now, the Gateway will automatically make backups for you based on the schedule of your settings.

Scheduled Backup Errors

If something is wrong with the scheduled backup system, Ignition will store error messages to the Gateway logs.

Related Topics ...

- Project Export and Import
- Gateway Backup and Restore

Project Export and Import

Project backup and restoring from a project backup is referred to as Project Export and Import. Projects are exported individually, and only include project-specific elements shown in the list below. They **do not** include Gateway resources, like database connections, Tag Providers, and Tags. The exported file (`.zip`) is used to restore / import a project.

The resources listed below are included in a project export.

- Alarm Pipelines
- Named Queries
- Perspective Properties
- Perspective Views
- Project Properties
- Reports
- Sequential Function Charts
- Transaction Groups
- Vision Client Tags
- Vision Windows
- Vision Templates
- Client Event Scripts
- Gateway Event Scripts

There are two primary ways to export and import a project:

- **Gateway Webpage** - exports and imports the entire project.
- **Designer** - exports and imports only those resources that are selected.

This page describes how to create a project export and import a project from an exported file.

On this page ...

INDUCTIVE UNIVERSITY

Making Project Backups

Watch the Video

Project Export

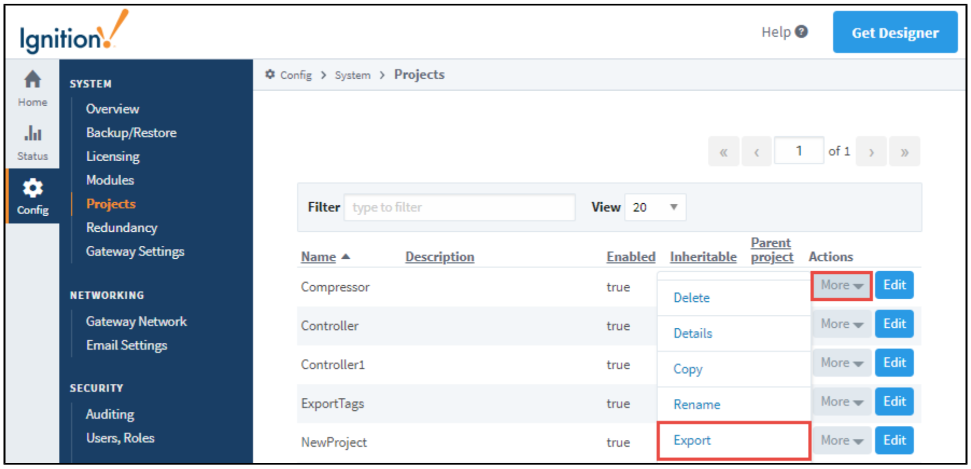
A Project Export is a little smaller and takes even less time to run than a Gateway backup! Once you have an exported file of your project, you can take it to any other Gateway and merge it in with the other projects. This makes it simple to keep a development server and push your projects after you complete them. The Enterprise Administration Module can even do this for you. You can perform a project export from two locations, the Gateway Webpage and the Designer. This secton describes how to create a project export from the Gateway Webpage and the Designer.

Export a Project from the Gateway Webpage

Making an export from the Gateway Webpage, exports the entire project to a (`.zip` file). The exported file only includes resources from the project. Notably missing from the project export are any Gateway resources, even if they are presented in the Designer such as Ignition Tags. They need to be exported separately. Refer to Exporting and Importing Tags.

> ⓘ **Before Exporting a Project**
>
> Remember that Tags and Gateway level configurations (such as device connections, database connections, tags, etc) are not included in a Project Backup. Those resources are only exported in a gateway backup.

1. Go to the **Config** tab of the Gateway Webpage, and click on **System > Projects.**
2. The **Projects** screen will be displayed and you can see your existing projects.
   To the right of project name, click the **More** button and select **Export**. This exports your project as a `.zip` file. The exported file will have the project name, date, and a 4 digit unique number followed by the file extension (i.e., Compressor_2019-03-21_1123.zip).



Name	Description	Enabled	Inheritable	Parent project	Actions
Compressor		true			More Edit
Controller		true			More Edit
Controller1		true			More Edit
ExportTags		true			More Edit
NewProject		true			More Edit

3. By default, the `.zip` file is saved in your web browser's default **Downloads** folder. This export of the project can be imported to any other instance of Ignition that is running the same version or later.

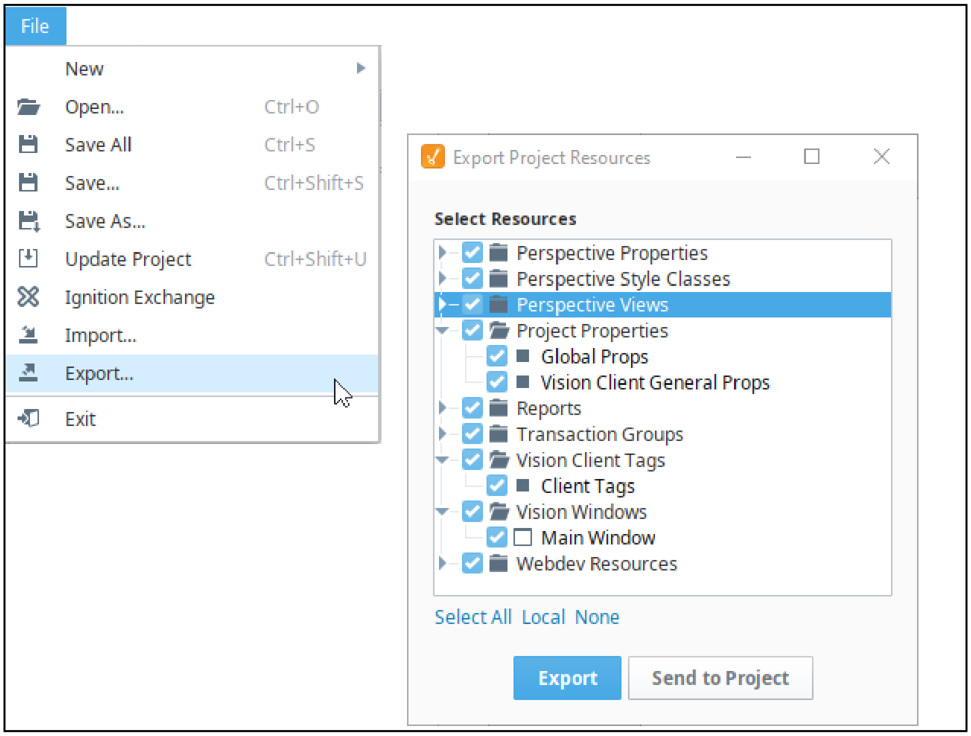
Export a Project from the Designer

When making a project export from the Designer, you get to choose which project resources are added to the export file. The export does not include any Gateway resources. When restoring a project export in the Designer, you also get to choose which project resources are restored into the currently open project from the exported `.zip` file.

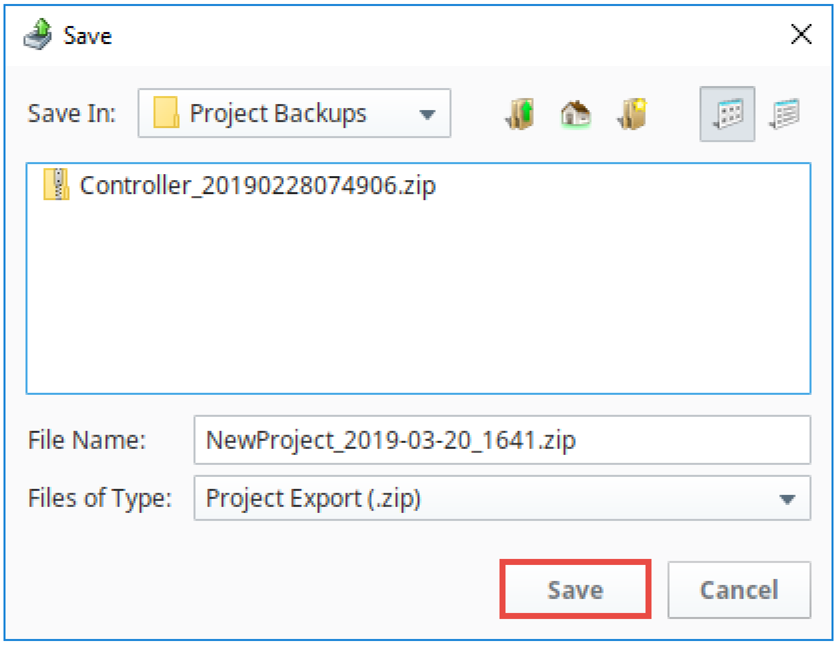
1. From the Designer, go to the top menubar, click on the **File** menu and select **Export**.
2. The Export screen will be displayed and you'll see a list of project resources to export. By default, local project resources are selected. Unselect any resources you don't want to include in the export.

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

When exporting a project, there is a **Send to Project** button that allows you to directly send the project (or resources in a project) to another project on the same gateway. In addition, a **'local'** option was added to the project export selection window. By selecting this option, only local (non-inherited) project resources are selected and will be exported.



3. Click **Export**. This opens a Save dialog window. Select a folder and click **Save**.



4. This export of the project can be imported to any other instance of Ignition that is running the same version or later.

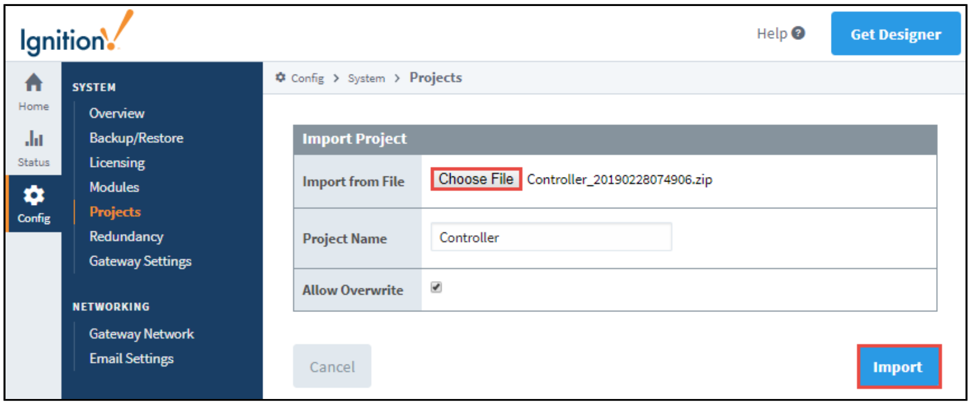
Project Import

When you restore / import a project from an exported file in the Gateway Webpage, it will be merged into your existing Gateway. If there is a naming collision, you have the option of renaming the project or overwriting the project. Project exports can also be restored / imported in the Designer. This will even allow you to select which parts of the project import you want to include and will merge them into the currently open project.

Import a Project from the Gateway Webpage

When restoring a project from the Gateway Webpage, Ignition imports the entire project from an exported file.

1. Go to the **Config** tab of the Gateway Webpage, and click on **System > Projects.**
2. The **Projects** screen will be displayed and you can see your existing projects. At the bottom of the screen, click the **Import project...** link.
3. A second Projects screen will open. Click on **Choose File**, find your exported `.zip` file from your browser, and then click **Open**.
4. Enter the **Project Name**. If there is already a project with the same name, the **Import button will be grayed out** preventing you from importing the file. The system gives you the option to **Rename** the project you want to import, or **Overwrite** the existing project.
5. Click **Import**.



Import from File: Controller_20190228074906.zip

Project Name: Controller

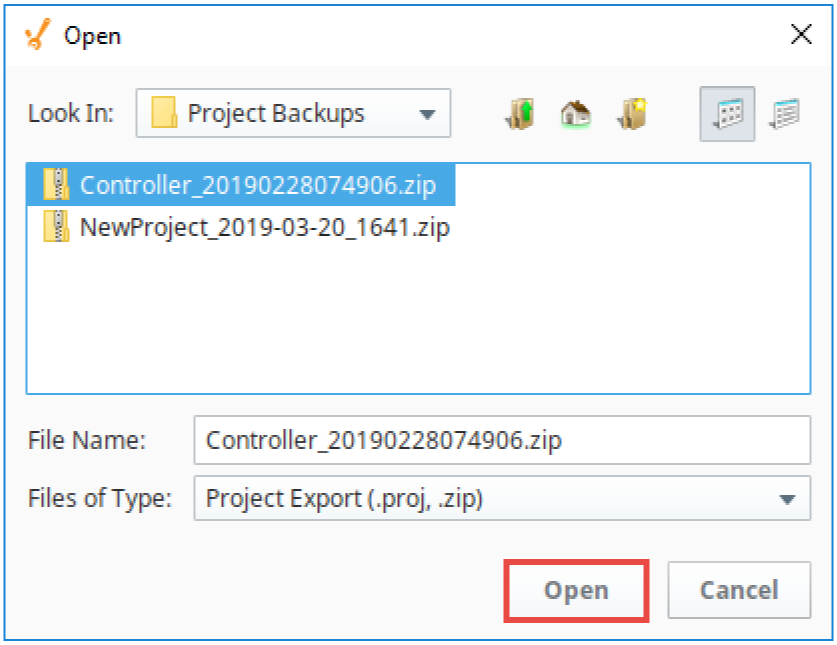
Allow Overwrite ☑

6. If you are importing a new project, you will see your new project added to the list along with the other projects.

Import a Project from the Designer

When restoring a project from the Designer, you can select the entire project or parts of the project, and merge them into the one you are currently working on. Ignition will let you choose which resources to import.

1. In Designer, select **File > Import**.
2. The dialog screen will open, select the project export file (`.zip`) from the default folder, and click **Open**.



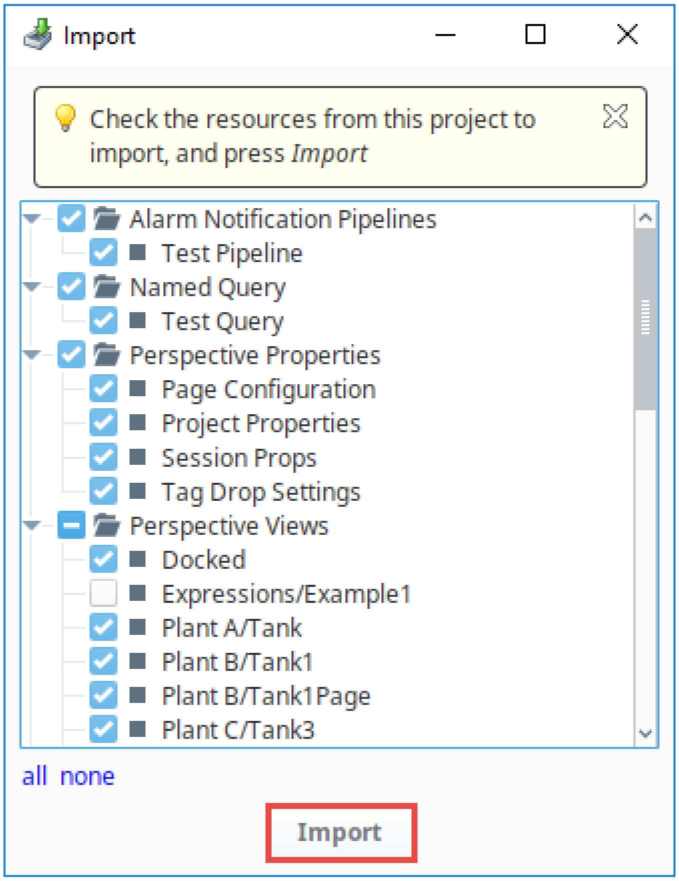
Look In: Project Backups

Controller_20190228074906.zip
NewProject_2019-03-20_1641.zip

File Name: Controller_20190228074906.zip

Files of Type: Project Export (.proj, .zip)

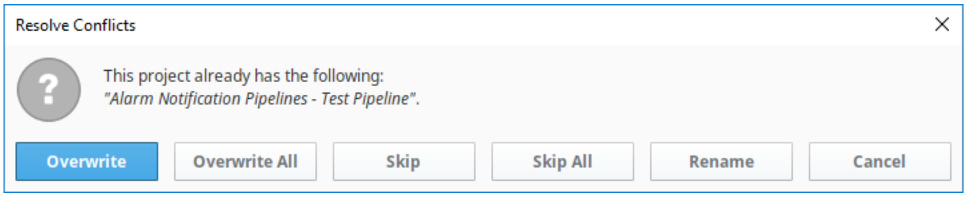
3. By default, all resources are selected. Unselect the resources from the project that you do not want to import. Click the **Import** button .



Check the resources from this project to import, and press *Import*

- ☑ Alarm Notification Pipelines
  - ☑ Test Pipeline
- ☑ Named Query
  - ☑ Test Query
- ☑ Perspective Properties
  - ☑ Page Configuration
  - ☑ Project Properties
  - ☑ Session Props
  - ☑ Tag Drop Settings
- ⊟ Perspective Views
  - ☑ Docked
  - ☐ Expressions/Example1
  - ☑ Plant A/Tank
  - ☑ Plant B/Tank1
  - ☑ Plant B/Tank1Page
  - ☑ Plant C/Tank3

4. If there are any conflicts when importing your project resources, a dialog box will appear and ask you to either **Overwrite**, **Overwrite All**, **Skip**, **Skip All**, **Rename** the affected items**,** or **Cancel**. Once all conflicts are resolved, **Save** your project.



This project already has the following:
"Alarm Notification Pipelines - Test Pipeline".

Related Topics ...

- Gateway Backups and Restore
- Gateway Scheduled Backups
- Exporting and Importing Tags
- Project Inheritance

Ignition Exchange

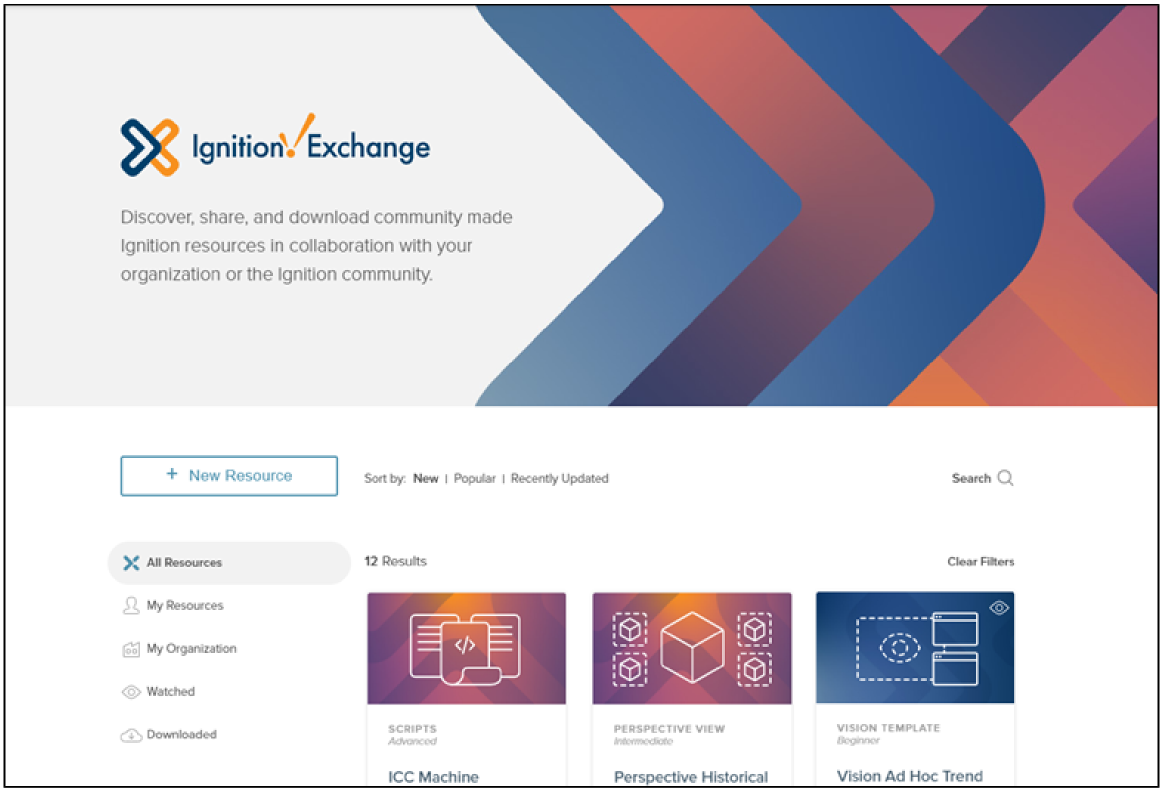
The following feature is new in Ignition version **8.0.5**
Click here to check out the other new features

On this page ...

In the Ignition Exchange you can access resources, templates, and tools that you can use in your own Ignition projects. The Ignition Exchange offers a wide variety of ways to store and access these resources. Ignition assets designed by others in various industries can be shared through the Exchange. This collection encompasses anything that can be built inside of Ignition including screens, graphics, templates, views, reports, alarm pipelines, scripting functions, database backups, projects, full systems, and Ignition demos.

As an individual, you can upload, browse, or download your own tools. If you'd like to keep these tools private, you can store them for your own use, or you can make the resource public, so anyone can access and use it.

Organizations or teams can use the Exchange to store and access private resources they want to make available to a select user group. This gives them the opportunity to collaborate and access company templates, and use them across corporate projects.

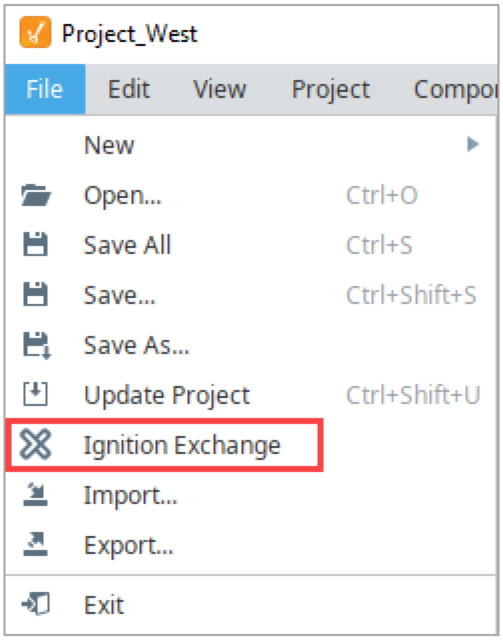


Discover, share, and download community made Ignition resources in collaboration with your organization or the Ignition community.

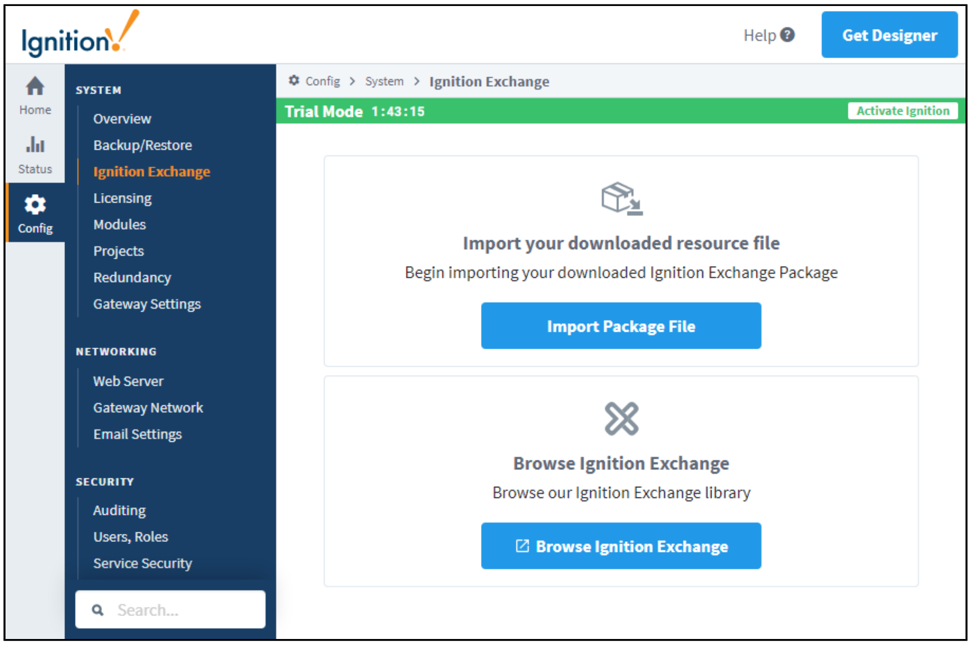
Access the Exchange

You can access the Ignition Exchange, in several ways:

- Go to https://inductiveautomation.com/exchange/.
- Within the Designer, select **File > Ignition Exchange**.



- Within the Gateway webpage, click on **Config > Ignition Exchange**. Then click **Browse Ignition Exchange**.



Import your downloaded resource file

Begin importing your downloaded Ignition Exchange Package

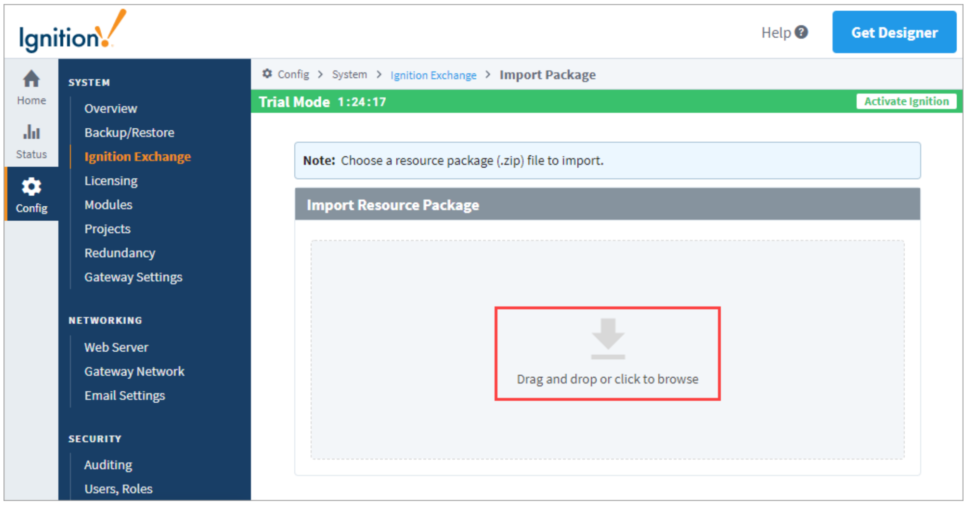
Browse Ignition Exchange

Browse our Ignition Exchange library

Import an Ignition Exchange Package from the Gateway Webpage

When you restore / import a project from an exported file in the Gateway Webpage, it will be merged into your existing Gateway. To import a downloaded resource file,

1. Go to the Gateway webpage, and click **Config > Ignition Exchange**.
2. Click on **Import Package File**.
3. Drag and drop your resource package file, or click the link to browse for the file.



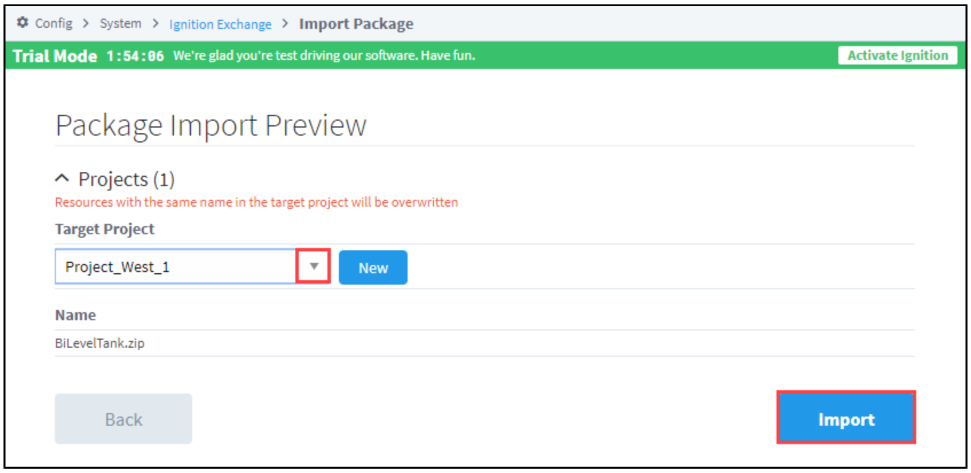
Note: Choose a resource package (.zip) file to import.

4. Navigate to the file you want to import and click **Open**.
5. The Gateway will display a confirmation message if the package is valid, and it will display some Readme instructions for the file. Click **Next**.
6. The Package Import Preview screen is displayed. Use the down arrow in the Target Project box to choose a project. Then click **Import**.

If the new project has a name conflict with an existing project, you can provide a new name for your project.

Important

Resources with the same name in the target project will be overwritten.



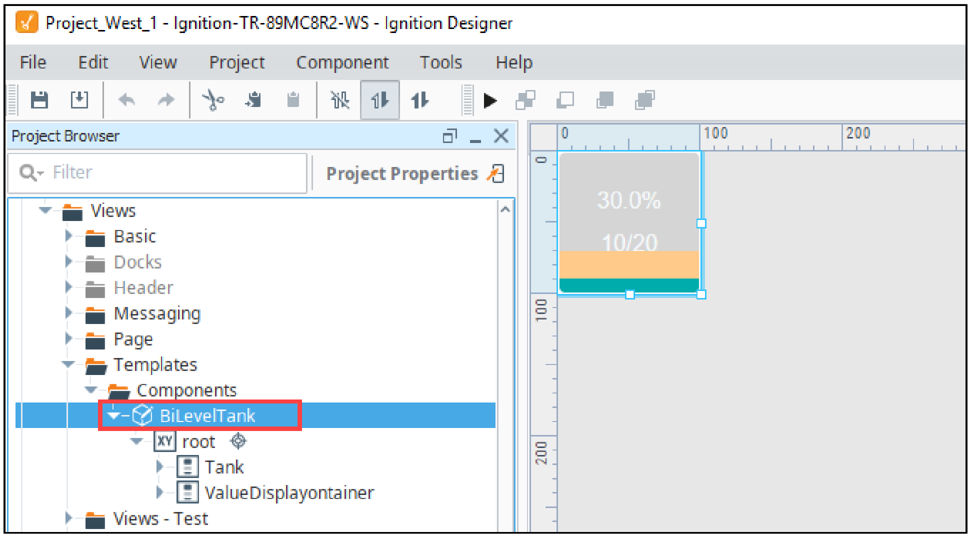
We're glad you're test driving our software. Have fun.

Resources with the same name in the target project will be overwritten

Target Project: Project_West_1

Name: BiLevelTank.zip

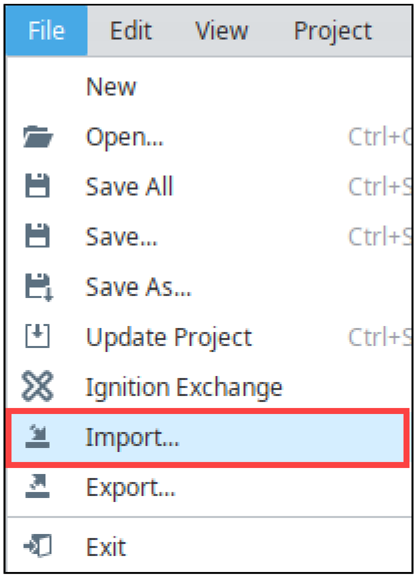
7. A confirmation message is displayed if the package was successfully imported. You can now go to your project in the Designer and see the new resource.



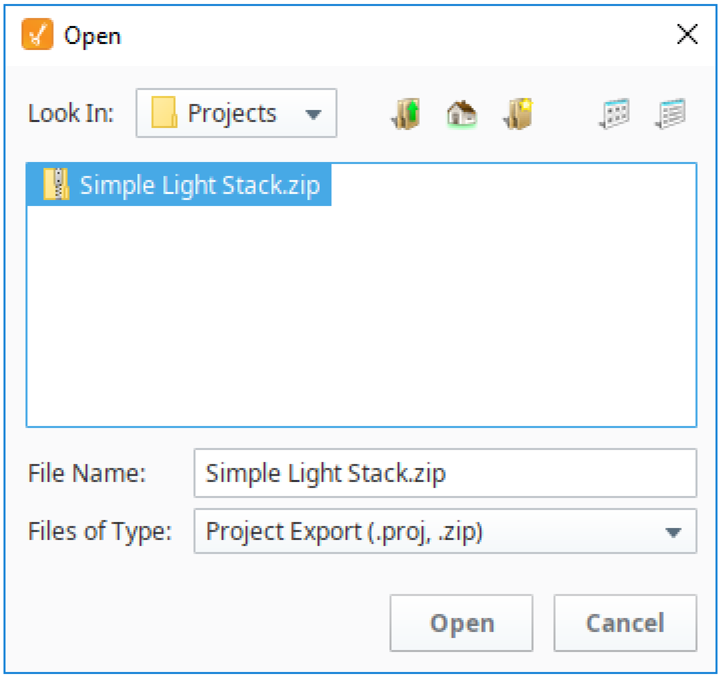
Import an Ignition Exchange Project Package from the Designer

In this example, we're going to import the Simple Light Stack project that we already downloaded from Ignition Exchange. Note that we also extracted the initial `.zip` file that was downloaded. This project contains a Perspective view with a light stack graphic. To import this view into a project that is open in the Designer, do the following.

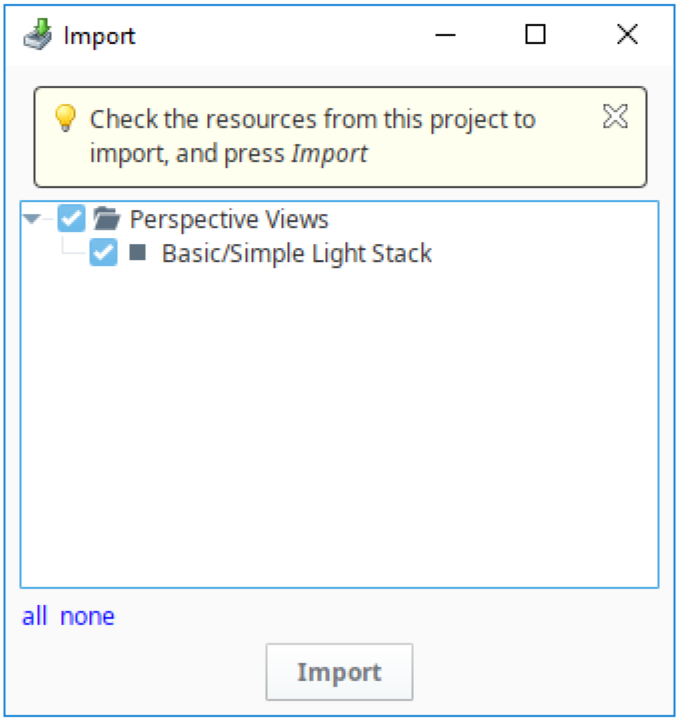
1. In the Designer, click **File > Import.**



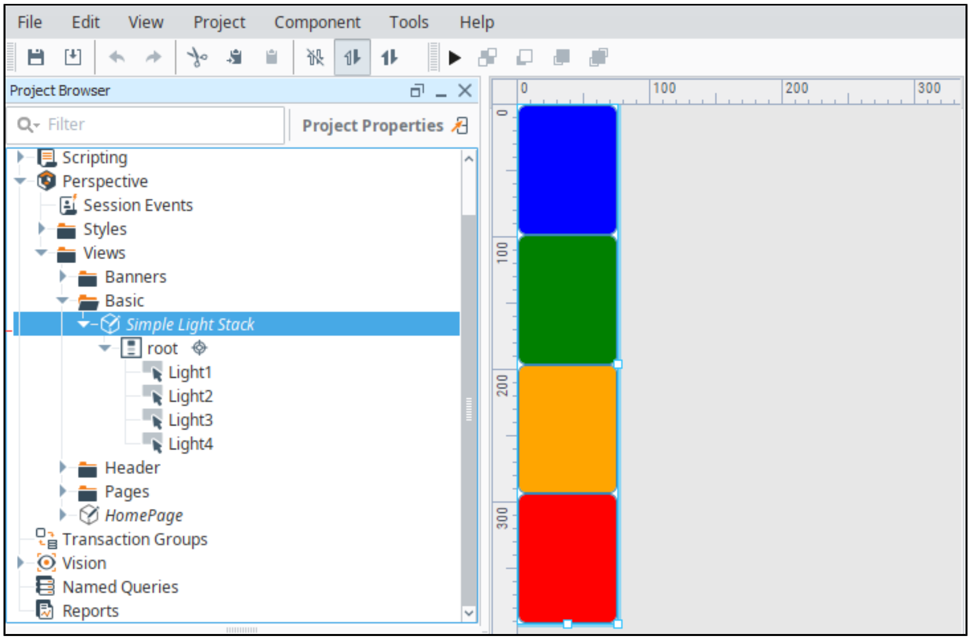
2. Navigate to the project file (`.proj` or `.zip`) you want to import, then click **Open**.



3. In the Import popup window, select the resources that you want to import from the project. Click **Import.**


Check the resources from this project to import, and press *Import*

4. The view is now part of the current project.



Restore a Gateway Backup from Ignition Exchange

Ignition Exchange can also have Gateway backup files as resources. After you've downloaded the file from Ignition Exchange, follow the steps for a Gateway Restore in Gateway Backup and Restore.

When you perform a Gateway Restore, **ALL** of the server's current configuration will be permanently lost! Restoring a Gateway backup overwrites all of the existing settings including your projects. There is no merge option for a Gateway backup. We recommend you always make a backup of the existing server immediately before performing a Gateway Restore.

Gateway Command-line Utility

The Gateway Command-line Utility provides a list of commands you can use to perform specific functions in the Gateway. The Gateway Command-line Utility or **'gwcmd'** provides basic commands, such as resetting the main password, changing the Gateway's port, or restarting the Gateway.

⚠ **A note to our legacy users...**

Older versions of Ignition featured a visual Gateway Control Utility or GCU that could start and stop the Ignition service. This visual element of the GCU, as well as the ability to start and stop the service have since been removed in Ignition 8.0. For more information on starting or stopping the service, please see Use the Command-line Utility to Start or Stop the Gateway below.

More information on the older version of the GCU can be found in Deprecated Features of the user manual.

Command-line Utility Options

The Gateway command-line utility supports Windows, Linux, and Mac OS platforms. The **'gwcmd'** only runs on the same machine as the Ignition Gateway and requires administrative privileges.

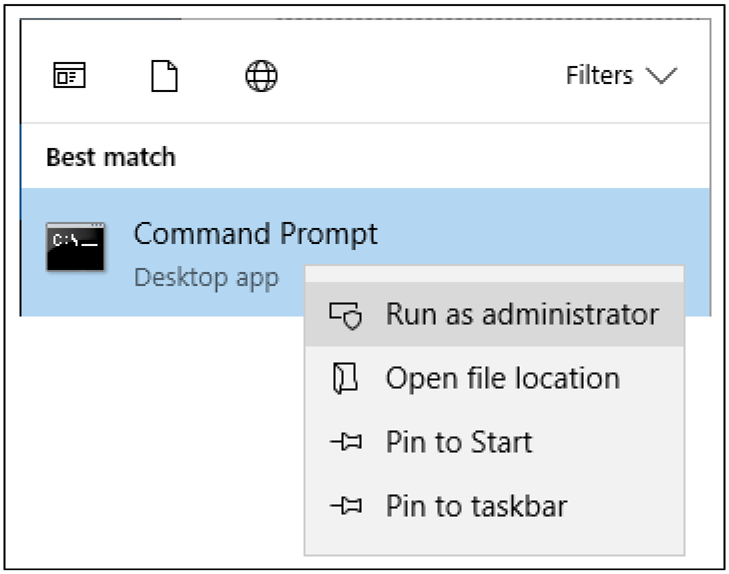
INDUCTIVE UNIVERSITY

Gateway Command-line Utility

Watch the Video

To run the Gateway Command-line Utility, open a command shell and type: **gwcmd <option>**. This example uses the Windows platform:

1. Open the **Command Prompt** with admin privileges. In the search bar, enter **cmd** then right click next to the **Command Prompt** to select **Run as administrator**.

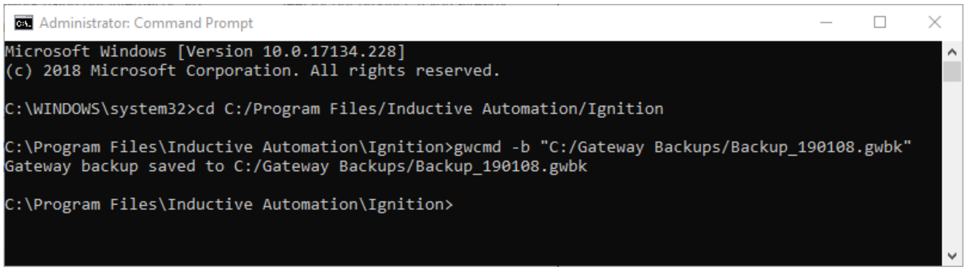


2. Any time you run any of the Gateway command utility options, you need to run them from the directory that Ignition is installed in. The default install directory is: **C:/Program Files/Inductive Automation/Ignition**.

3. From here, you can enter any of the command options listed in the table below. This example is of a Gateway backup using the following command: **gwcmd -b "C:/Gateway Backups/Backup_190108.gwbk"**.

   > **Using spaces in a file path**
   >
   > When using spaces in a file path name, use quotation marks around the full path name, as shown in the example above. Forward or backward slashes can be used to separate folders. The file path includes the disk name, folder path, and file name.

4. The above command created a Gateway Backup on the C drive, in the Gateway Backups folder with a file name of Backup_190108.gwbk. The image below shows all the commands used in Steps 2 and 3.



```
Microsoft Windows [Version 10.0.17134.228]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>cd C:/Program Files/Inductive Automation/Ignition

C:\Program Files\Inductive Automation\Ignition>gwcmd -b "C:/Gateway Backups/Backup_190108.gwbk"
Gateway backup saved to C:/Gateway Backups/Backup_190108.gwbk

C:\Program Files\Inductive Automation\Ignition>
```

Use the Command-line Utility to Start or Stop the Gateway

One of the common uses for the Command-Line Utility is to start or stop the Gateway.

Windows

Ignition's installation directory contains `start-ignition.bat` and `stop-ignition.bat`, which can start or stop the service. Example:

```
C:\Program Files\Inductive Automation\Ignition> start-ignition.bat
```

However, you can also use Windows native service commands to control the running state of the Gateway:

```
net start ignition
```

```
net stop ignition
```

Linux

You can control the service using the `ignition.sh` script. It can be called with the `start` and `stop` parameters to perform the relevant operations.

For example:

```
/usr/local/bin/ignition/ignition.sh start
```

Additionally, you can use native terminal commands to start or stop the service:

```
service ignition start
```

```
service ignition stop
```

> This feature was changed in Ignition version **8.0.14**:
> The "ignition" command is now "Ignition-Gateway".
>
> ```
> service Ignition-Gateway start
> ```
>
> ```
> service Ignition-Gateway stop
> ```

Mac OS X

You can access the service from the install directory using the "ignition.sh" script. On a typical Mac install using the dmg installer, the full command (without a custom location specified) is the following:

```
/usr/local/ignition/ignition.sh start
```

Command-line Utility 'gwcmd' Options

The following table lists all available **'gwcmd'** options.

Options	Description
`-a,--activate <license-key> Offline activation`	Creates an `activation_request.txt` file that can be used to request a `license.ipl` file from the Inductive Automation website. You must specify the license key to use for activation. The `activation_request.txt` file is saved in the current directory.
`-b,--backup <new filepath>`	Downloads a Gateway backup `.gwbk` file and saves the file to the specified path. The path can be either an absolute path or a relative path. You will be prompted whether it is OK to overwrite the file if another `.gwbk` file with the same name already exists. You can override with the `-y` option to force the file to always be overwritten.
`-c,--clearks`	The following feature is new in Ignition version **8.0.3** Click here to check out the other new features Clears the gateway's SSL / TLS setup. The gateway's SSL / TLS connector will be immediately shut down.
`-d, --disabled`	Use with the `--restore` flag to disable all items after gateway restoration and restart.
`-e, --exportks <new filepath>`	The following feature is new in Ignition version **8.0.3** Click here to check out the other new features Exports the gateway's SSL KeyStore in PKCS12 format and saves to the specified path.
`-f, --exportpk<new filepath>`	The following feature is new in Ignition version **8.0.3** Click here to check out the other new features Exports the private key from the gateway's SSL KeyStore in PEM format and saves to the specified path.
`-g,--reloadks`	The following feature is new in Ignition version **8.0.3** Click here to check out the other new features Reloads the Gateway's SSL KeyStore from disk. Any update to the KeyStore will be automatically applied to any new connections.
`-h,--help`	Shows the usage for this command.
`-i,--info`	Retrieves server status and port information from the Gateway if it is running.
`-k,--port <new port>`	Changes the Gateway http port.
`-l,--sslport <new port>`	Changes the Gateway https port.
`-n, --nocrypt`	The following feature is new in Ignition version **8.0.3** Click here to check out the other new features

	Add to the export private key command to not encrypt the private key.
`-o,--name`	The following feature is new in Ignition version **8.0.3** Click here to check out the other new features Specifies a Gateway name while restoring a backup. Additionally, the `-y` command now skips prompts asking for a Gateway name override.
`-p,--passwd`	Enables a password change for the initial user.
`-r,--restart`	Restarts the Gateway.
`-s,--restore`	Restores a Gateway backup (.gwbk) file.
`-t,--tdump`	Performs a thread dump in the Gateway and prints the dump to the command-line.
`-u,--unactivate Offline unactivation`	Creates an `unactivation_message.txt` file that you can use to unactivate a license via the Inductive Automation website. The `unactivation_message.txt file` is saved in the current directory.
`-w,--uselicense <license.ipl path>`	Applies a `license.ipl` file that was downloaded from the Inductive Automation website. You must supply the location of the `license.ipl` file. If it is in the current directory, use `license.ipl` for the location.
`-y,--promptyes`	Automatically answers `yes` to any prompt that may appear in the above commands, such as permission to overwrite an existing file.

Reset the Password

If something happened and you can't get into the Gateway Webpage anymore, you may need to reset the master username/password. You can use this Gateway Command-line Utility to reset the main password for the user with full privileges. This 'gwcmd' utility triggers system commissioning, and prompts the user to enter a username and password for the main user (with full privileges) after the Gateway restarts.

Projects

What Is a Project?

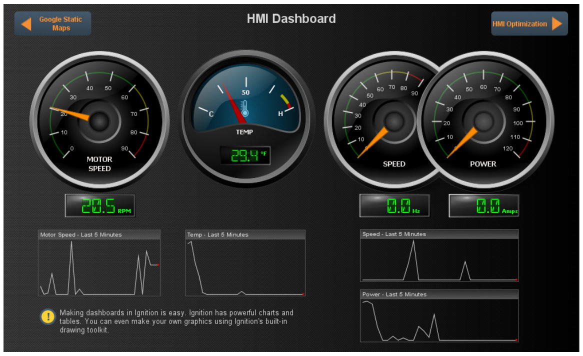
Ignition has two main parts, the Gateway and your projects. The Gateway holds all the shared information like database connections, device connections, and Tags. Projects hold all the designed elements that do the real work. Your projects can hold both interactive elements (like controls, charts, reports, and entry forms) and persistent elements (like historical loggers and automated reports).

Projects are predominantly used to create the screens that your users can interact with, the visualization part of Ignition. Here you can create any system you want, from copying existing HMI /SCADA applications to whole new systems with anything you could want to do. The windows in your project can be used for history charts, reports, database forms, alarms, drawing components, scripting, and templates.

In Ignition, a project is a unit of configuration that contains:

- **Windows, views, and components**: The HMI and SCADA controls to interact with Tags and databases
- **Transaction Groups**: A bi-directional link between databases and PLCs
- **Templates**: A collection of components that can be re-used and quickly updated
- **Reports**: PDF reports for displaying and recording data
- **Scripts**: Timer and event based scripts used throughout the system
- **General settings and properties**: The settings that control access, resource connections, layout, and timing

You use the Designer to configure and create projects. The projects are then viewed in the runtime (Vision Clients or Perspective Sessions). You can create as many projects as you want, and users can easily jump between projects on the fly or open multiple projects at the same time.



Making dashboards in Ignition is easy. Ignition has powerful charts and tables. You can even make your own graphics using Ignition's built-in drawing toolkit.

On this page

...

Visualization Systems - Vision and Perspective

When starting a new project, there are many things to consider, such as who the users will be, how much data you need, what kind of time and resources you have to work with, visualization needs, and so forth. Deciding whether to use Vision or Perspective really comes down to which module best fits your project at the visualization level.

When to Use Vision for a Project

Vision is the best choice for traditional industrial plant-floor and desktop screens , standalone HMIs, and the like. If you need a full, production-ready application right now, then it is best to choose Vision. If you're using multi-monitor or multi-desktop workstations, dedicated control panels and applications, desktop-dedicated or dedicated-access applications, terminals, or parallel screens, then Vision is the best way to go.

When to Use Perspective for a Project

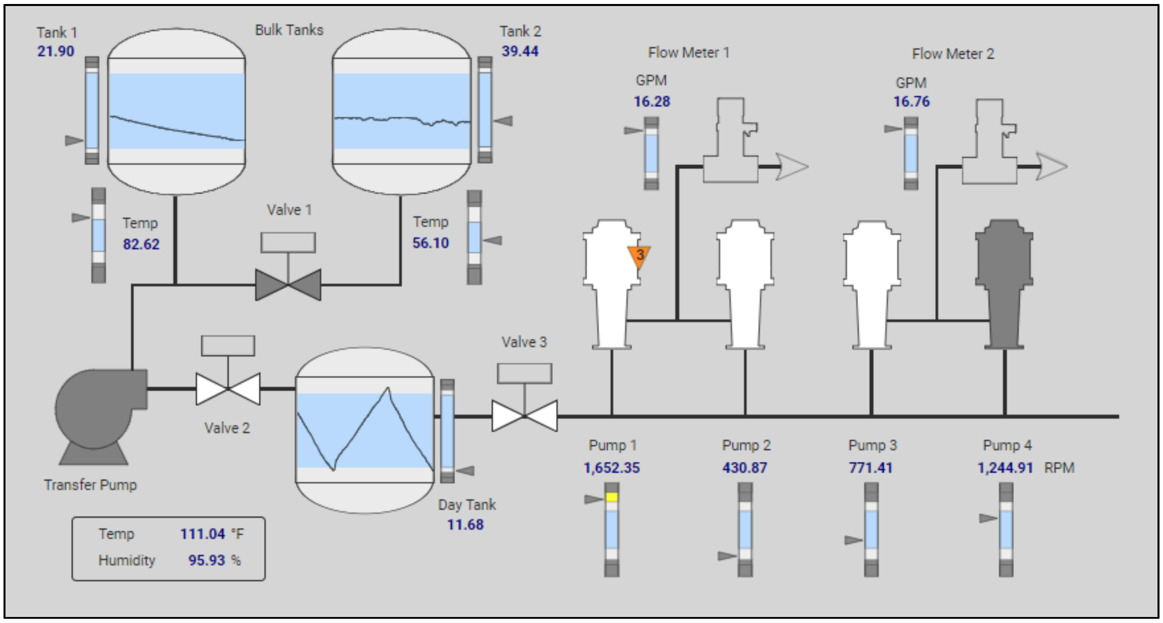
If you need to build mobile-responsive applications, then Perspective is the recommended way to go. If your application needs to run on a mobile OS , namely iOS or Android, then choose Perspective. If you need Two-Factor Authentication or federated identities for the application you're building, choose Perspective.

With Perspective, you can automatically adapt to fit any screen size using mobile-optimized container types. It provides the ability to use your device's sensors and intuitive touch commands, as well as message handling, flexible property bindings, and CSS3 styles,

Designing a Project

When you launch the Designer, you're prompted to select or create a project. The Designer then launches the project and you can set and modify the different types of project settings and resources. In the Designer, you can create any number of projects using either Vision or Perspective. Projects that contain viewable elements, such as Vision Windows and Perspective Pages, will have a launch link on the Gateway homepage. Non-viewable elements such as Transaction Groups, Named Queries, SFCs and Reports exist in a project and execute in the Gateway. These resources do not have a runtime, and run independently of any Vision Clients or Perspective Sessions being open.

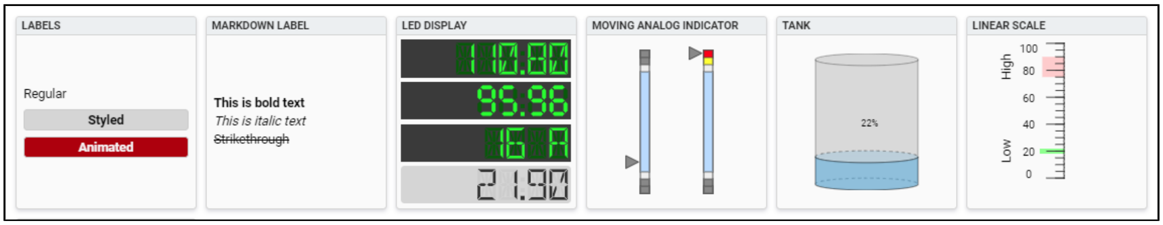
There are no limits to the number of projects that can be created on a Gateway, but each runtime Client, Session, or Designer can operate only on one project at a time.



What Is in a Project?

You use the different tools in Ignition such as components, shapes, images, Symbol Factory graphics, and Scalable Vector Graphics (SVGs) to create the components. Configuring components is the bulk of the designer's work when designing a project. The basic workflow is to take a component from the palette and drop it into a container on a Vision Window or Perspective View. You can use the **Property Editor** panel to alter the component's properties which changes the component's appearance and behavior.

To make the component do something useful, like display dynamic information or control a device register, you configure property bindings for the component. To make the component react to user interaction, you configure event handlers for it. It is primarily through property bindings that you bring windows to life, and have them do useful things. A property binding simply links one component's property to another on the same window.



Switching Between Projects

When you launch an Ignition Client or Perspective Session, it opens a single project to display. If you want to open multiple projects you can have multiple clients open on the same computer, or you can use Ignition's Retargeting system to make seamless transitions between projects. The Retargeting feature allows you to jump from one project to another without closing the client. This allows your users to jump from area to area while still allowing you to keep your designs compartmentalized in multiple projects.

Your projects may all use different authentication sources or role sets for security. This means as your users jump from project to project, they may be able to use some or all of the controls on one project, but only see what is happening on another project.

Project Workflow

Getting up and running quickly with your project is simple:

1. Open the Designer
2. Choose either Perspective or Vision.
3. Start designing!

The only challenge is figuring out exactly what you want to make with your system. If you're new to Ignition, we have a basic workflow with a few common tasks for you. See our Quick Start Guide for simple step-by-step instructions of a sample workflow.

For additional instructions, see Creating a Project.

Project Export and Import

In Ignition, a project backup and restore is referred to as Project Export and Import. Projects are exported individually and only include **project-specific** resources such as Perspective Views, Perspective properties, Vision Windows, Vision Templates, client event scripts, alarm pipelines, named queries to name a few. They **do not** include any Gateway resources like database connections and Tag Providers. A project is exported to a **`.zip`** file, and you can import it to any other Gateway that you have permission to access.

When you import a project from the **`.zip`** file, it will be merged into the existing Gateway. If a project already exists on the Gateway you are attempting to import it on, you have the option of renaming the project or overwriting the existing project.

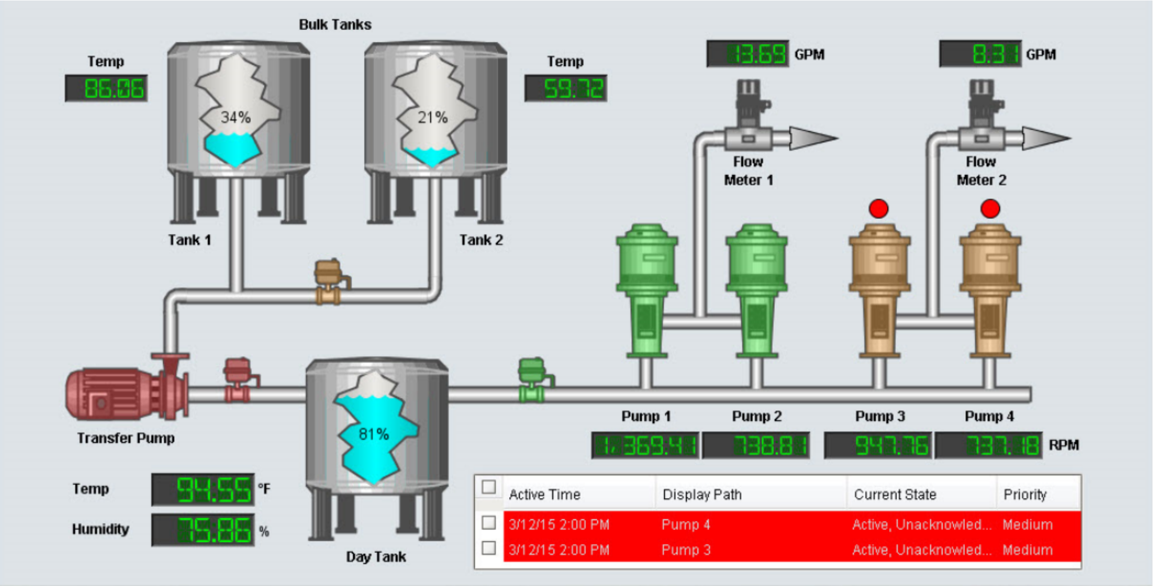
Project Export and Import are covered in detail on the Project Export and Import page.

Project Versioning and History

Project versioning is handled outside of Ignition. The file system that stores data in Ignition stores everything as a series of files. You can use any tool you'd like to save or export theses files outside of Ignition, including using versioning software to keep track of your changes.

Project Examples

Ignition is such a diverse application that it can work in any field you can imagine. Just a few examples: Oil and gas, food and beverage, automotive, medical, air traffic control, water, and waste water. You can get anything you want set up in Ignition. To see a few live example screens, check out our online demo project examples for Vision and Perspective. They are packed with samples of the features and functions you might want to use from reporting, to history, to HMI optimization.



□	Active Time	Display Path	Current State	Priority
□	3/12/15 2:00 PM	Pump 4	Active, Unacknowled...	Medium
□	3/12/15 2:00 PM	Pump 3	Active, Unacknowled...	Medium

What Is not in a Project?

The Designer allows you to create and modify several types of resources that are shared by ALL projects. Depending on the resource, this means that either they run independently of the projects, or they are available for use by any project. Here are a few of them:

- **Alarm Pipelines** - Control alarm notification (runs independently).
- **Sequential Function Charts (SFC)** - Logic to step through a process (runs independently).
- **Transaction Groups** - Perform various actions such as storing data historically, synchronizing database values to OPC, or loading recipe values.
- **Project Library** - Blocks of scripting code (available to all projects).
- **Tags** - Basic or UDT Tags provide realtime data (available to all projects).
- **Alarming** - Alarms exist on Tags, and so are not in a project (available to all projects).

In addition to these resources, the Gateway connections and settings are available to all projects and are set up in the Gateway. These include resources such as database connections and OPC server and device connections.

Related Topics ...

- Designer
- Perspective Sessions
- Property Bindings in Perspective
- Scripting in Perspective
- Vision Client Launcher
- Property Bindings in Vision
- Scripting in Vision
- Alarming
- SQL Bridge (Transaction Groups)

In This Section ...

Project Inheritance

Project Inheritance allows one project to inherit resources from another project. If you have project resources that you want other projects to use such as views, windows, scripts, templates, or pipelines, you can create an inheritable project allowing other projects to inherit those resources. The project inheriting the resources can also overwrite the resources and let you re-define them specifically for that project.

Project inheritance is extremely flexible in that it allows each project to have a parent project that inherits all the resources of that parent project. The inherited project, in turn, can also be configured to be 'inheritable' and become a parent project itself allowing for complex hierarchies of re-usable resources to be designed. Within each project, inherited resources may be used by other, “local” resources. For example, an inherited Vision template could be embedded in a window, or an inherited script could be executed by a button.

This page demonstrates how project inheritance allows you to share resources across multiple projects as you deem fit by pointing one project to another project, and overriding resources to re-redefine resources specifically for the project you're working on.

On this page ...

Configuring Project Inheritance

In order to make a project inheritable, you need to enable project inheritance on the project containing the resources that you want to share.

In this example, we have two projects: one project called 'global' which contains some project resources, and another project that is not inheriting any resources called 'Project_X.' Since we will be sharing the resources from the 'global' project with Project_X, let's first configure the 'global' project to be inheritable.

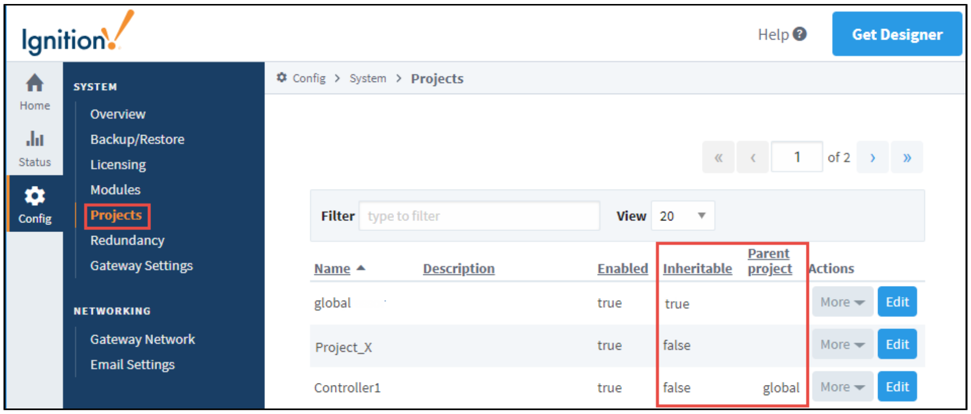
INDUCTIVE UNIVERSITY

Project Inheritance

Watch the Video

Configuring Parent Project

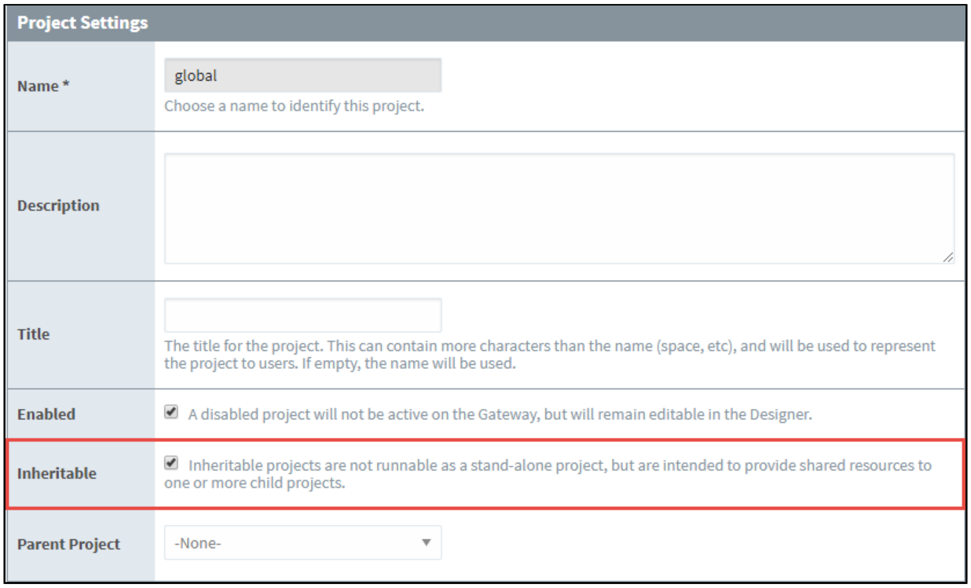
1. Go to the **Config** tab on the Gateway Webpage, and select **Systems > Projects**. This brings up a list of all your projects. You'll notice that for each project listed, you'll see at a glance if a project is Inheritable, and if so, the name of the Parent Project will be displayed.



Name ▲	Description	Enabled	Inheritable	Parent project	Actions
global		true	true		More Edit
Project_X		true	false		More Edit
Controller1		true	false	global	More Edit

2. Find the project you want to make inheritable, and click the **Edit** next to the project name (i.e., global) to open the Project Settings window.
3. Enable the **Inheritable** property, and click **Save**.

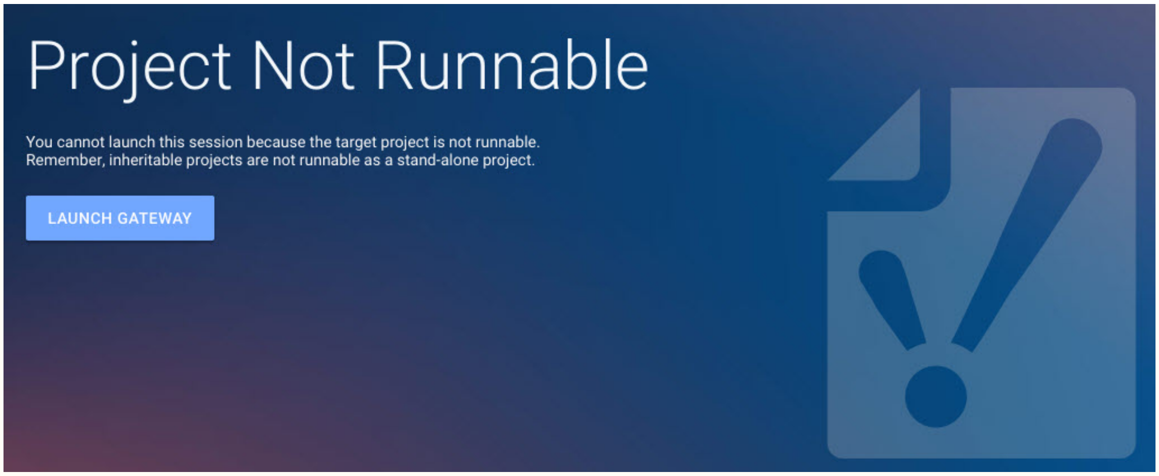
> **Allow Overrides**
>
> When you make a project inheritable, the Allow Overrides function is set by default on all project resources in the inheritable project. This allows all project resources to be propagated to all inherited projects.



Project Settings

Name *	global Choose a name to identify this project.
Description	
Title	The title for the project. This can contain more characters than the name (space, etc), and will be used to represent the project to users. If empty, the name will be used.
Enabled	☑ A disabled project will not be active on the Gateway, but will remain editable in the Designer.
Inheritable	☑ Inheritable projects are not runnable as a stand-alone project, but are intended to provide shared resources to one or more child projects.
Parent Project	-None-

> **Cannot Launch Inheritable Projects**
>
> When you have a project that's flagged as inheritable, you can not launch it as a stand-alone project (i.e., Perspective Session or Vision Client). You will get a 'Project Not Runnable" error message. If you have an inheritable project that you want to launch, you must have another project to inherit from it.



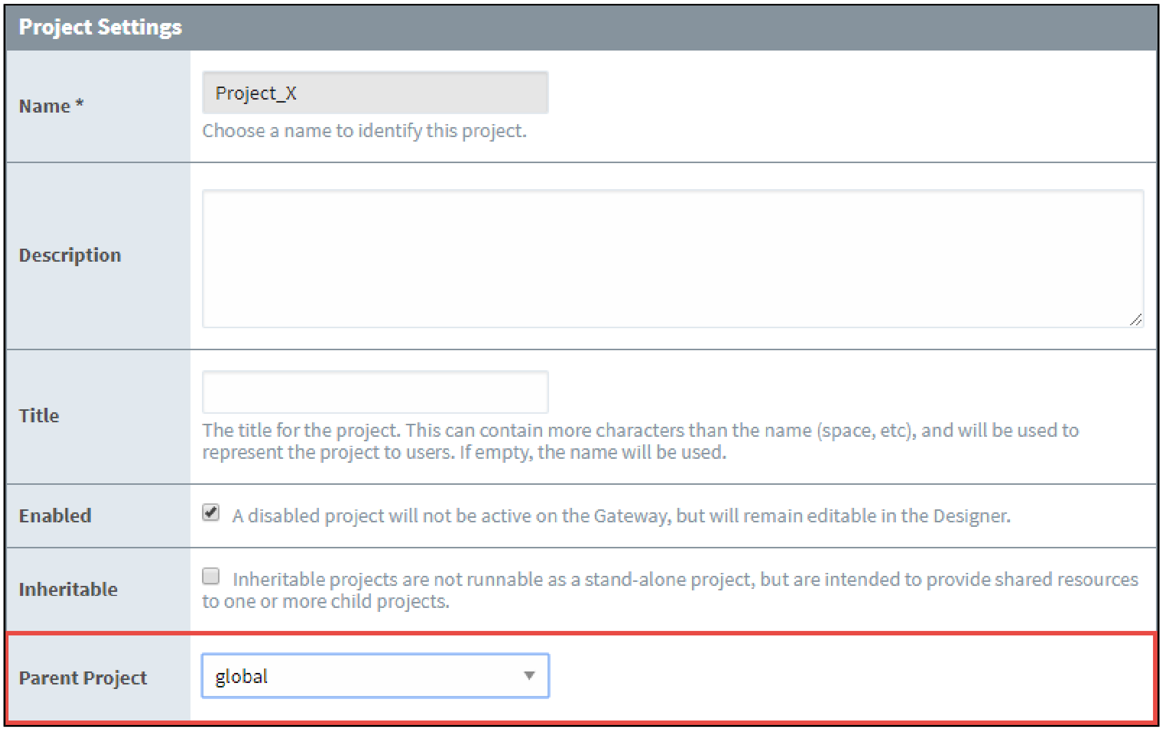
Project Not Runnable

You cannot launch this session because the target project is not runnable.
Remember, inheritable projects are not runnable as a stand-alone project.

Configuring Child Project

Now that we have an inheritable project, let's setup an existing project (i.e., Project_X) so it inherits resources from the 'global' project.

1. Under the **Config** tab select **System > Projects** page
2. Find your project and click the **Edit** button.
3. Select the inheritable project from the **Parent Project** dropdown list (i.e. global), and click **Save**.



Project Settings

Name *	Project_X Choose a name to identify this project.
Description	
Title	The title for the project. This can contain more characters than the name (space, etc), and will be used to represent the project to users. If empty, the name will be used.
Enabled	☑ A disabled project will not be active on the Gateway, but will remain editable in the Designer.
Inheritable	☐ Inheritable projects are not runnable as a stand-alone project, but are intended to provide shared resources to one or more child projects.
Parent Project	global ▾

Using Inherited Resources

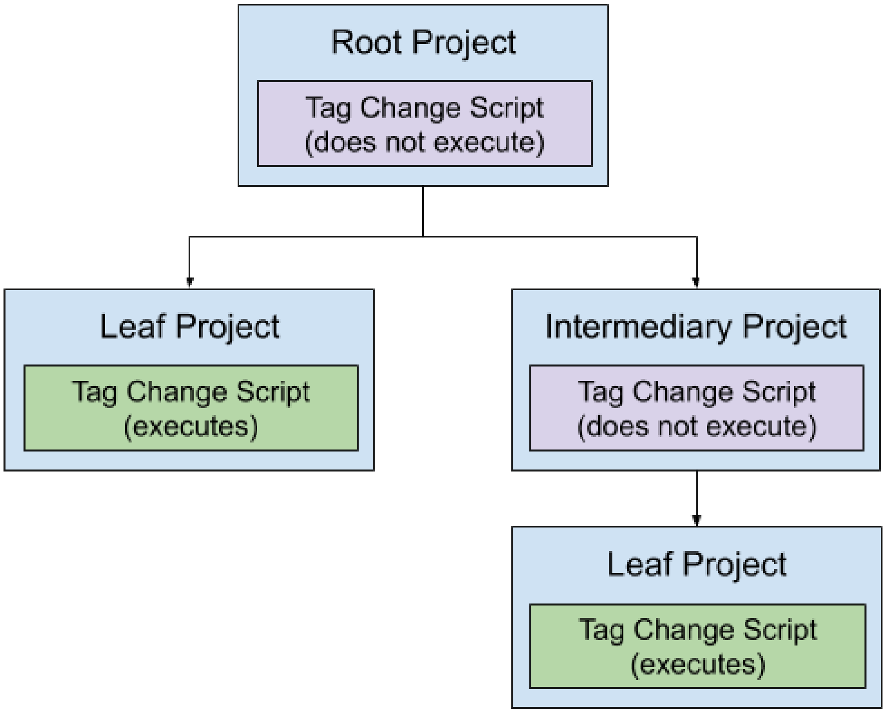
You can treat an inheritable project as a library of resources for use in other projects. When project resources are changed in the original project, these changes will get passed down to the inherited projects. The same thing is true if new resources are added to the inheritable project. By using inheritable projects, you can create a resource library that will help designers build their projects more quickly, ensure consistency and reusability across all their projects.

To view your inherited resources, go to the Designer and open your inheriting project (i.e., Project_X). Expand the folders that contain resources, and you'll notice that the inherited project resources are grayed out, including the Perspective Views and Vision Windows folders. Grayed out resources mean that those resources are inherited, and can only be edited from their original / parent project. Anytime a new resource is added or an existing resource is changed in the inheritable / parent project, it will propagate down to the inherited project and will appear grayed out unless that resource is overridden.

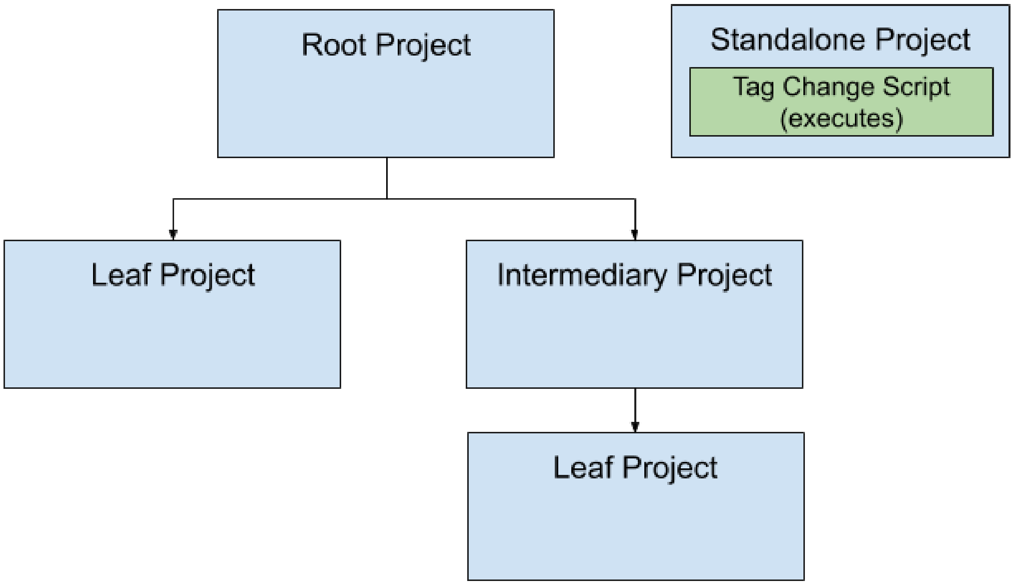
Inheritance and "runnable" Resources

Some resources in a project "run" or execute in every leaf project (that is, a project that is **not** inherited by any other project) in the inheritance chain. Thus, if multiple projects inherit from the same parent project, then each leaf project will contain runnable resources, potentially resulting in duplicate executions.

The diagram below represents an inheritance hierarchy on a single gateway. A "Root" project contains a Gateway Tag Change script. Two projects inherit from the Root, so they'll inherit the script. In both cases, the inheritance chain leads to leaf projects. In this single gateway, two running instances of the same tag change script exist, meaning there will be duplicate script executions.


Root Project
Tag Change Script (does not execute)
Leaf Project
Tag Change Script (executes)
Intermediary Project
Tag Change Script (does not execute)
Leaf Project
Tag Change Script (executes)

In the case of resources that "run", it is highly advised that they exist in a leaf project, or a standalone project (a project that does not participate in inheritance at all). In regards to the diagram above, we could prevent duplicate execution of our Tag Change Script easily by moving the Tag Change Script to a separate, standalone project:



"Runnable" Resources

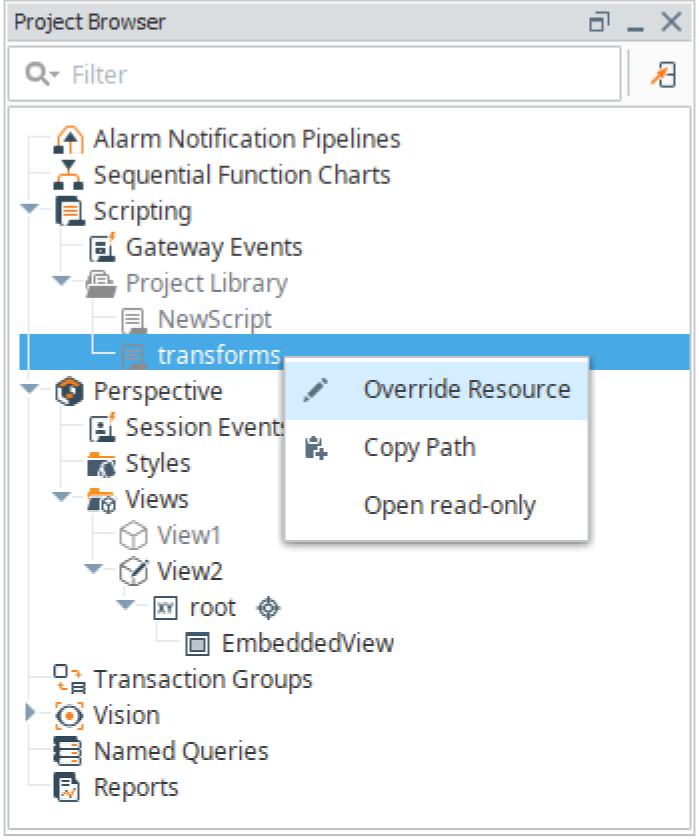
The following resources are considered "runnable":

- Gateway Event Scripts
- Alarm Notification Pipelines
- Sequential Function Charts
- Transaction Groups

Overriding Inherited Resources

To edit a resource in an inherited project, you need to override the resource by right clicking the resource and selecting **Override Resource**. By overriding the inherited resource, the resource is recreated in the inherited project, and any future changes made to the original project resource will *not* propagate down to the inherited resource.

If *new* resources are added in the inheritable / parent project, they will automatically propagate down to inherited projects. New resources that are added in the inherited project will be displayed as grayed out. You will need to override each new resource to edit it.

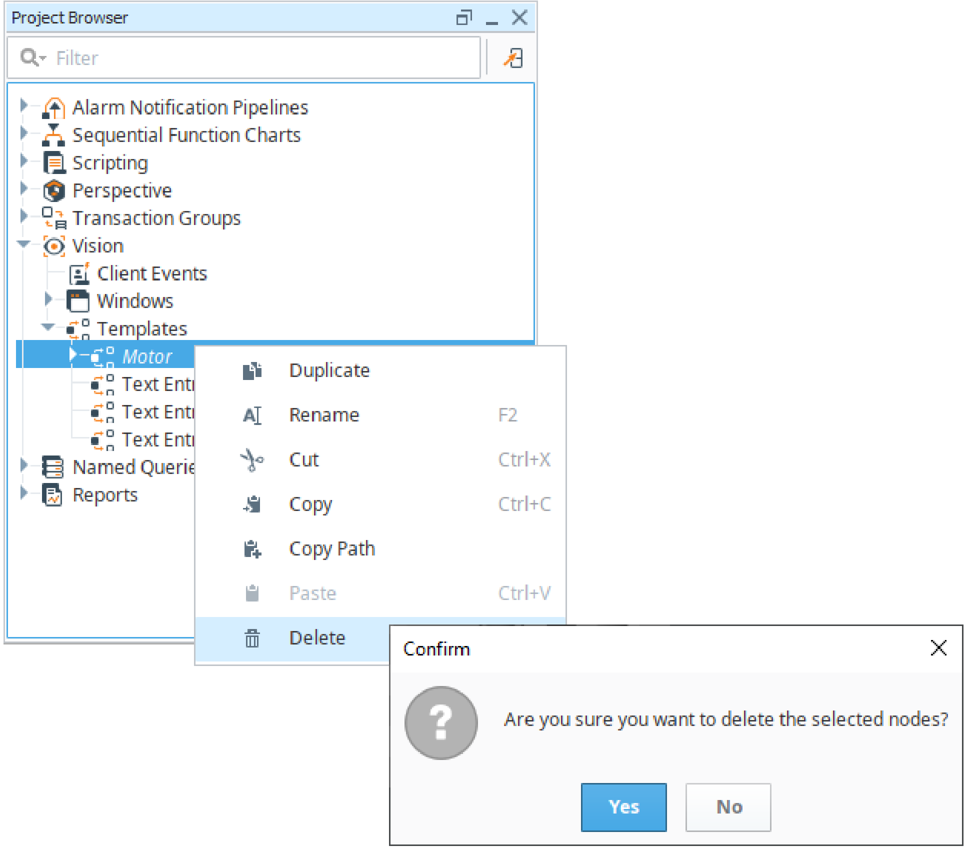


Deleting Inherited Resources

Inherited resources that have been overridden can be deleted from an inherited project. When you delete an inherited resource, it doesn't delete the resource from the inheritable / parent project, it deletes the 'Override' you previously placed on the inherited resource. The inherited project resource will return to the version that is currently in the inheritable / parent project (without your edits), and the resource becomes grayed out in the Project Browser.

1. To delete the override on an inherited resource, right click on the resource and select **Delete**.

2. A dialog box will popup confirming you want to delete the selected node, click **Yes**. The resource will immediately be grayed out preventing designers from editing that resource.



Are you sure you want to delete the selected nodes?

Renaming an Inherited Resource

You can easily rename an inherited resource, but beware that when you rename a resource, the inheritable project will propagate the original resource to the inherited project. For example, say you have an inheritable project resource named 'Map_Transform' and you renamed it to 'CA _Map_Transform.' Ignition knows that the original project resource is no longer there, and because the project is flagged as inheritable, it will propagate that original resource to the inherited project. Now you have both the renamed 'CA_Map_Transform' and the original inheritable resource 'Map_Transform.'

Inheritable Project Examples

There are many ways how you might want to configure and organize your inheritable projects. It's whatever works best for your organization and design projects. Here are a couple of common ways to organize your shareable resources.

- You can create one inheritable project that contains many project different resources: scripts, pipelines, views, templates, windows, SFCs, etc. This is one inheritable project containing all your inheritable project resources.
- Another option is to create several inheritable projects. You can have one inheritable project dedicated for each type of rersource: one for scripts, one for views, one for templates, one for pipelines, etc.

Project Export

When a project is exported, you don't get any inherited resources with your export. However, if a project resource was overridden and edited, that essentially becomes a new resource and will be included in a project export. Resources that are inherited and not overridden will have to be included as part of an inheritable project export. Once a new project is imported into a Gateway, you can always configure that project to inherit from another project.

Related Topics ...

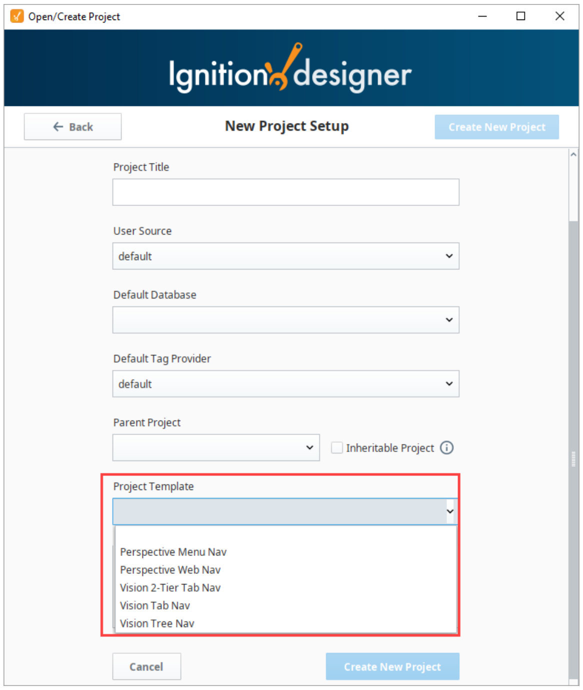
- Project Inheritance - Upgraded Features in Ignition 8
- Project Export and Import

Project Templates

When you launch the Designer, you will be presented with the Open/Create Project window. This window let's you hit the ground running almost immediately by populating a few fields, the most important ones being your project Name and Title. You can use the defaults in the remaining fields and change them in the **Config > Project** page later, but don't overlook the **Project Template** field. Ignition provides several project templates for you to choose from to help you quickly get started developing your project. The templates get you started with a basic navigation structure and let you add more items as your project matures.

On this page ...

> **Note:** You must be connected to the internet in order to see the templates available in the Project Template list.



Open/Create Project

New Project Setup

Project Title:

User Source: default

Default Database:

Default Tag Provider: default

Parent Project:

☐ Inheritable Project

Project Template:

- Perspective Menu Nav
- Perspective Web Nav
- Vision 2-Tier Tab Nav
- Vision Tab Nav
- Vision Tree Nav

Perspective Project Templates

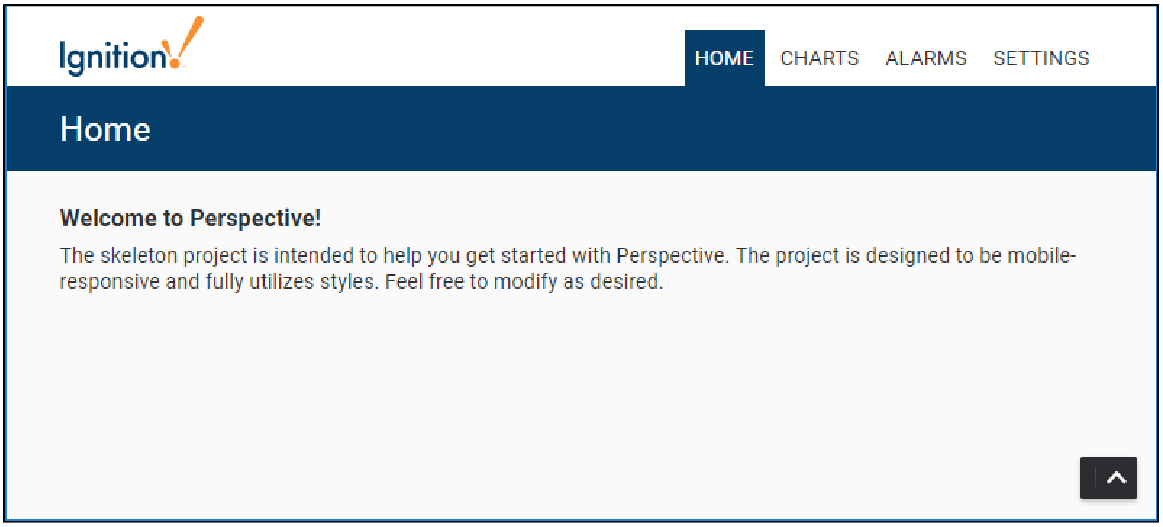
Here are two Perspective Project Templates you can choose from to get you off to a quick start: **Menu Nav** and **Web Nav**.

The Perspective Menu Nav and Web Nav templates contain several pre-defined views: Home, Charts, Alarms, and Settings. These views can easily be edited by selecting the view in the Project Browser and editing its Props to anything you want them to be. You can also go to the Page Configuration and edit the page configuration settings and docked view properties.

The images below show what the Menu Nav and Web Nav look like in a Perspective Session.

Web Nav

The Web Nav is good for small size project structures where you only have a few main Views. It allows you to navigate the areas of your project using the tabs at the top of the screen. It has a docked view that contains tabs that are always open to do navigation, and the main view which fills the rest of the space. It is a flat structure similar to the what you see on many webpages.



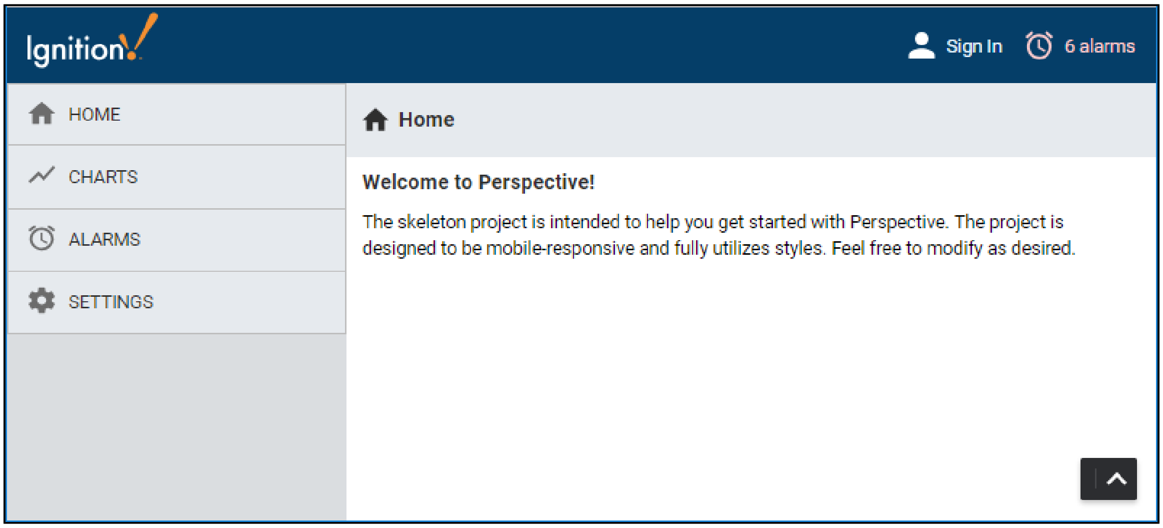
Home

Welcome to Perspective!

The skeleton project is intended to help you get started with Perspective. The project is designed to be mobile-responsive and fully utilizes styles. Feel free to modify as desired.

Perspective Menu Nav

The Menu Nav template is a hierarchical view of groups of information that can be configured to expand submenu branches and menu items and is docked on the left side of the screen. The Menu Nav is good for medium and large project structures because you can expose important information at a glance and allow users to navigate and expand submenus if any exist. The menu automatically hides itself on smaller screens devices.



Home

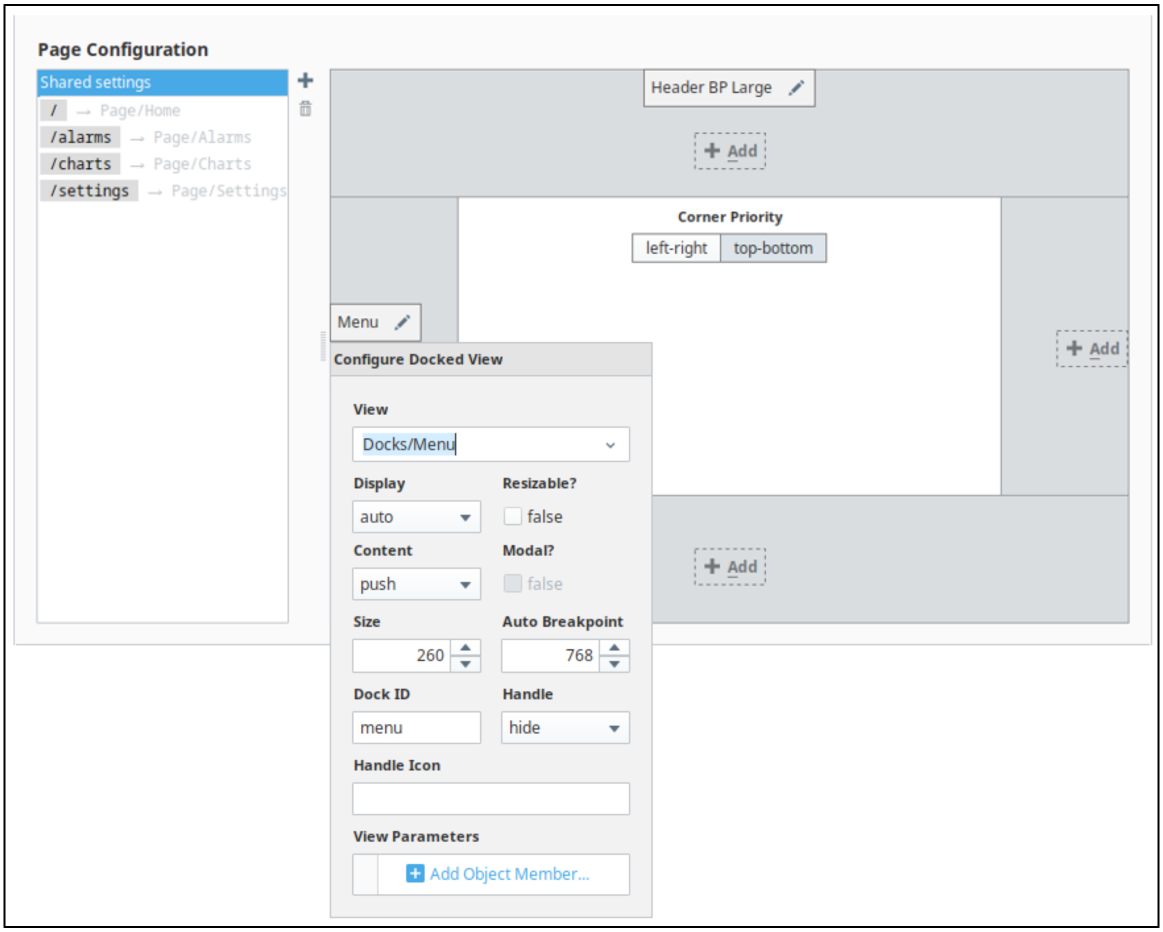
Welcome to Perspective!

The skeleton project is intended to help you get started with Perspective. The project is designed to be mobile-responsive and fully utilizes styles. Feel free to modify as desired.

Editing Properties and Page Configuration Settings

To edit the Menu Nav and Web Nav menus, first you have to have pages configured in your project. Once you have pages configured, you can go to the Navigation view in the Project Browser, select the component and change the corresponding component props in the Perspective Property Editor to add, remove, or update the menu items.

Page Configuration Settings



Page Configuration

Shared settings

/ → Page/Home
/alarms → Page/Alarms
/charts → Page/Charts
/settings → Page/Settings

Header BP Large

Corner Priority: left-right | top-bottom

Menu

Configure Docked View

View: Docks/Menu

Display: auto

Resizable?: □ false

Content: push

Modal?: □ false

Size: 260

Auto Breakpoint: 768

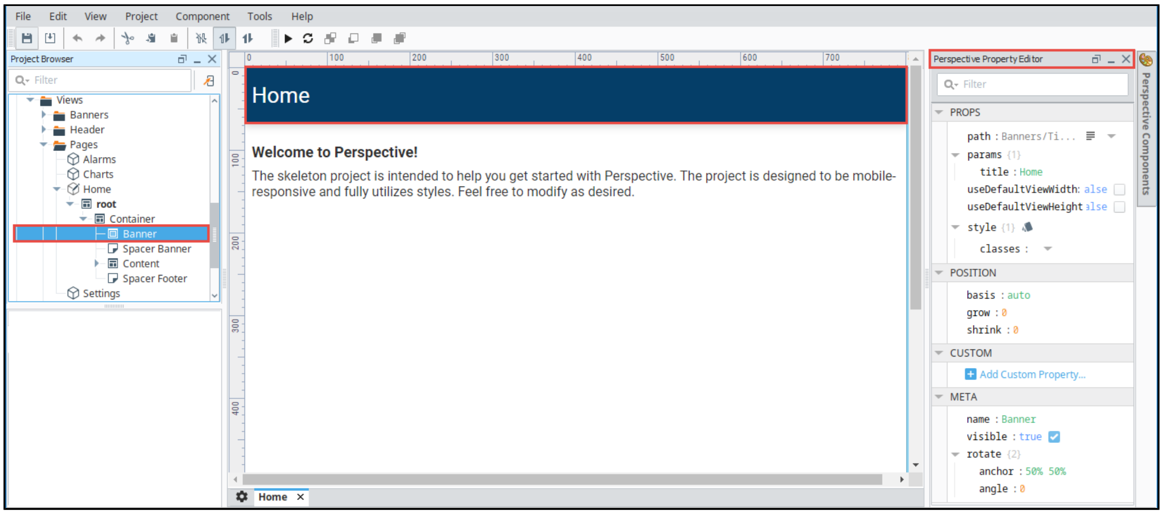
Dock ID: menu

Handle: hide

Handle Icon:

View Parameters

Project Browser and Project Editor



Home

Welcome to Perspective!

The skeleton project is intended to help you get started with Perspective. The project is designed to be mobile-responsive and fully utilizes styles. Feel free to modify as desired.

PROPS
path : Banners/Ti...
params {1}
title : Home
useDefaultViewWidth: alse □
useDefaultViewHeight alse □
style {1}
classes :

POSITION
basis : auto
grow : 0
shrink : 0

CUSTOM
Add Custom Property...

META
name : Banner
visible : true ☑
rotate {2}
anchor : 50% 50%
angle : 0

Vision Templates

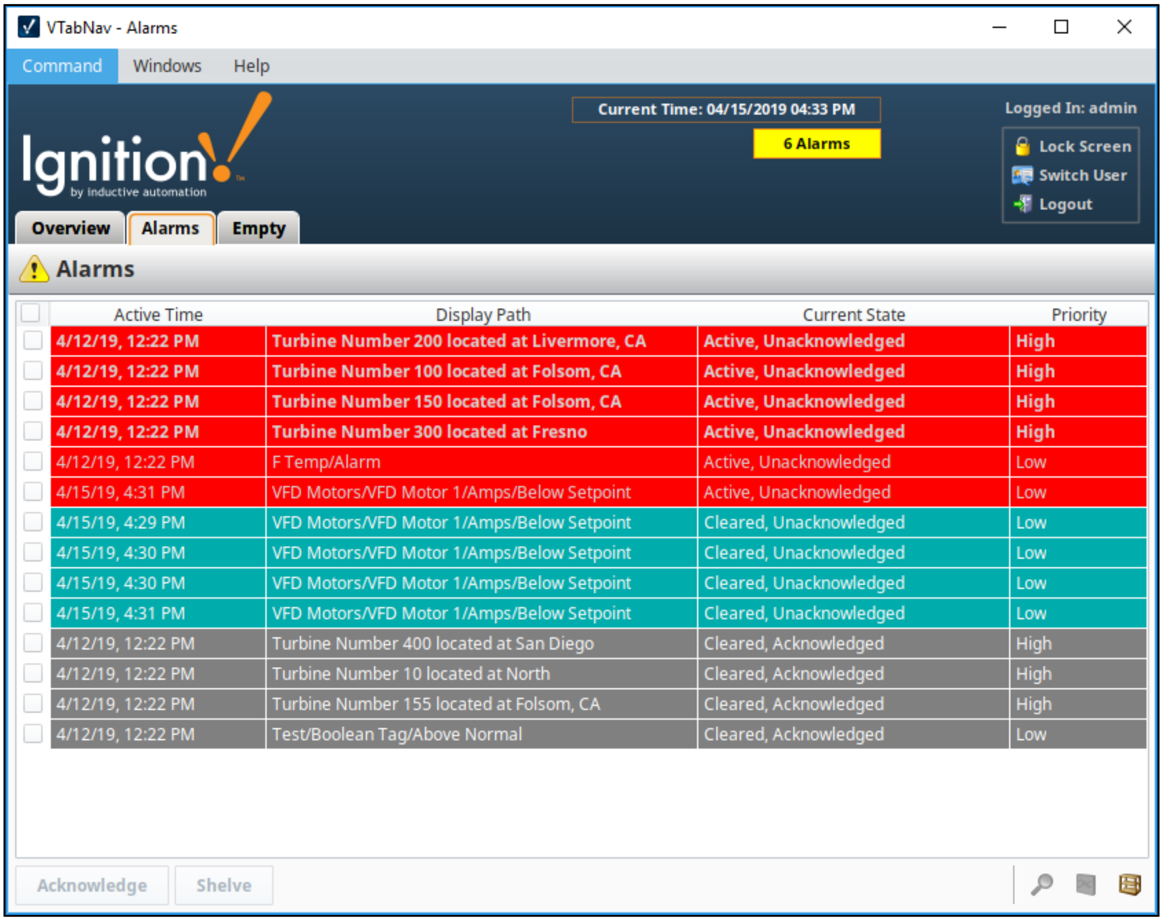
There are three Vision Project Templates to choose from to help kick start your project: **2-Tier Tab Nav**, **Tab Nav** and **Tree Nav.**

The Vision templates, just like the Perspective templates contain several pre-defined menu objects. These objects can easily be edited by selecting the object in the Project Browser by editing its properties and changing them to anything you want them to be.

The images below show what each of the Vision project templates look like in a Vision Client.

Tab Nav

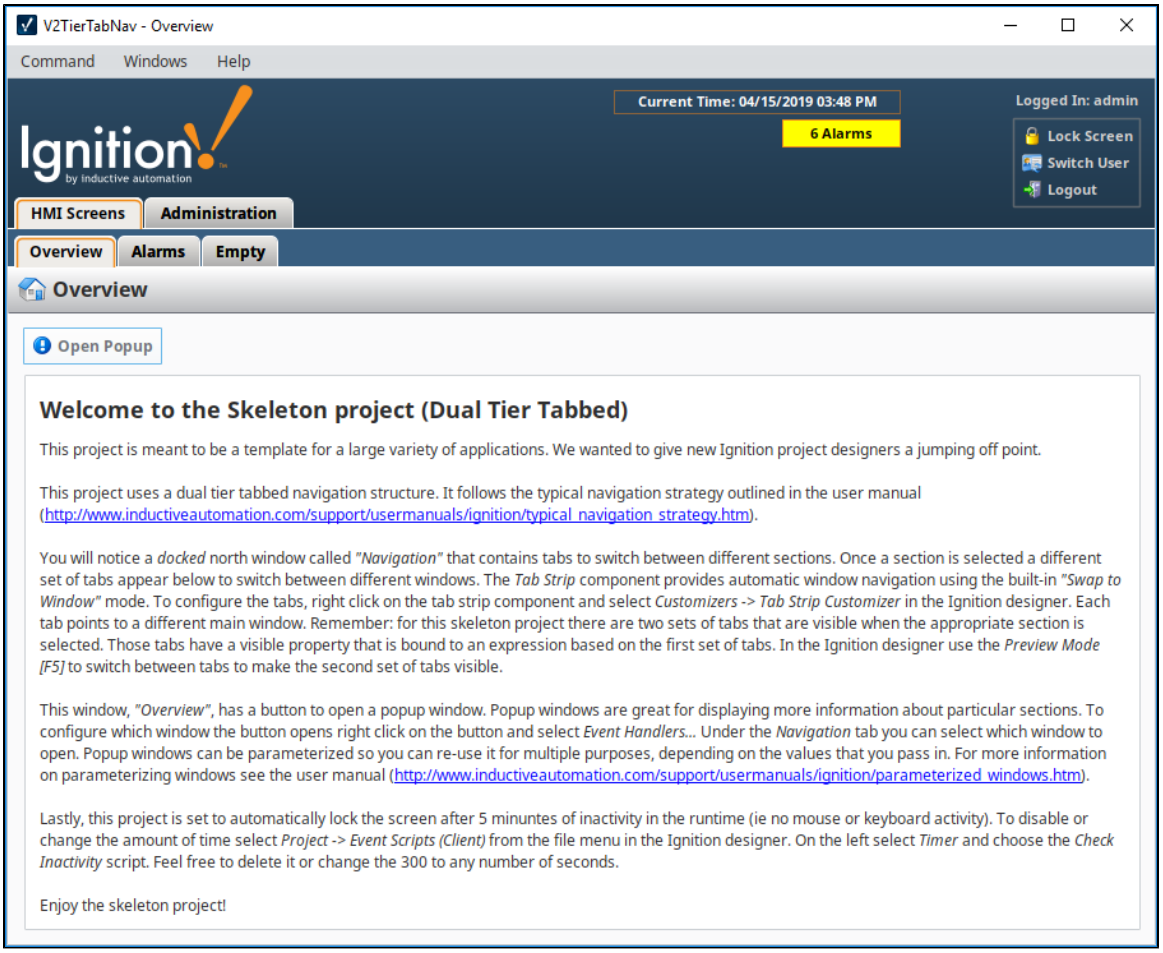
The Tab Nav is good for small size project structures where you only have a few main windows. It allows you to navigate the areas of your project using the tabs. The Tab Nav project template has a docked window that contains tabs that are always open to do navigation, and the main window which fills the rest of the space.



□	Active Time	Display Path	Current State	Priority
□	4/12/19, 12:22 PM	Turbine Number 200 located at Livermore, CA	Active, Unacknowledged	High
□	4/12/19, 12:22 PM	Turbine Number 100 located at Folsom, CA	Active, Unacknowledged	High
□	4/12/19, 12:22 PM	Turbine Number 150 located at Folsom, CA	Active, Unacknowledged	High
□	4/12/19, 12:22 PM	Turbine Number 300 located at Fresno	Active, Unacknowledged	High
□	4/12/19, 12:22 PM	F Temp/Alarm	Active, Unacknowledged	Low
□	4/15/19, 4:31 PM	VFD Motors/VFD Motor 1/Amps/Below Setpoint	Active, Unacknowledged	Low
□	4/15/19, 4:29 PM	VFD Motors/VFD Motor 1/Amps/Below Setpoint	Cleared, Unacknowledged	Low
□	4/15/19, 4:30 PM	VFD Motors/VFD Motor 1/Amps/Below Setpoint	Cleared, Unacknowledged	Low
□	4/15/19, 4:30 PM	VFD Motors/VFD Motor 1/Amps/Below Setpoint	Cleared, Unacknowledged	Low
□	4/15/19, 4:31 PM	VFD Motors/VFD Motor 1/Amps/Below Setpoint	Cleared, Unacknowledged	Low
□	4/12/19, 12:22 PM	Turbine Number 400 located at San Diego	Cleared, Acknowledged	High
□	4/12/19, 12:22 PM	Turbine Number 10 located at North	Cleared, Acknowledged	High
□	4/12/19, 12:22 PM	Turbine Number 155 located at Folsom, CA	Cleared, Acknowledged	High
□	4/12/19, 12:22 PM	Test/Boolean Tag/Above Normal	Cleared, Acknowledged	Low

2-Tier Tab Nav

2-Tier Tab is good for small and regular size project structures where windows are grouped. The Tab Nav project template is similar to the Tab Nav template only it has a second tier of tabs added. It contains a second level of tabs allowing you to navigate around various areas of your project. The 2-Tier Tab Nav project template has a docked window that contains tabs that show and hide based on selection that are always open to do navigation, and the main window which fills the rest of the space.



Welcome to the Skeleton project (Dual Tier Tabbed)

This project is meant to be a template for a large variety of applications. We wanted to give new Ignition project designers a jumping off point.

This project uses a dual tier tabbed navigation structure. It follows the typical navigation strategy outlined in the user manual (http://www.inductiveautomation.com/support/usermanuals/ignition/typical_navigation_strategy.htm).

You will notice a *docked* north window called *"Navigation"* that contains tabs to switch between different sections. Once a section is selected a different set of tabs appear below to switch between different windows. The *Tab Strip* component provides automatic window navigation using the built-in *"Swap to Window"* mode. To configure the tabs, right click on the tab strip component and select *Customizers -> Tab Strip Customizer* in the Ignition designer. Each tab points to a different main window. Remember: for this skeleton project there are two sets of tabs that are visible when the appropriate section is selected. Those tabs have a visible property that is bound to an expression based on the first set of tabs. In the Ignition designer use the *Preview Mode [F5]* to switch between tabs to make the second set of tabs visible.

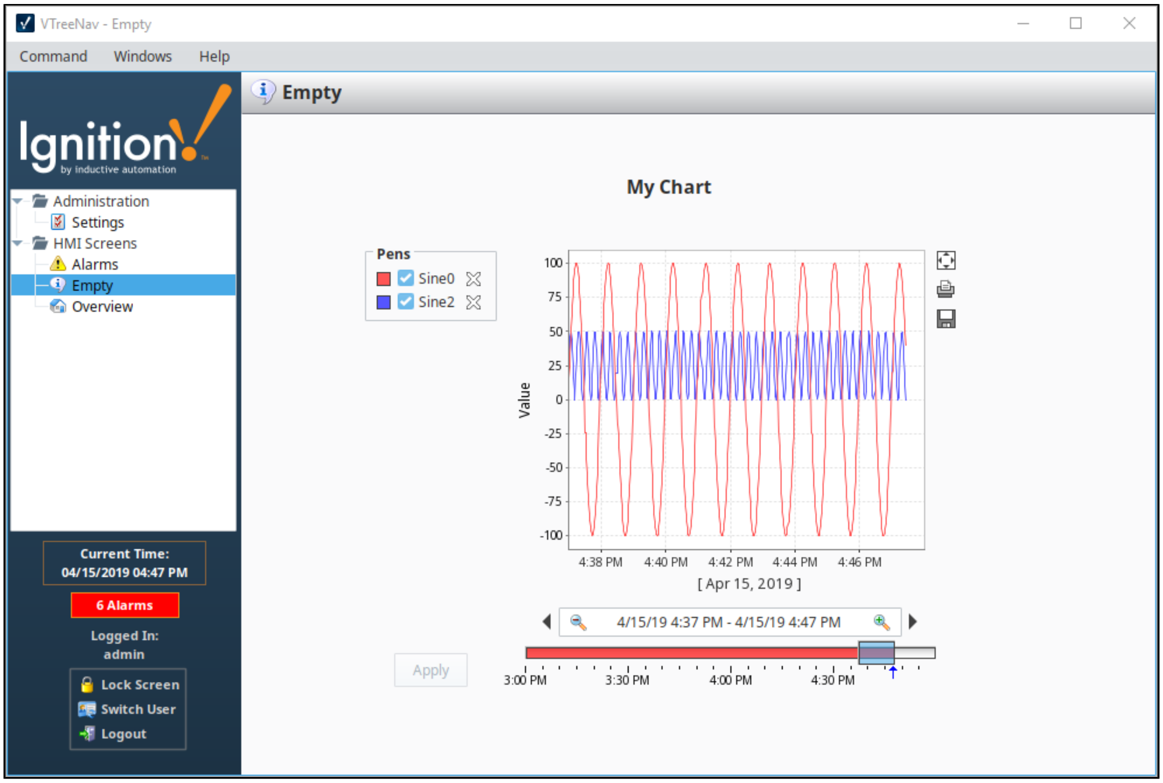
This window, *"Overview"*, has a button to open a popup window. Popup windows are great for displaying more information about particular sections. To configure which window the button opens right click on the button and select *Event Handlers...* Under the *Navigation* tab you can select which window to open. Popup windows can be parameterized so you can re-use it for multiple purposes, depending on the values that you pass in. For more information on parameterizing windows see the user manual (http://www.inductiveautomation.com/support/usermanuals/ignition/parameterized_windows.htm).

Lastly, this project is set to automatically lock the screen after 5 minuntes of inactivity in the runtime (ie no mouse or keyboard activity). To disable or change the amount of time select *Project -> Event Scripts (Client)* from the file menu in the Ignition designer. On the left select *Timer* and choose the *Check Inactivity* script. Feel free to delete it or change the 300 to any number of seconds.

Enjoy the skeleton project!

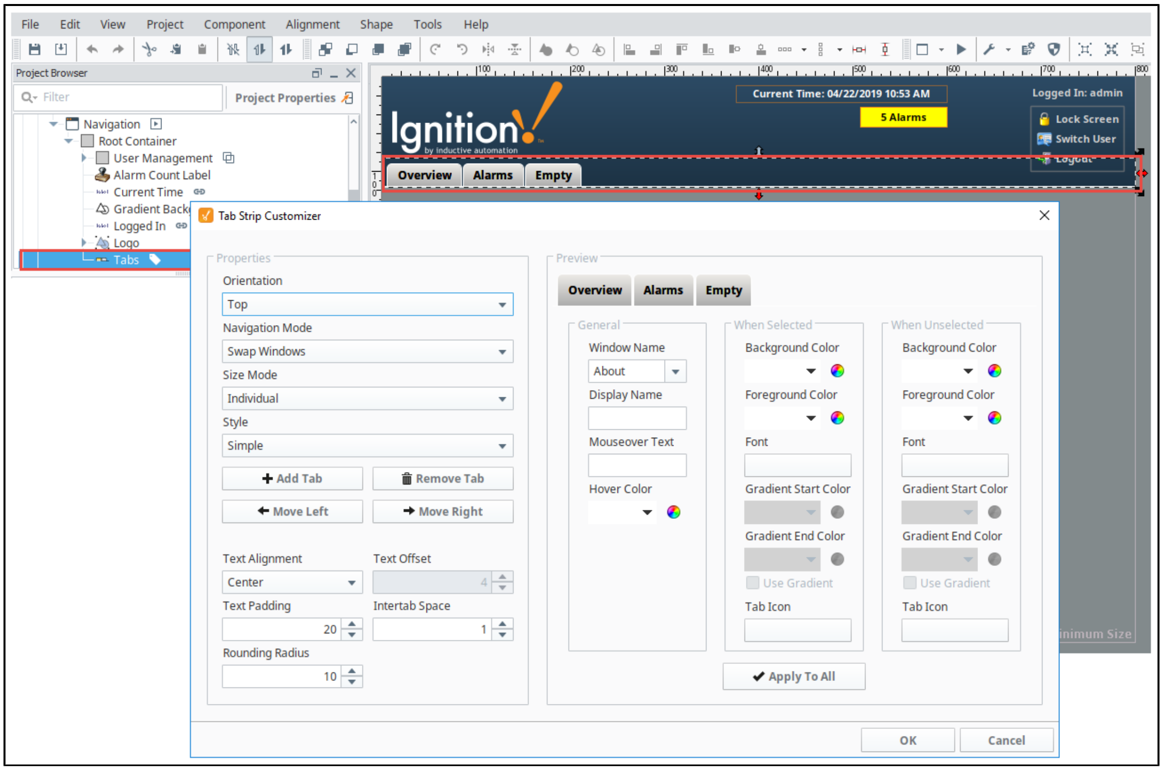
Tree Nav

The Tree Nav project template is a hierarchical view of groups of information that can be configured to expand submenu branches and menu items and is docked on the left side of the screen. It is great for medium and large project structures because you can view the entire project structure at a glance allowing you to navigate to any structure within the tree view. The tree structure uses folders to group main windows, and can be as many levels deep as you need.



Adding Windows to the Navigation

To edit the Tab Nav and Tree Nav menus, you just need to edit the properties of the navigation components. You can go to the Navigation window in the Project Browser, select the Tab Strip, and open the Tab Strip Customizer to change the corresponding component props to add, remove, or update the menu items.



Orientation: Top
Navigation Mode: Swap Windows
Size Mode: Individual
Style: Simple
Text Alignment: Center
Text Offset: 4
Text Padding: 20
Intertab Space: 1
Rounding Radius: 10

General
Window Name: About
Display Name
Mouseover Text
Hover Color

When Selected
Background Color
Foreground Color
Font
Gradient Start Color
Gradient End Color
☐ Use Gradient
Tab Icon

When Unselected
Background Color
Foreground Color
Font
Gradient Start Color
Gradient End Color
☐ Use Gradient
Tab Icon

Related Topics ...

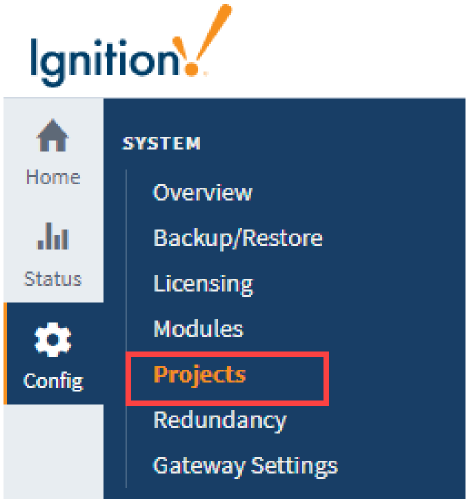
- Quick Start Guide
- Designer
- Pages in Perspective
- Navigation Strategies in Vision

Project Settings

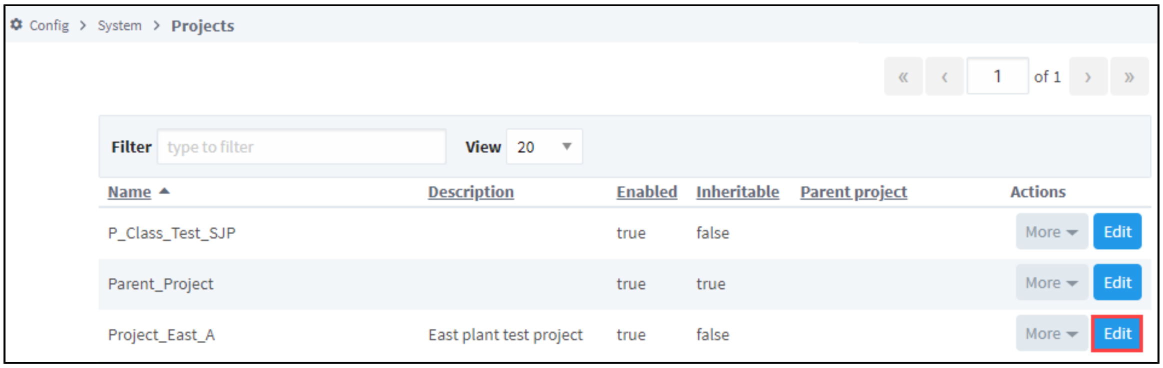
Project Settings

When you create a new project, the Open/Create Project window captures most of the project settings. See Creating a Project in the Designer. If you want to view or edit your project settings once your project is saved, go to the **Config** section on the Gateway webpage and select **System > Projects.**

On this page ...

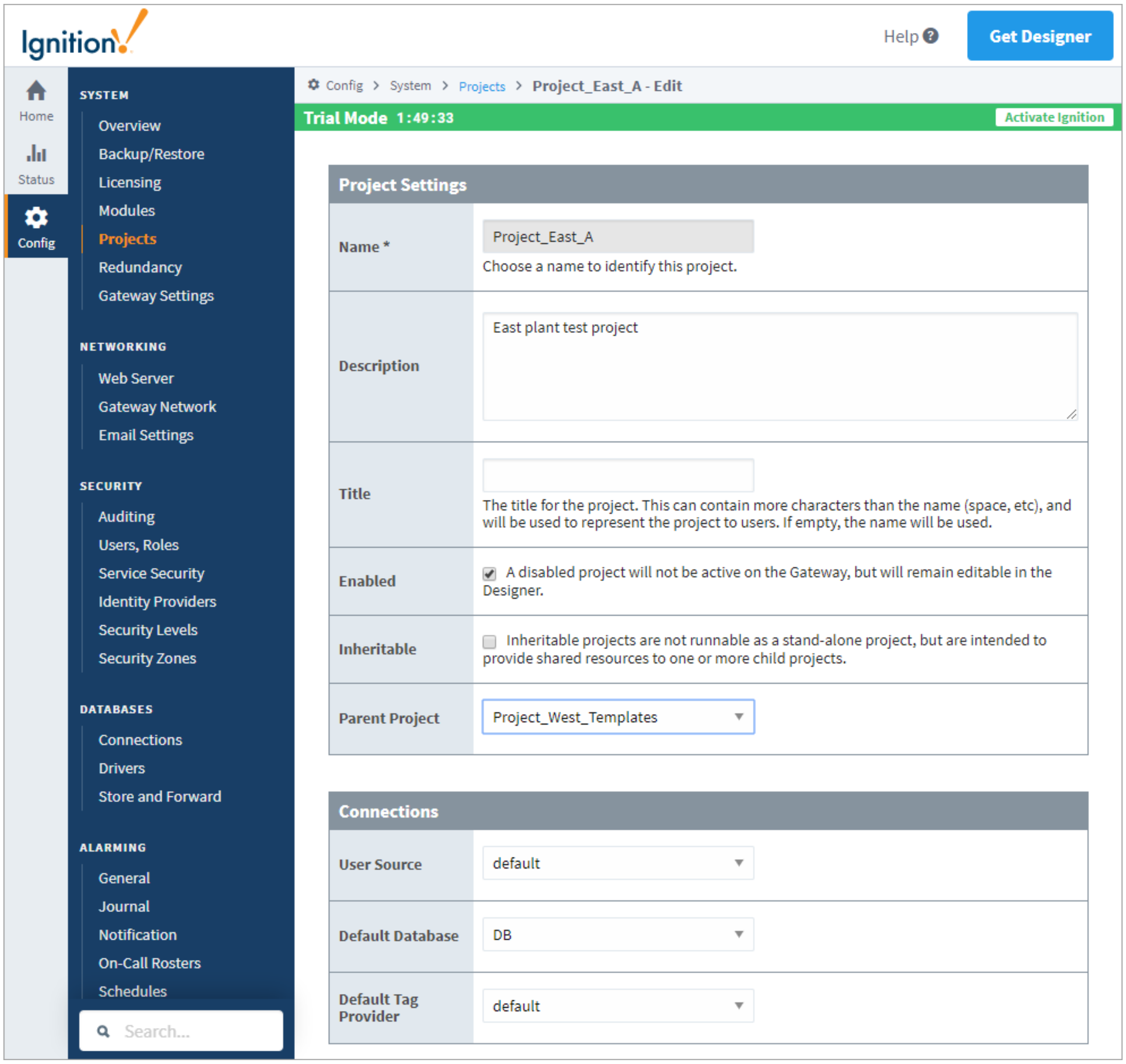


Click on the **Edit** to the right of the project name.



Name ▲	Description	Enabled	Inheritable	Parent project	Actions
P_Class_Test_SJP		true	false		More ▾ Edit
Parent_Project		true	true		More ▾ Edit
Project_East_A	East plant test project	true	false		More ▾ Edit

This opens up the project settings page.



Project Settings

Name *	Project_East_A Choose a name to identify this project.
Description	East plant test project
Title	The title for the project. This can contain more characters than the name (space, etc), and will be used to represent the project to users. If empty, the name will be used.
Enabled	☑ A disabled project will not be active on the Gateway, but will remain editable in the Designer.
Inheritable	☐ Inheritable projects are not runnable as a stand-alone project, but are intended to provide shared resources to one or more child projects.
Parent Project	Project_West_Templates

Connections

User Source	default
Default Database	DB
Default Tag Provider	default

Project Settings Table

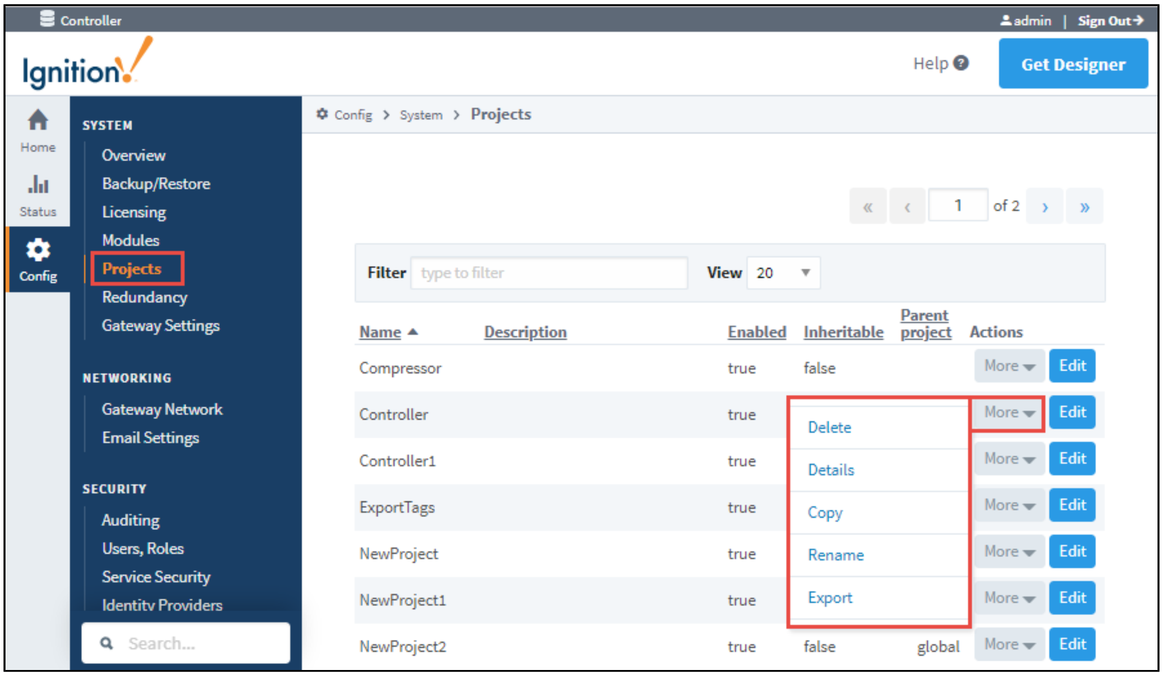
Project Settings	
Property	**Description**
Name	Name of the project. (read only)
Description	Brief description of the project (optional). This description can be viewed on the Open/Create Project screen when you over over the Information ⓘ icon.
Title	Title for the project. (optional) This is the name that will be displayed on the launch page of the Gateway and in the runtime Client or Session. There are no restrictions to special characters or spaces. If no title is specified, the project name will be used instead.
Enabled	A disabled project will not be active on the Gateway, but will remain editable in the Designer.
Inheritable	Inheritable projects are not runnable as a stand-alone project, but are intended to provide shared resources to one or more child projects.
Parent Project	Each project can have a parent project, and will inherit all of the resources of that parent project.
Connections	

User Source	A group of users with their associated roles. Security policies are defined in terms of these roles.
Default Database	Database to be used for historical data logging, reporting, storing alarm logs, and Tags storage. You can also query existing data and update data in the database.
Default Tag Provider	Identifies a Tag database (a collection of Tags) and a name.

Managing Projects

Once you have a project or two (or twenty) setup, you might want to change some of the base settings. You can manage the projects from the **Config** section of Gateway by going to **System > Projects**. The **Projects** window displays all your projects. To the right of the project name, click the **More** button, and here you can **Edit**, **Delete**, view **Details**, **Copy**, **Rename** or **Export** your project.

> ⚠ You can manage many of these settings in the Designer. See the Project Properties page for more information.



Name ▲	Description	Enabled	Inheritable	Parent project	Actions
Compressor		true	false		More Edit
Controller		true			More Edit
Controller1		true			More Edit
ExportTags		true			More Edit
NewProject		true			More Edit
NewProject1		true			More Edit
NewProject2		true	false	global	More Edit

Viewing the Details of a Project

To view the project details, go to the **Config** section of the Gateway webpage, and select **System > Projects**. Find your project, press the **More** button to the right of the project name, and select **Details**. The **Details** link on the **Project** window takes you to the **Project details for '<Project Name>'** page where you can choose to view the **Project Settings**.

Editing a Project

To edit some of the project settings, go to the **Config** section of Gateway, and then to **System > Projects**. You will see a list of all your projects. Click **Edit** to the right of project name in the list. The **Project - Edit** page is displayed. You can now change some of the project settings such as project Description, Title, Enabled, etc. You can also change connections to the User Source, Default Database, and Default Tag Provider.

Recommend Changing the Title, not the Project Name

It is not advisable to change the Project Name after it's been created, instead, change the Title property if you want to change how the project appears. Shortcuts that refer to the project will no longer work if the project name is changed.

Many of these settings can be modified in the Designer too, but it is important to note that you cannot change any of these project settings from the Gateway if the project is currently open in a Designer.

Deleting a Project

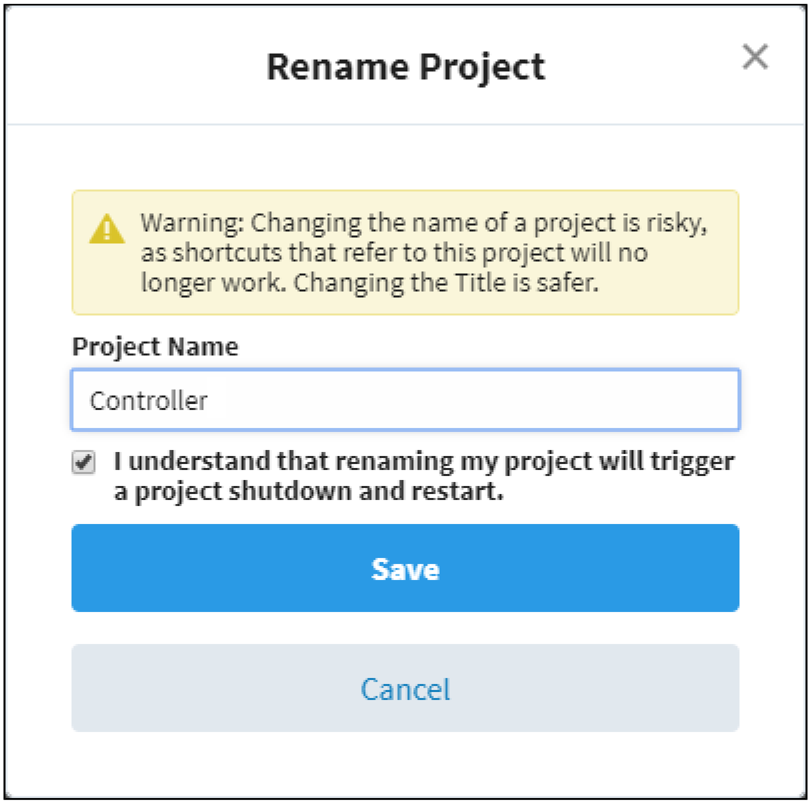
Be aware that once you delete a project, it cannot be undone, a deleted project is gone forever (unless it can be recovered from a project export, Gateway backup or a Gateway scheduled backup). Always make a project export or Gateway Backup before deleting a project.
To delete a project, go to the **Config** section of Gateway, and then to **System > Projects**. You will now see a list of all your projects. To the right of the project name in the list, look for **Delete** and click on it. This deletes your project.

Copying a Project

Copying your project is useful when you need a *snapshot* of a project before starting major changes, or for creating a starting point for a new project based on an old one. To copy a project, go to the **Config** section of Gateway, and then to **System > Projects**. You will now see a list of all your projects. To the right of the project name in the list, click **Copy**. This creates a copy of your project.

Renaming a Project

Changing the name of a project is risky. Shortcuts that refer to the project will no longer work. It is advisible to change the Title which is much safer. If you decide to rename a project, go to the Gateway webpage, select **Config** and then **System> Projects**. To the right of the project name in the list, click **Rename**. This will open a Rename Project window. Read it carefully, and if you want to proceed, enter a new project name, mark the checkbox acknowledging that the project will shutdown and restart, then press **Save**.



Rename Project

Warning: Changing the name of a project is risky, as shortcuts that refer to this project will no longer work. Changing the Title is safer.

Project Name

Controller

☑ I understand that renaming my project will trigger a project shutdown and restart.

Project Shutdown and Restart

Renaming a project triggers a project shutdown and restart.

Exporting a Project

Project Export is a project backup. It takes less time than a Gateway backup and it's smaller. This exports your project as a **`.zip`** file. Once you have an exported file of your project, you can take it to any other Gateway and merge it in with other projects. Refer to the Project Export and Import page to learn exactly what is included in a project export.

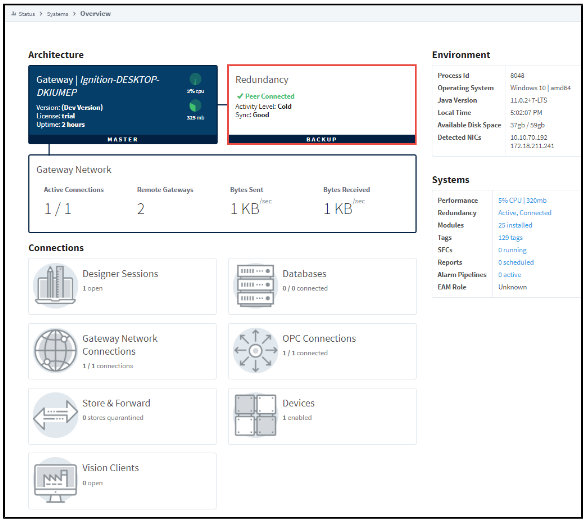
Related Topics ...

- Project Properties
- Project Export and Import

Ignition Redundancy

Ignition redundancy supports 2-node systems, this means there are two copies of the Ignition Gateway. One node (the Master Gateway) is considered the **master node** and the other (the Backup Gateway) is the **backup node**. Both nodes share the same **state** or configuration. In other words, all projects, Gateway settings, and so on are shared between the nodes. The master node manages the configuration then replicates it to the backup node.

When you have the redundant system in place, you can get detailed status information by going to the **Status** tab of the Gateway Webpage as shown here and looking at the system's map under **Systems > Overview**.



Gateway | *Ignition-DESKTOP-DKIUMEP*
3% cpu
325 mb
Version: **(Dev Version)**
License: **trial**
Uptime: **2 hours**

Redundancy
✓ Peer Connected
Activity Level: **Cold**
Sync: **Good**

Gateway Network

Active Connections	Remote Gateways	Bytes Sent	Bytes Received
1 / 1	2	1 KB/sec	1 KB/sec

Designer Sessions
1 open

Databases
0 / 0 connected

Gateway Network Connections
1 / 1 connections

OPC Connections
1 / 1 connected

Store & Forward
0 stores quarantined

Devices
1 enabled

Vision Clients
0 open

Process Id	8048
Operating System	Windows 10 | amd64
Java Version	11.0.2+7-LTS
Local Time	5:02:07 PM
Available Disk Space	37gb / 59gb
Detected NICs	10.10.70.192 172.18.211.241

Performance	5% CPU | 320mb
Redundancy	Active, Connected
Modules	25 installed
Tags	129 tags
SFCs	0 running
Reports	0 scheduled
Alarm Pipelines	0 active
EAM Role	Unknown

On this page ...

INDUCTIVE UNIVERSITY

How Redundancy Works

Watch the Video

Updating or Patching a Redundant Ignition Pair

Learn about updating redundant servers and how to make the process a success.

Link to Knowledge Base Article

Node Communication

The master and backup nodes communicate over TCP/IP. Therefore, they must be able to see each other over the network, through any firewalls that might be in place. All communication goes from the backup to the master node over the gateway network (default **port 8088** without SSL, **port 8060** with SSL). Therefore, that port must allow TCP listening on the master machine.

Configuration Synchronization

The master node maintains the official version of the system configuration. You must make all changes to the system on the master Gateway, the backup Gateway does not allow you to edit properties. Similarly, the Designer only connects to the master node.

When changes are made on the master, they are queued up to be sent to the backup node. When the backup connects, it retrieves these updates, or downloads a full system backup if it is too far out of date.

If the master node has modules that aren't present on the backup, they are sent across. Both types of backup transfers, **data only** and **full**, will trigger the Gateway to perform a soft reboot.

Runtime State Synchronization

Information that is only relevant to the running state, such as current alarm states, is shared between nodes on a differential basis so that the backup can take over with the same state that the master had.

On first connection or if the backup node falls too far out of sync, a full state transfer is performed. This information is light-weight and does not trigger a Gateway restart.

Status Monitoring

Once connected, the nodes begin monitoring each other for liveliness and configuration changes. While the master is up, the backup runs according to the **stand by activity level** in the settings.

When the master cannot be contacted by the backup for the specified amount of time, it is determined to be down and the backup assumes responsibility. When the master becomes available again, responsibility is dictated by the recovery mode and the master either takes over immediately or waits for user interaction.

System Activity

When a node is active, it runs fully, connecting to any configured OPC servers, and communicating with devices. When it is not active, its activity level is dictated by the settings, either **warm** or **cold**.

- In **warm** standby, the system runs as if it were active, with the exception of logging data or writing to devices, allowing for faster fail-over.
- In **cold** standby, the system connects to all OPC servers but does not subscribe to Tag values. The Ignition OPC UA server does not communicate with any device, but third party OPC UA servers may still have device connections. This allows the system to standby without putting additional load on the devices and network. Failover takes slightly longer, as Tags must be subscribed and initialized.

Historical Logging

Historical data presents a unique challenge when working with redundancy because it is never possible for the backup node to know whether the master is truly down or simply unreachable. If the master was running, but unreachable due to a network failure, the backup node becomes active and begins to log history at the same time as the master, who is still active.

In some cases this is OK because the immediate availability of the data is more important than the fact that duplicate entries are logged. But in other cases, it's desirable to avoid duplicates, even at the cost of not having the data available until information about the master state is available.

Ignition redundancy provides for both of these cases, with the **backup history level**, which can be either **Partial** or **Full**.

- In **Full** mode, the backup node logs data directly to the database.

- In **Partial** mode, however, all historical data is cached until a connection is reestablished with the master. At that time, the backup and master communicate about the uptime of the master, and only the data that was collected while the master was truly down is forwarded to the database.

Client Failover

Vision Clients

All Vision clients connect to the active node. When this system fails and is no longer available, they automatically re-target to the other node. The reconnection and session establishment procedures are handled automatically, but the user is notified that they have been transferred to a different node so that they can notify the system administrator that the system may need attention.

Perspective Sessions

Like Vision clients, Perspective sessions connect to the active node. When connection to the active node is lost, or the activity level of the Gateway changes from **active**, the session will simultaneously attempt to:

1. Re-establish the connection to the Gateway it was connected to, and check to make sure its activity level is **active**.
2. Monitor the backup Gateway. If the backup Gateway becomes reachable and **active** before the connection to the active Gateway can be re-established, the Perspective session navigates in the browser to the same project and page on the backup Gateway.

In This Section ...

Setting Up Redundancy

Enabling Redundancy

In redundancy, both nodes will share the exact same configuration state. When a Backup node connects to a Master node, the Backup will attempt to synchronize itself with the Master. Therefore, before you set up for redundancy the following should be considered:

1. **Start with a fresh install for the Backup node.**
   Because the current configuration of the Backup node will be overwritten, make sure that it does not contain anything valuable. It is a good idea to export any projects that are unique to the Backup before enabling redundancy.

2. **All system configurations relative to the Master node must also resolve on the Backup node.**
   For example, OPC UA connections and database connections must use addresses that resolve from both nodes, or any OPC-COM servers must be installed and configured identically on both nodes. This means using "localhost" in any of the database connections won't work. You should use the IP Address of the computer instead.

3. **Configure firewalls between the redundancy nodes.**
   Redundant systems need TCP connectivity between each other on the default gateway network ports. Turning off software firewalls or adding special exception rules for each others' addresses is required. The default Gateway Network port is **port 8088** (without SSL), and **port 8060** (with SSL), and the Backup node must be able to send outgoing data on that port. The port can be changed from Gateway Network settings.

Note: Two Edge Gateways can be set up with redundancy. An Edge Gateway can only failover to another Edge Gateway (not a standard Ignition Gateway). Also, an Edge Gateway cannot be used as backup to a Standard Ignition Gateway.

Note:

While the OS platform (i.e., Windows, OS X, Linux) for the Master and Backup can differ, it is recommended to have similar OS platforms. If the OS platforms do differ, the Windows machine should be the master system or else the Force Failover option will not work.

However, different versions of the same operating system such as Windows 10 and Windows 8 or OSX 10 and OSX 11 have full functionality.

On this page ...

Setting Up Redundancy

Watch the Video

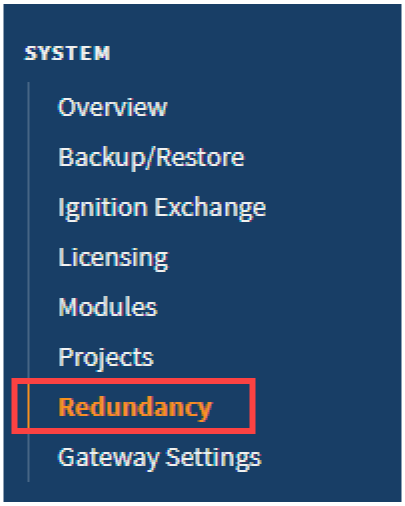
Setting Up Redundancy

Once the above points are done, setting up redundancy is fairly simple:

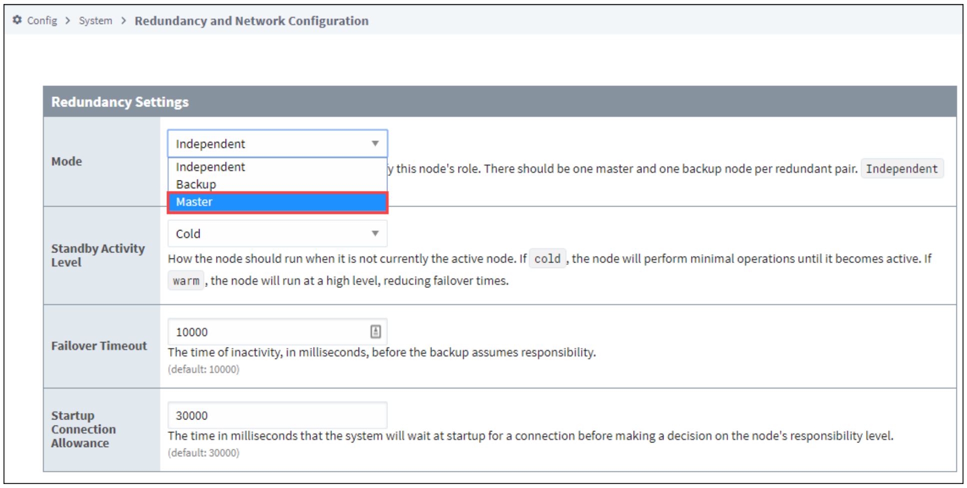
On the Master Gateway

1. Go to the **Config** section of the Master Gateway Webpage.

2. Select **System > Redundancy.**
   The Redundancy and Network Configuration page is displayed showing different sections and settings. See the table below for a description of all settings.



3. Change the following settings:
   - Under **Redundancy Settings**, set **Mode** to **Master**.
   - Optionally, configure any desired settings under **Master Node Settings**.



Mode: Independent

y this node's role. There should be one master and one backup node per redundant pair. Independent

Standby Activity Level: Cold

How the node should run when it is not currently the active node. If cold, the node will perform minimal operations until it becomes active. If warm, the node will run at a high level, reducing failover times.

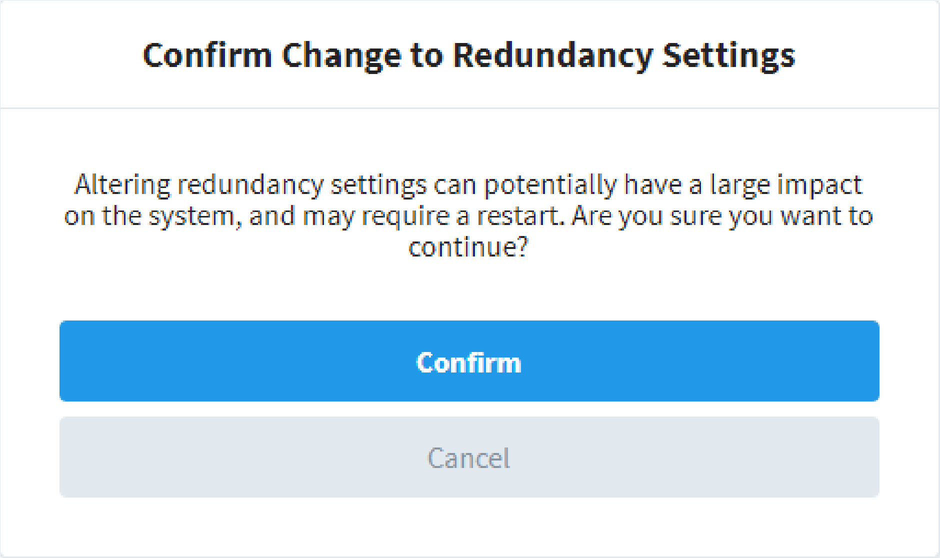
Failover Timeout: 10000

The time of inactivity, in milliseconds, before the backup assumes responsibility.
(default: 10000)

Startup Connection Allowance: 30000

The time in milliseconds that the system will wait at startup for a connection before making a decision on the node's responsibility level.
(default: 30000)

4. Click **Save Changes**. The Confirm change to Redundancy Settings page is displayed.



Confirm Change to Redundancy Settings

Altering redundancy settings can potentially have a large impact on the system, and may require a restart. Are you sure you want to continue?

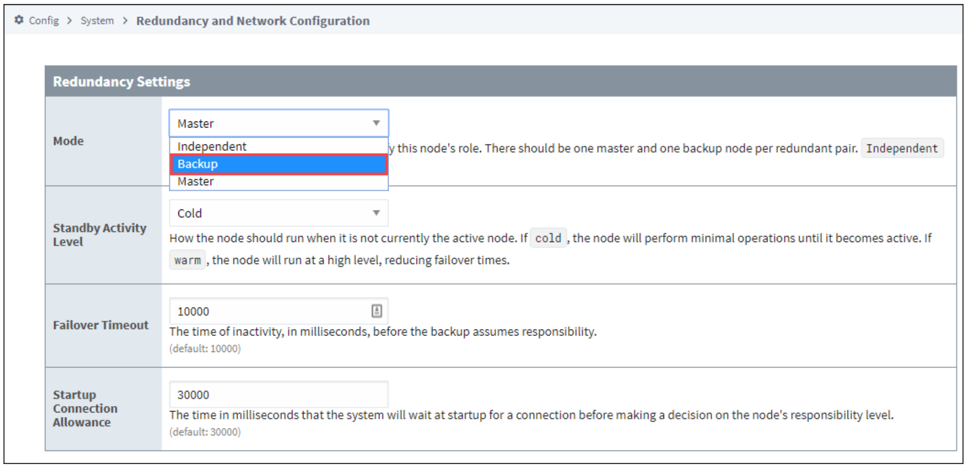
5. Click **Confirm** to apply your settings.

6. Go to the **Config** tab and select **System > Redundancy** to ensure the redundancy mode and state is properly set.

On the Backup Gateway

Do the exact same steps 1-6 above on the Backup Gateway Webpage, except replace step 3 with the following:

- Under **Redundancy Settings**, set **Mode** to **Backup**.
- Under **Backup Node Settings**, configure the Master Node Address and Port to point to the Master Gateway. The Master Node address should be a hostname or IP address. The Port setting (assuming default configurations) should be **8060** if using SSL, otherwise **8088**.



Config > System > Redundancy and Network Configuration

Redundancy Settings

Mode: Master (Independent / Backup / Master)
y this node's role. There should be one master and one backup node per redundant pair. Independent

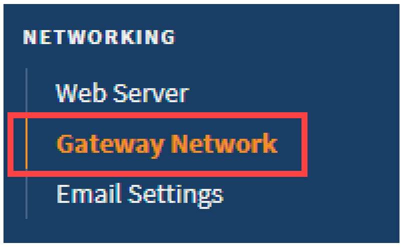
Standby Activity Level: Cold
How the node should run when it is not currently the active node. If cold, the node will perform minimal operations until it becomes active. If warm, the node will run at a high level, reducing failover times.

Failover Timeout: 10000
The time of inactivity, in milliseconds, before the backup assumes responsibility.
(default: 10000)

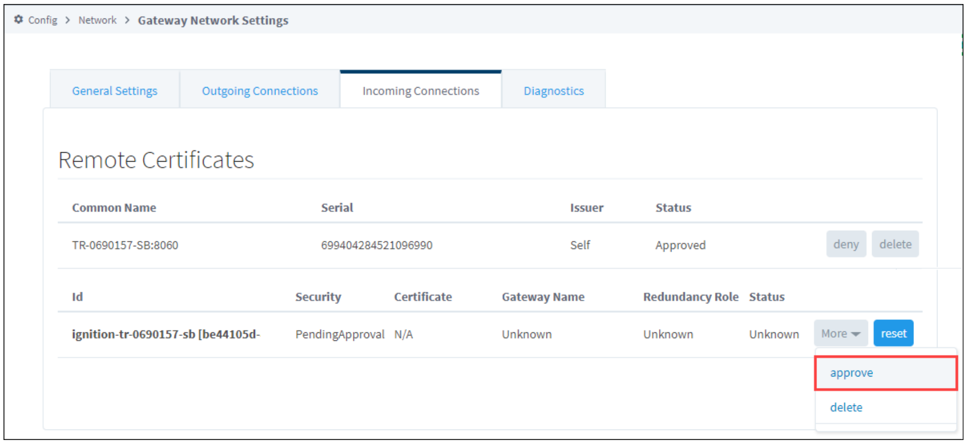
Startup Connection Allowance: 30000
The time in milliseconds that the system will wait at startup for a connection before making a decision on the node's responsibility level.
(default: 30000)

On the Master Gateway - Approve the Connection and Verify

1. Return to the **Config** section of the Master Gateway Webpage.
2. Select **Networking > Gateway Network**.



3. Navigate to the **Incoming Connections** tab. You should see a new incoming connection from the Backup Gateway. Find the connection, select **More**, and select **approve**.



Config > Network > Gateway Network Settings

General Settings | Outgoing Connections | Incoming Connections | Diagnostics

Remote Certificates

Common Name	Serial	Issuer	Status	
TR-0690157-SB:8060	699404284521096990	Self	Approved	deny delete

Id	Security	Certificate	Gateway Name	Redundancy Role	Status	
ignition-tr-0690157-sb [be44105d-	PendingApproval	N/A	Unknown	Unknown	Unknown	More reset

4. To verify the redundancy setup, that is, to ensure the Master and the Backup Gateways are connected, go the **Status** tab of the Gateway Webpage and click on **System > Redundancy**. The Redundancy page will show the connected nodes and their current states.

   After approving the connection, the Backup connects to the Master and downloads a system backup, then restarts. Once the restart is complete, the Backup node is synchronized and in communication with the Master.

Redundancy Settings

All redundancy settings are configured in the Gateway Webpage under the **Config** tab**, Systems > Redundancy**. Most settings are used by both the Master and Backup nodes, with their individual settings broken out into separate categories.

It is important to know that while the full system configuration is shared between nodes, redundancy settings are not shared between nodes. Therefore, it is perfectly acceptable to have different values for the same settings on the two nodes. For example, it is possible to have a different Standby Activity Level on both nodes, and, of course, the network settings will often be different.

> ⚠ The Master node shares all configuration with the Backup node, and this means that changes cannot be made to your project from the Backup. In fact, the Designer can never be opened from a Backup node, even if the Master is currently offline.

Redundancy Settings	
Mode	Enable or disable redundancy, and specify this node's role. There should be one master and one backup node per redundant pair. Independent turns off redundancy. **Independent** - Redundancy is not enabled and this Ignition system runs as an independent node. **Master** - This is the Master node, who listens for a connection from the Backup node, and is in charge of managing system synchronization. **Backup** - This is the Backup node, who will connect to the Master and receive system updates.
Standby Activity Level	How the node should run when it is not currently the **Active** node. **Cold** - The node performs minimal operations until it becomes active. The purpose is to minimize the load on the network and on devices. **Warm** - The node runs at a high level, reducing failover times.
Failover Timeout	The time of inactivity, in milliseconds, before the backup assumes responsibility.
Startup Connection Allowance	The time in milliseconds that the system will wait at startup for a connection before making a decision on the node's responsibility level. This is used to prevent unnecessary switch over caused by a node starting as active, only to connect and find that the other node is active, resulting in one of the nodes being de-activated. It is important to note that this setting can interfere with the Master Recovery Mode: • If the Master is active, it will always request the Backup to de-activate. • If this setting is low, or 0, the Master will always become active before connecting to the Backup, and thus "manual recovery" will not be possible.

Network Settings	
Auto Detect Network Interface	If true, the system will automatically select which network interface to use. Most commonly disabled on systems with multiple network cards, in order to explicitly specify which interface to use. If false, the system will bind itself to the interface of the specified address.
Network Bind Interface	The IP address of the network interface to use for redundancy. Only used if "Auto Detect" is false.
Autodetect HTTP Address	The following feature is new in Ignition version **8.0.3** Click here to check out the other new features This setting has been moved to the Web Server page in the Gateway > Config> Networking section.

HTTP Addresses	The following feature is new in Ignition version **8.0.3** Click here to check out the other new features This setting has been moved to the Web Server page in the Gateway > Config> Networking section.

Master Node Settings	
Recovery Mode	How the Master node resumes responsibility after starting again. **Automatic** - The Master automatically takes back responsibility, and becomes active. The Backup node goes to standby. **Manual** - The Backup node is allowed to stay active. The Master will become active if the Backup node fails, or if the user requests a switchover from the Gateway configuration page.
Runtime Update Buffer Size	How many "runtime state" updates can be queued in memory before the system stops tracking and a full transfer is performed. These updates represent information that the other node should have in order to have the same running state as the Master when it's forced to take over. This is most often the values of static Tags and the current alarm state. Given that the update buffer is only used once the nodes are connected, the default value is usually fine, and only needs to be increased on systems that may have many alarms that change together, or many static Tag writes.
Config Update Queue Size	The maximum size (in megabytes) of config updates allowed before a full transfer is performed.

Backup Node Settings	
Master Node Address	The address of the Master Ignition system.
Port	The Gateway Network port used by the Master to listen on. For the Backup, the port to connect to on the Master.
Use SSL	Use SSL to connect to the remote machine.
Ping Rate	How often, in milliseconds, to send a message from the Backup to the Master.
Ping Timeout	The maximum time, in milliseconds, allowed for a ping response. Pings that time out are counted as missed pings.
Ping Max Missed	The amount of missed pings that will force the connection to the master to be considered faulted.
Websocket Timeout	The maximum time, in milliseconds, allowed for a new web socket to connect to the Master.
HTTP Connect Timeout	The maximum time, in milliseconds, allowed to establish an HTTP connection to the Master.
HTTP Read Timeout	The maximum time, in milliseconds, allowed to read or send HTTP data to the Master.
History Mode	How history is treated by the Backup system. If **Full**, history will be stored normally, as it would be on the Master system. If **Partial**, history will be cached until the Master is available again and the Backup node is able to determine the exact time that the Master was down.

Troubleshooting

Redundancy Connectivity

When the two redundant nodes are connected, you will be able to see their state details in the Status section of the Gateway Webpage. There are also various other places where the redundancy state is shown as **connected**.

If the two nodes cannot connect, check the following:

- Verify that the Master address is correct in the Backup. Try to ping the Master machine from the Backup machine, and verify that you're using the correct address for the network card that the Master is connected through.
- If using system names (or domain names), verify that the name is resolving to the correct address by performing a ping.
- Verify that the firewall on the Master is set to allow TCP traffic to the designated port.
- Verify that the Backup is not connecting and then immediately disconnected for some reason.
- Viewing the error log in the Gateway console section should show this. If errors are occurring at regular intervals, look at the message for an indication of what is happening. An example of a potential problem is when the failover time is set too low for the given network, which results in many socket read timeout exceptions, which in turn leads to many disconnect/reconnect attempts.
- If errors are occurring, but the cause isn't clear, contact Inductive Automation Support.

Advanced Troubleshooting

A variety of loggers can be found under the Gateway console section by going to "Levels" and searching for "Redundancy". By setting these loggers to a finer level, more information will be logged to the console. This is generally only useful under the guidance of Inductive Automation support personnel, though more advanced users may find the additional logged information helpful.

Related Topics ...

- Database Considerations

Database Considerations

Ignition Database Requirements

Given that many parts of the Ignition system interact with the database, it's important to give some thought as to how it will be used when redundancy is turned on, and the different database architectures that are possible.

When evaluating database architectures for use with Ignition, it's important to look carefully at how the system will use the database. Which pieces are critical? Which pieces are "optional" so that the system continues to function while the database is down? Which pieces can operate in "read-only" mode if necessary?

Ignition uses the database for many purposes. Here are some common areas where they are used, and how availability can impact the system:

Tags

Tags rely on the database for Tags that execute queries. These Tags will error out if the database is unavailable, but the status and control functionality of the system will function on the whole.

History - Tags and Other

All history in Ignition goes through the Store-and-Forward system, meaning that it will be cached until the database is available. However, while the data is cached, it will be unavailable to view or analyze on the clients. Therefore, when looking at how the database should be set up, it is necessary to determine how crucial rapid-availability of the data is.

Alarming

The alarm status system does not reside in the database, so it will continue to function if the connection is down. Alarm Journal information will go through the Store-and-Forward system as history data.

Project Screens

Almost all projects use database access for providing information on screens. These queries will error out as long as the database is unavailable. Screens that only use Tags (in an internal provider) will continue to function, so it would be beneficial to make a distinction between status screens and history screens, if a failover database is not used.

On this page ...

Database Considerations

Watch the Video

Database Architectures

Single Shared Server

A single database server is used. Any Ignition Gateways will use it, so it is expected to be available even when one of the nodes is not. For that reason, it almost always resides externally, on a separate server machine. This arrangement is the easiest to use with Ignition. A single database connection configured on the master will be replicated to the backup, and both nodes will use the connection as necessary.

Clustered/Replicated Database Servers

There is a wide variety of capabilities supported by the different brands of database servers. To obtain fault-tolerance on the database front, it is usually necessary to have some sort of cluster/replication system in place. However, it can be very important to examine how Ignition is using the databases, and what capabilities the clustering solution provides.

For example, in many replication scenarios, the master database copies data to the backup. The backup can be used for read purposes, but new data inserted will not be replicated back to the master. Therefore, it is possible to have a failover connection to the backup database, so that clients will continue to receive data, but it would be necessary to run in partial history mode, so that the historical data was cached and inserted only to the master database. The failover connection would be set to standard mode, so the primary connection would be used when possible.

In a more complete cluster environment, where writes to either node would be replicated, a sticky failover connection could be used with full history mode.

Pertinent Settings

When working with various database architectures, there are a few settings in various parts of the system that are important.

Database Connection Settings - Failover Datasource

Any database connection can have a failover datasource. If the main connection is unavailable, any queries executed on it will pass through to the secondary connection. In this way, a secondary database can be used when the first is not available, and the system will continue to function. It is important to note that everything passed through to the failover will function normally- no special considerations will be made. For example, the system won't cache data for the primary connection, it will forward it to the secondary. In cases where you want to allow reading from the secondary database, but not writing, you can set up another connection directly to the first database, with no failover, and set all of your write operations to use that.

Clustering Settings - History Mode

The history mode dictates how history will be treated when the node is not active. If partial, the data will be cached, and only forwarded when the master node is available. This mode can be used to prevent data from being inserted into a backup database in some cases. This setting can be found on the **Redundancy** page under the **Config** section of the Gateway.

Related Topics ...

- Redundant Licensing
- Setting Up Redundancy

Redundant Licensing

Types of Redundant Licensing

When working with Redundancy, both nodes will require a license. However, there are two approaches that are detailed below. In both cases, the license for the Backup node generally contains most, if not all, the same modules as the Master node. For example, if the Master is storing history with the SQL Bridge module, then the Backup would also require the module in the event a failover occurs, otherwise, data would be lost.

However, non-critical modules do not need to be added to the Backup License. The Master license could make use of the Symbol Factory module, and since the Designer only connects to the Master node, there is no reason to add this module to the Backup node.

Two Standard Licenses - Classic Redundancy

Traditionally, Redundancy involves two standard licenses: one license will be applied to the Master Gateway, and the other license will be applied to the Backup Gateway. Since two standard licenses are being used, this approach allows users to disable Redundancy, and set both Gateways to independent modes.

Backup-Only Licenses

Backup-Only licenses are available for purchase. This license forces the Gateway into a Backup mode, and the mode can not be changed while the license is applied. The benefit of this type of license is that they come at a discounted price. For more details, contact your Account Representative.

On this page

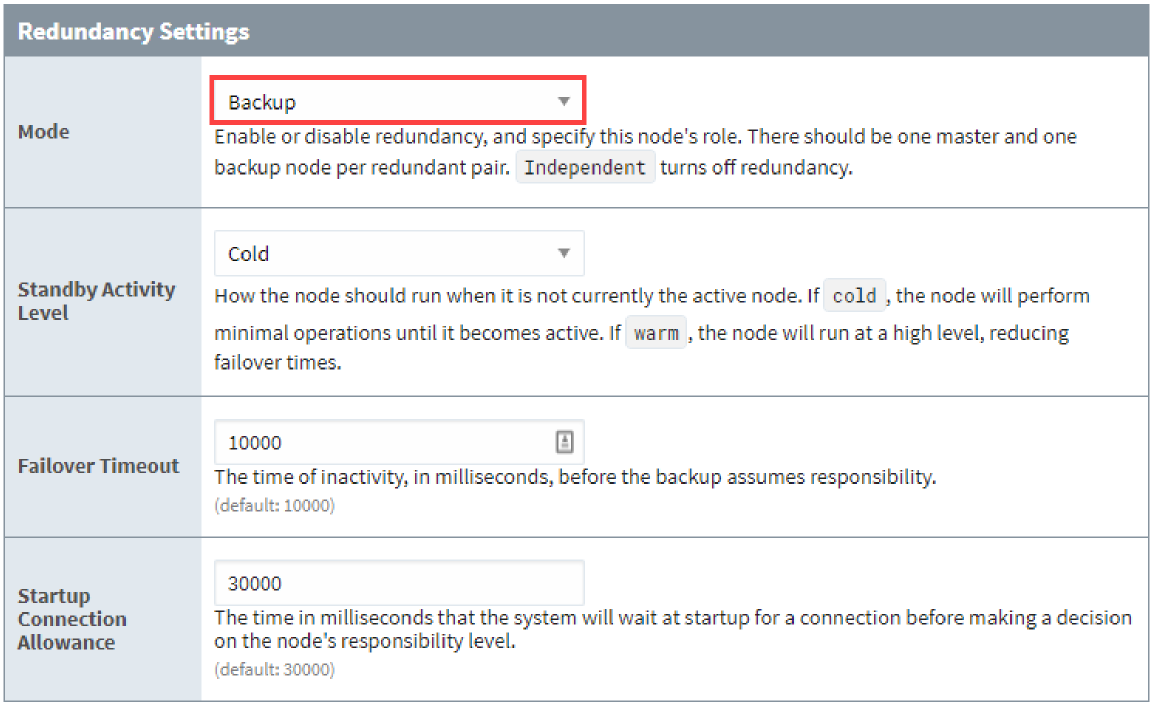
...

INDUCTIVE UNIVERSITY

Redundant Licensing

Watch the Video

If using a Backup only license, the Mode property on the Backup Gateway's Redundancy page will look like the image below.



Redundancy Settings

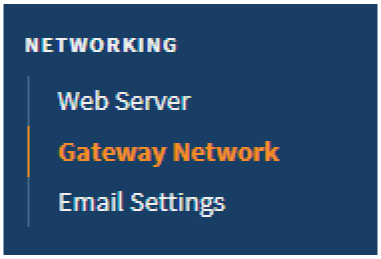
Mode	Backup Enable or disable redundancy, and specify this node's role. There should be one master and one backup node per redundant pair. `Independent` turns off redundancy.
Standby Activity Level	Cold How the node should run when it is not currently the active node. If `cold`, the node will perform minimal operations until it becomes active. If `warm`, the node will run at a high level, reducing failover times.
Failover Timeout	10000 The time of inactivity, in milliseconds, before the backup assumes responsibility. (default: 10000)
Startup Connection Allowance	30000 The time in milliseconds that the system will wait at startup for a connection before making a decision on the node's responsibility level. (default: 30000)

Instead of the dropdown list that normally appears on this property, the license forced this Gateway into a Backup mode.

If you would like to change the mode of this Gateway, the backup-only license must first be unactivated. Once unactivated, a new license will need to be applied, otherwise, the Gateway will operate in 2 hour trial mode.

Gateway Network

The Gateway Network allows you to connect multiple Gateways together over a wide area network, and opens up many distributed features between Gateways.



The Gateway Network provides the following features:

- A dedicated HTTP data channel that can handle multiple streams of message data.
- The ability to set up a node to act as a proxy for another node.
- Security settings that restrict incoming connections based on a white list or on manual approval of the connection. Incoming connections can also be disabled entirely.
- An available SSL mode. When enabled, connections must send SSL certificates to prove their identity. A connection will not be accepted until its SSL certificate is approved.

On this page ...

Gateway Network Features

The Gateway Network opens up certain services for use that make managing multiple Gateways and having them effectively communicate with each other a snap. It also has special security that can restrict certain services from happening in certain zones of the Gateway Network.

Enterprise Administration

The Enterprise Administration Module (EAM) uses the Gateway Network for message and file transfer, and can monitor network connections for availability. The EAM reports whenever a comm link is lost via alarm events and system Tags.

Distributed Services

Distributed services included the following:

- Remote Providers: Remote Realtime and Historical Tag providers make remotely controlling and storing Tag data even easier.
- Remote Alarming: Remote Alarming makes notifying all Gateways in the network possible, to quickly and effortlessly track down issues.

Security Zones and Service Security

Security Zones can be setup to lock down or prevent access to certain parts of Gateways within the Gateway Network.

Gateway Network Overview

Watch the Video

Outgoing vs. Incoming Connections

When using the Gateway Network, you will be working with two type of connections.

- **Outgoing Connections:** To establish communications, create an outgoing connection on the local machine. The outgoing connection always begins the connection process to a remote machine. After the outgoing connection is created, the local machine will attempt to use the connection to establish communications with the remote machine.
- **Incoming Connections:** On the remote machine, an incoming connection will automatically be created when the new connection attempt is detected. For connections where security settings require manual approval, you will need to approve the incoming connection before it can be used. If no security controls have been set, the incoming connection will automatically accept the connection from the local machine and begin sharing data.

Connections and Servers

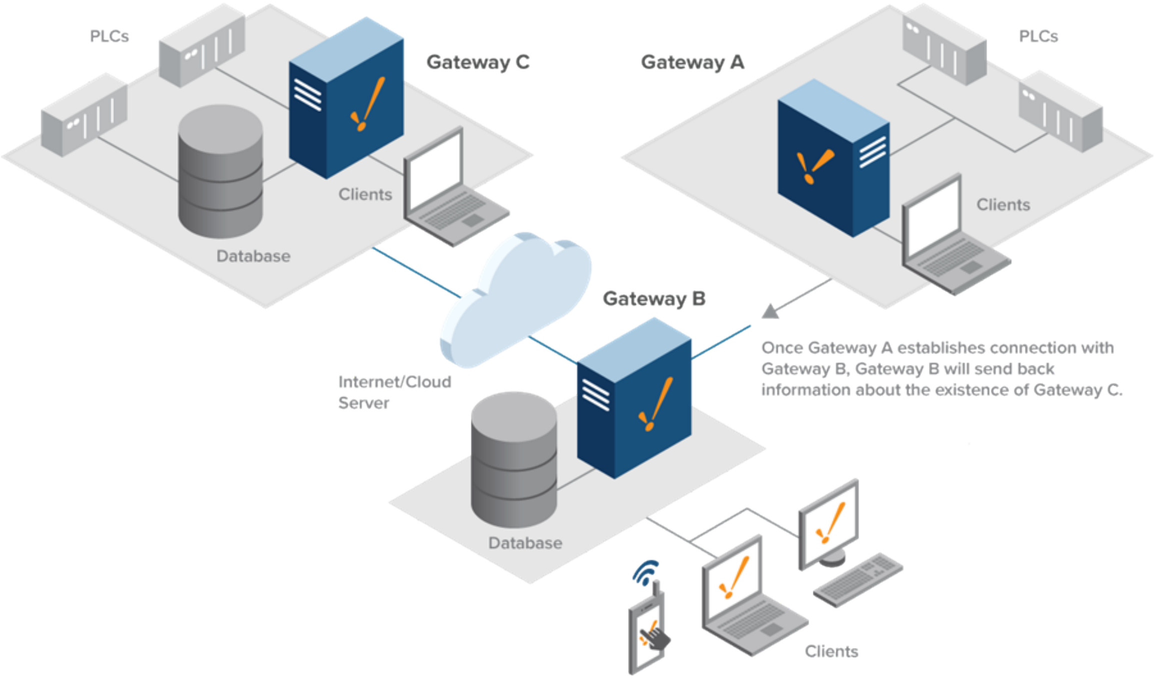
Every machine on the Gateway Network is known as a Server. When you establish a connection to a remote machine, the remote Server sends data about itself and also sends data about any other Servers known to that machine. For example, assume your local machine is GatewayA. The remote machine is known as GatewayB. GatewayB also knows about another remote machine named GatewayC. As soon as your local GatewayA establishes a connection with GatewayB, GatewayB also sends information about the existence of GatewayC.

Modules such as the Enterprise Administration Module (EAM) are aware of this relationship and allow communication between GatewayA and GatewayC, even though there is no direct connection from the local machine to GatewayC.

Which Server Should I Configure the Outgoing Connection On?

In regards to connecting multiple Gateways over the Gateway Network, there is little difference between an Outgoing and Incoming connection: these terms simply note which server the connection was configured on, and are mostly ignored by the rest of Ignition. Thus, assuming GatewayA and GatewayB, configuring an outgoing connection from **A** to **B** is equivalent to configuring an outgoing connection from **B** to **A**. However, it is important to note that when connecting two Gateways, only a single connection is required between them.

If you are cloning Gateways to then be connected via Gateway Network, it is important to note that there is a Gateway unique identifier in %IgnitionInstallationDirectory%/data/.uuid. No two Gateways connected via Gateway Network should share a .uuid. Generally, Gateways are cloned by restoring the same Gateway backup on multiple servers. Since Gateway backups carry their .uuid with them, restoring the same Gateway on multiple servers will result in multiple Gateways having the same .uuid. To get around this, you must stop your Ignition service, delete %IgnitionInstallationDirectory%/data/.uuid, then start your Ignition service so that a new, unique .uuid is generated. Doing this before connecting two cloned Gateways will prevent any .uuid collisions.


PLCs
Gateway C
Database
Clients
Gateway A
PLCs
Clients
Gateway B
Internet/Cloud
Server
Database
Clients
Once Gateway A establishes connection with
Gateway B, Gateway B will send back
information about the existence of Gateway C.

General Settings

Main

The Gateway Network General Settings set the basic rules for the system. By default, these settings are lenient to allow for easy setup, but can be set for security.

Setting	Description
Enabled	Uncheck this checkbox to disable using the Gateway Network.
Require SSL	If true, only connections that use SSL to encrypt traffic will be allowed. This setting only applies to incoming connections. Default is true.
Require Two Way Auth	Enforces two-way SSL authentication. If true, you will need to install the remote machine's certificate on this machine, in addition to manual approval of this machine's certificate on the remote machine. If you check this setting, you will need to provide the remote machine's certificate. To do this, manually export a certificate from the remote machine's metro KeyStore, located in <installdir>/webserver/metro-keystore. Default KeyStore password is **metro**, and the alias is **metro-key**. Then place the certificate on the local machine, in data/certificates/gateway_network.
Send Thread	The following feature is new in Ignition version **8.0.13** Click here to check out the other new features The maximum number of threads that will be used to upload messages. Applies to outgoing connections. Default is 5.
Send Buffer Limit	The number of outstanding messages that can be waiting for acknowledgement at a time. Default is 5.
Processing Queue Limit	The following feature is new in Ignition version **8.0.13** Click here to check out the other new features Number of received messages that can be held until they are processed by the local system. When this capacity is exceeded, new messages are rejected and errors are reported to the remote Gateway. Applies to incoming connections.
Receive Queue Limit	This feature was removed from Ignition in version 8.013 If there is too much traffic on the Gateway Network, then it can tell Agents to stop sending messages for a specified amount of time. This setting is the number of received messages that can be held until they are processed by the local system. When this capacity is exceeded, the calling side will wait until capacity is available. Default is 100.
Receive Threads	The maximum number of threads that will be used to download messages. Applies to outgoing connections. Default is 5.

Security

By default, the security level for incoming connections is set to “Unrestricted”, meaning that every remote machine that attempts to connect to the local machine will be accepted without question. You have several options to control security from the Gateway Network settings. Note that these settings are independent of SSL mode, which is detailed below. To change security settings, go to the Gateway Webpage and navigate to **Config -> Networking -> Gateway Network** and select the **General Settings** tab.

Setting	Description
Allow Incomi	

ng Connections	If false, only outward connections defined on this gateway will be allowed. Uncheck this checkbox to disable all remote machines from being able to establish an incoming connection. To establish any connections with remote machines, you will need to create outgoing connections from this machine. Default is true.
Connection Policy	Dictates what connections are allowed. Options as follows: • **Unrestricted** - Default mode, allows all incoming connections unless the Allow Incoming Connections checkbox is unchecked. • **ApprovedOnly** - Incoming connections are created, but cannot be used to send or receive data until you approve the connection under **Gateway Network -> Incoming Connections** tab. To approve an incoming connection, click the Approve link on the right side of the connection. You can also deny a previously approved connection by clicking the deny link. Note that the approve and deny links will appear next to a connection only if you have enabled the ApproveOnly setting. • **SpecifiedList** - An incoming connection will only be allowed if its server name is on this list. Separate server names with a comma.
Specified List	Connections with a Gateway Name in this list are automatically allowed if the security mode is set to SpecifiedList. Separate Gateway names with a comma.
Allow Proxying	If enabled, this Gateway will be allowed to act as a proxy, and forward requests between Gateways that do not have direct connections.

Setting Up a Gateway Network Connection

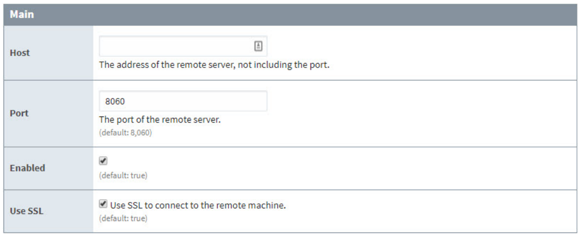
When you create a new outgoing Gateway Network connection, you need to specify the address for the remote server. There are also settings for ping rates and timeouts. The defaults can be used for these fields. The following are all the available settings for setting up an outgoing Gateway Network.

INDUCTIVE UNIVERSITY

Setting up a Gateway Network Connection

Watch the Video

Main



Main

Host
The address of the remote server, not including the port.

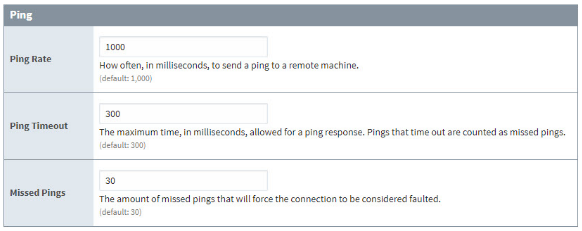
Port
8060
The port of the remote server.
(default: 8,060)

Enabled
☑
(default: true)

Use SSL
☑ Use SSL to connect to the remote machine.
(default: true)

Setting	Description
Host	The address of the remote server, not including the port. For example, http://10.10.123.4.
Port	The port of the remote server. Default is 8060.
Enabled	Whether this connection is enabled. Default is true.
Use SSL	Use SSL to connect to the remote machine. Default is true.

Ping



Ping

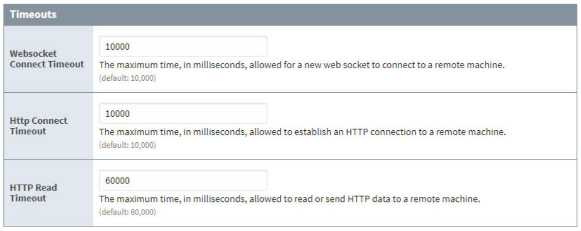
Ping Rate: 1000
How often, in milliseconds, to send a ping to a remote machine.
(default: 1,000)

Ping Timeout: 300
The maximum time, in milliseconds, allowed for a ping response. Pings that time out are counted as missed pings.
(default: 300)

Missed Pings: 30
The amount of missed pings that will force the connection to be considered faulted.
(default: 30)

Setting	Description
Ping Rate	How often, in milliseconds, to send a ping to a remote machine. Default is 1,000.
Ping Timeout	The maximum time, in milliseconds, allowed for a ping response. Pings that time out are counted as missed pings. Default is 300.
Missed Pings	The amount of missed pings that will force the connection to be considered faulted. Default is 30.

Timeouts



Timeouts

Websocket Connect Timeout: 10000
The maximum time, in milliseconds, allowed for a new web socket to connect to a remote machine.
(default: 10,000)

Http Connect Timeout: 10000
The maximum time, in milliseconds, allowed to establish an HTTP connection to a remote machine.
(default: 10,000)

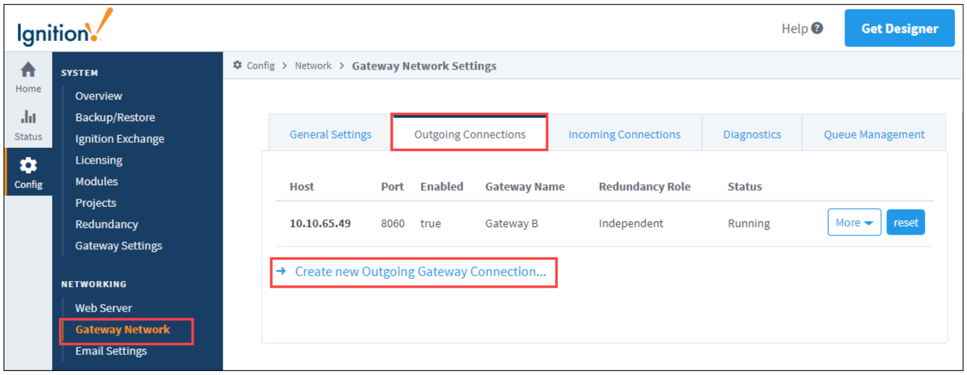
HTTP Read Timeout: 60000
The maximum time, in milliseconds, allowed to read or send HTTP data to a remote machine.
(default: 60,000)

Setting	Description
Websocket Connect Timeout	The maximum time, in milliseconds, allowed for a new web socket to connect to a remote machine. Default is 10,000.
HTTP Connect Timeout	The maximum time, in milliseconds, allowed to establish an HTTP connection to a remote machine. Default is 10,000.
HTTP Read Timeout	The maximum time, in milliseconds, allowed to read or send HTTP data to a remote machine. Default is 60,000.

Gateway Network Connection Example

To establish a basic communication link between two Gateways, first log into the Gateway where you want to establish the outgoing connection. For this example, we use an SSL connection.

1. On the Gateway Webpage, navigate to **Config -> Networking -> Gateway Network**.
2. Click on the **Outgoing Connections** tab. Click the **Create new Outgoing Gateway Connection** link.



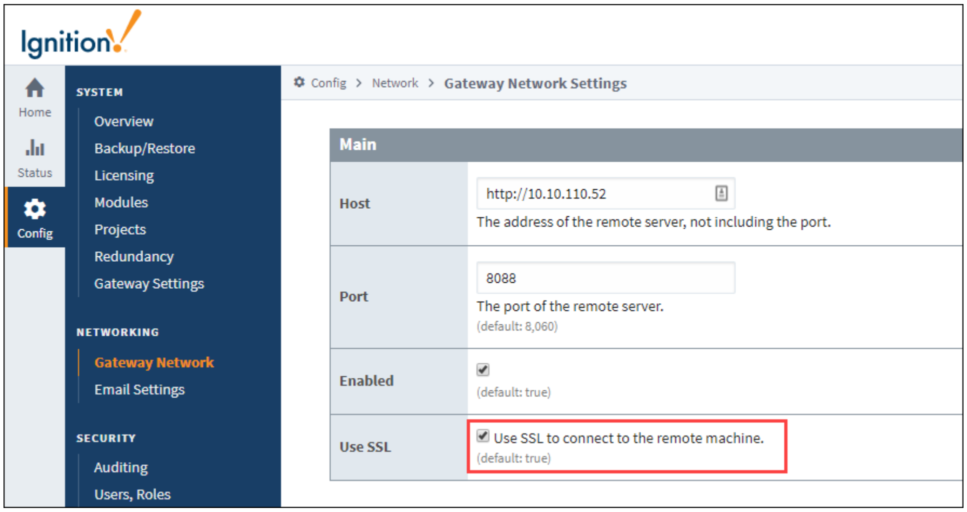
Host	Port	Enabled	Gateway Name	Redundancy Role	Status
10.10.65.49	8060	true	Gateway B	Independent	Running

3. In the **Host** field, enter the network address of the remote server.

4. In the **Port** field, enter the SSL port used by the remote server. By default, this is set to **8060** (which is defined /data/gateway.xml).

> ⚠ This port is different from the default SSL port an Ignition Gateway would use when communicating to a client (default port 8043).

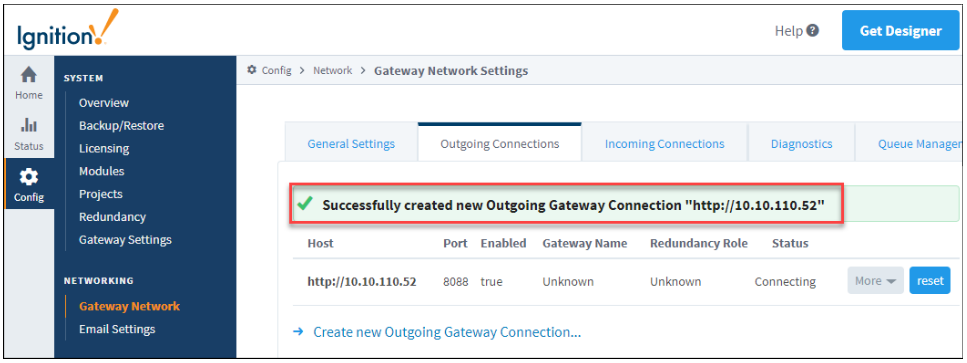
5. Check the **Use SSL** checkbox.



Main

Host	http://10.10.110.52 The address of the remote server, not including the port.
Port	8088 The port of the remote server. (default: 8,060)
Enabled	☑ (default: true)
Use SSL	☑ Use SSL to connect to the remote machine. (default: true)

6. Use the default settings in the **Ping** section and **Timeouts** section of the page.
7. Click the **Create New Outgoing Gateway Connection** button at the bottom of the page.
8. You'll see a confirmation message that the connection was created.

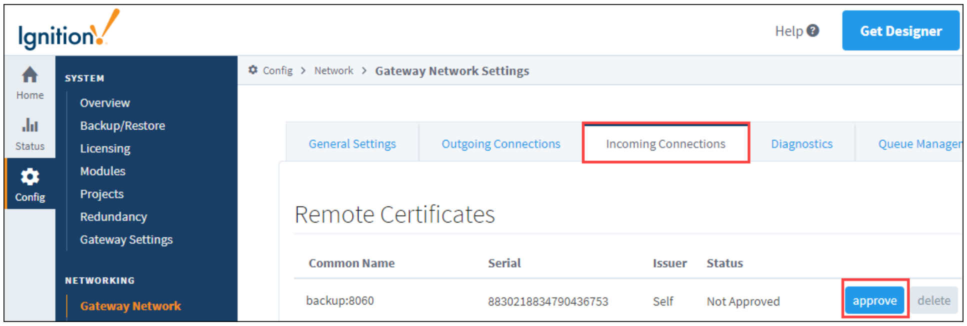


✓ Successfully created new Outgoing Gateway Connection "http://10.10.110.52"

Host	Port	Enabled	Gateway Name	Redundancy Role	Status		
http://10.10.110.52	8088	true	Unknown	Unknown	Connecting	More	reset

→ Create new Outgoing Gateway Connection...

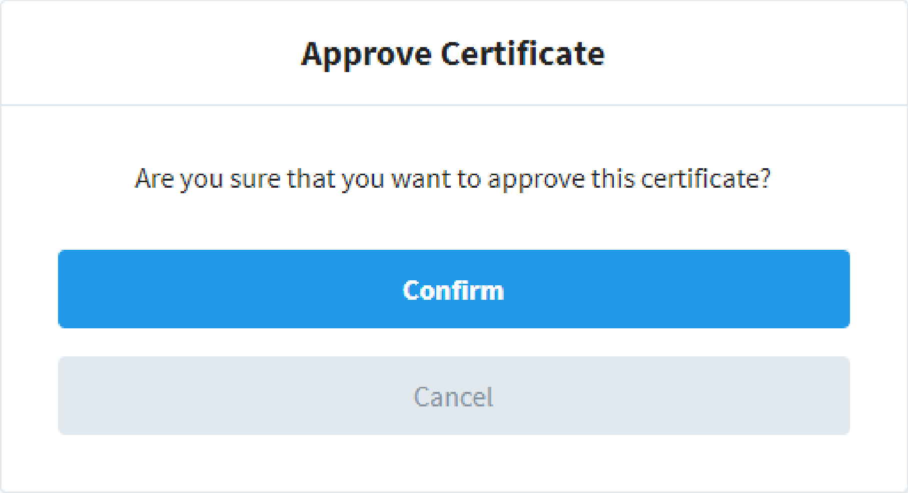
9. At this point, your Gateway transmitted its certificate to the connected Gateway, but the incoming connection is not yet allowed. The Gateway's connection will not show up under the Incoming Connections tab until after the certificate has been approved.
10. Log into the other Gateway. Navigate to **Config -> Networking -> Gateway Network**.
11. Click on the **Incoming Connections** tab. The first Gateway's certificate should be present. The certificate Common Name field holds the network address of the machine that transmitted the certificate. The Serial field holds a numeric string that is automatically generated when the certificate is created, and is unique to every certificate.



Remote Certificates

Common Name	Serial	Issuer	Status		
backup:8060	8830218834790436753	Self	Not Approved	approve	delete

12. Click the **approve** button to accept the certificate. You'll see a confirmation message. Click the **Confirm** button.



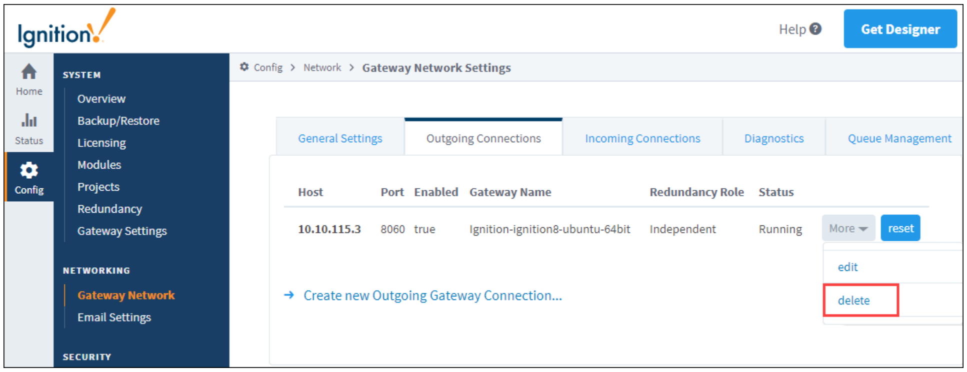
Approve Certificate

Are you sure that you want to approve this certificate?

Deleting Connections

Outgoing and incoming connections can be deleted for cases when the connection no longer exists on the other side.

1. To delete a connection, navigate to **Config -> Networking -> Gateway Network**.
2. Click on either the **Outgoing Connections** tab or the **Incoming Connections** tab.
3. Click **More**, and then select **Delete** next to the connection. Note that for incoming connections, if a remote machine is still connected to the local machine with an outgoing connection, a new incoming connection will be created after deletion. For these cases, you must log into the remote Gateway and delete the outgoing connection. Then you can delete the local incoming connection.



Host	Port	Enabled	Gateway Name	Redundancy Role	Status
10.10.115.3	8060	true	Ignition-ignition8-ubuntu-64bit	Independent	Running

Certificates and SSL

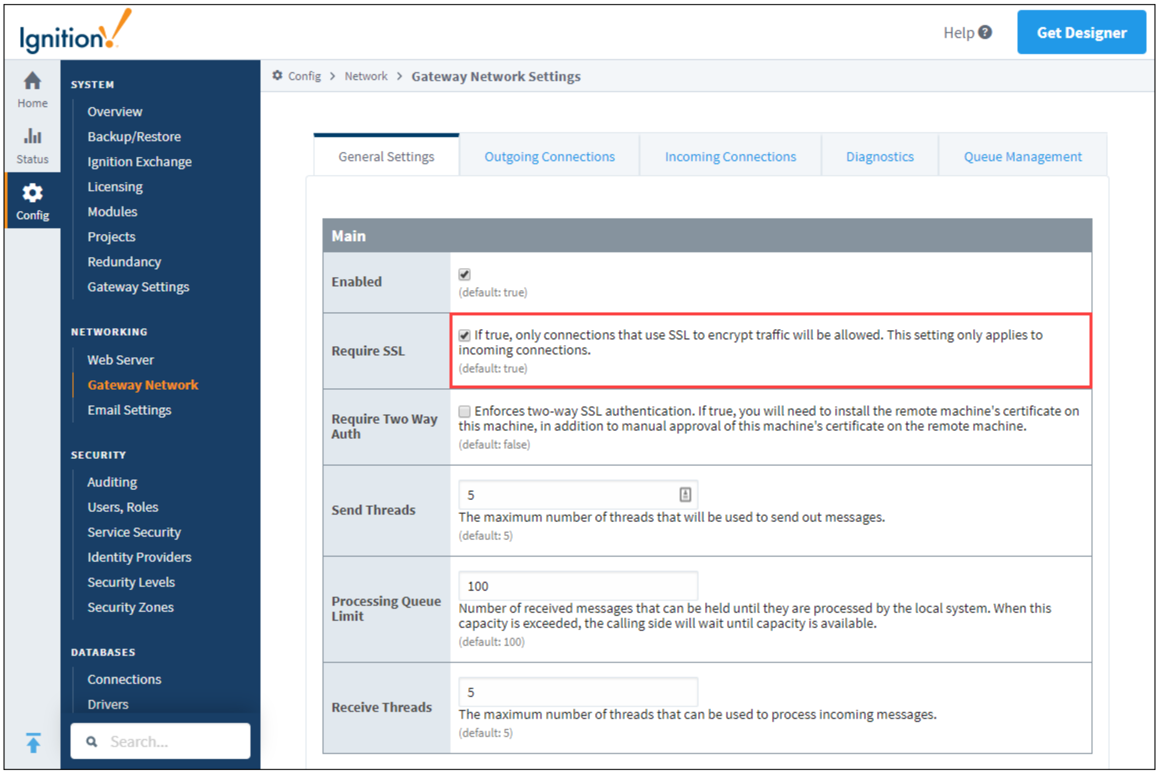
When a remote machine establishes an incoming connection, its Gateway server name is transmitted and appears in the Server Name field under **Gateway Network -> Incoming Connections**. However, no identity authentication is performed when the connection is created. The local system accepts the remote system id without question. To perform identity authentication on a connection, you must use Secure Socket Layer (SSL) and certificates. By default, SSL is enabled.

⚠ When using the Gateway Network and Redundancy, SSL Certificates are automatically pushed from the redundant Master to the Backup.

Requiring a Certificate

1. To require all incoming Gateways to use SSL, navigate to **Config -> Networking -> Gateway Network**.

2. Select the **General Settings** tab, and check the **Require SSL** checkbox.
3. Click the **Save Changes** button.



Main

Enabled	☑ (default: true)
Require SSL	☑ If true, only connections that use SSL to encrypt traffic will be allowed. This setting only applies to incoming connections. (default: true)
Require Two Way Auth	□ Enforces two-way SSL authentication. If true, you will need to install the remote machine's certificate on this machine, in addition to manual approval of this machine's certificate on the remote machine. (default: false)
Send Threads	5 The maximum number of threads that will be used to send out messages. (default: 5)
Processing Queue Limit	100 Number of received messages that can be held until they are processed by the local system. When this capacity is exceeded, the calling side will wait until capacity is available. (default: 100)
Receive Threads	5 The maximum number of threads that can be used to process incoming messages. (default: 5)

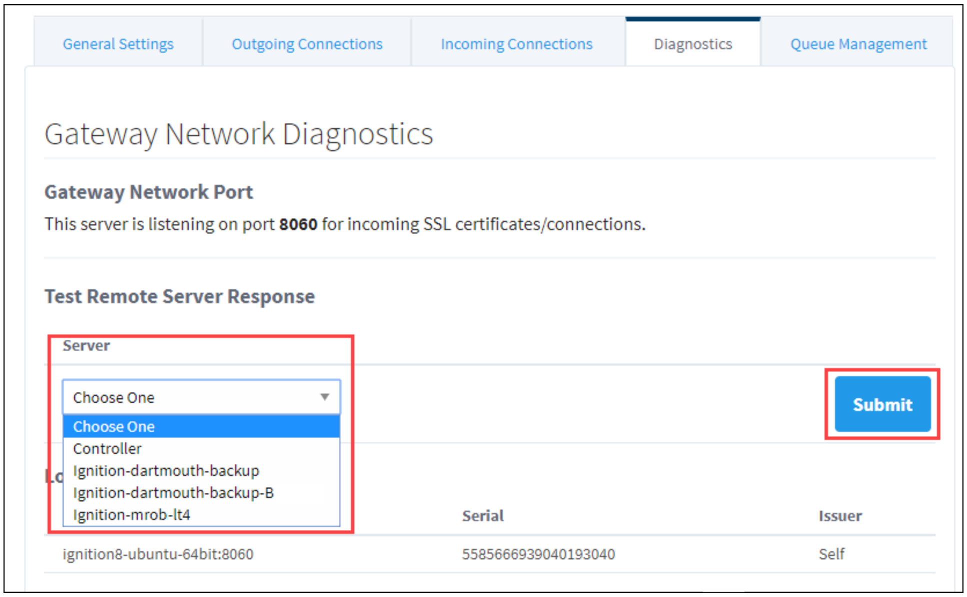
Denying a Certificate

You can deny a certificate under the Certificates tab by clicking the deny link to the right of the certificate. The connection that has been using that certificate will no longer be allowed to connect. You can delete certificates that are no longer in use. Keep in mind that if you delete a certificate, and a remote machine is still using that certificate, it will reappear on the Certificates page. In this case, you must navigate to the remote Gateway and delete its outgoing connection. Then you can permanently delete the certificate from the Certificates page.

Gateway Network Diagnostics

The Diagnostics tab on the Gateway Network Settings page gives you insight to the Gateway and and remote server response times.

1. To test the response time of a remote server, select the server name from the **Server** dropdown list.
2. Click the **Submit** button.



Gateway Network Diagnostics

Gateway Network Port

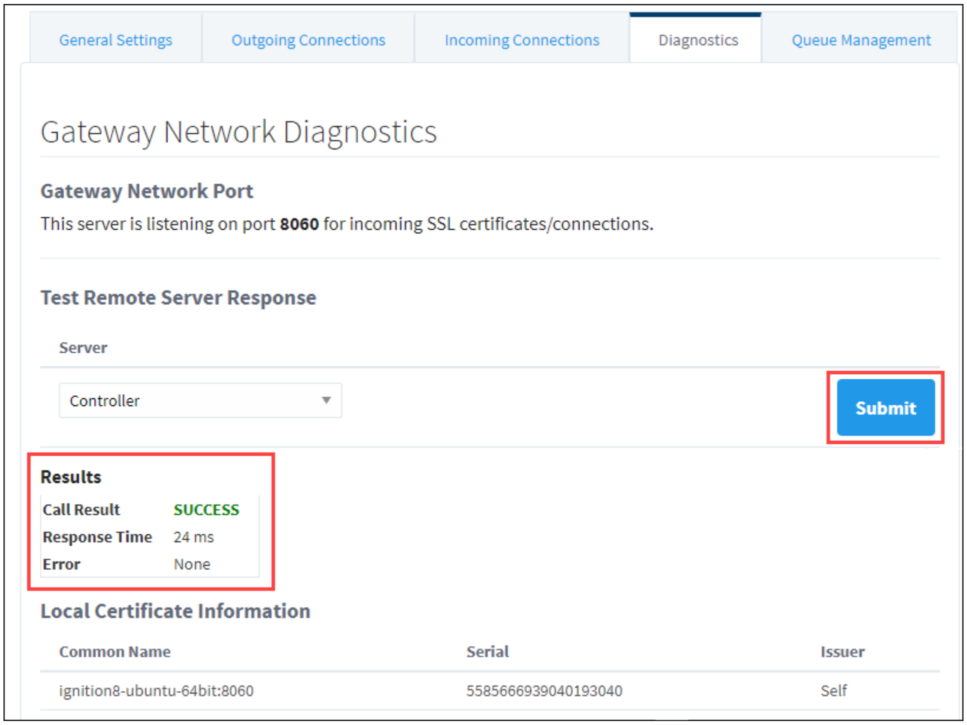
This server is listening on port **8060** for incoming SSL certificates/connections.

Test Remote Server Response

Server: Choose One

	Serial	Issuer
ignition8-ubuntu-64bit:8060	5585666939040193040	Self

3. The results will be displayed indicating if the call to the remote server was successful, what the response time was, and if there were any errors.



Gateway Network Diagnostics

Gateway Network Port

This server is listening on port **8060** for incoming SSL certificates/connections.

Test Remote Server Response

Server: Controller

Results

Call Result	SUCCESS
Response Time	24 ms
Error	None

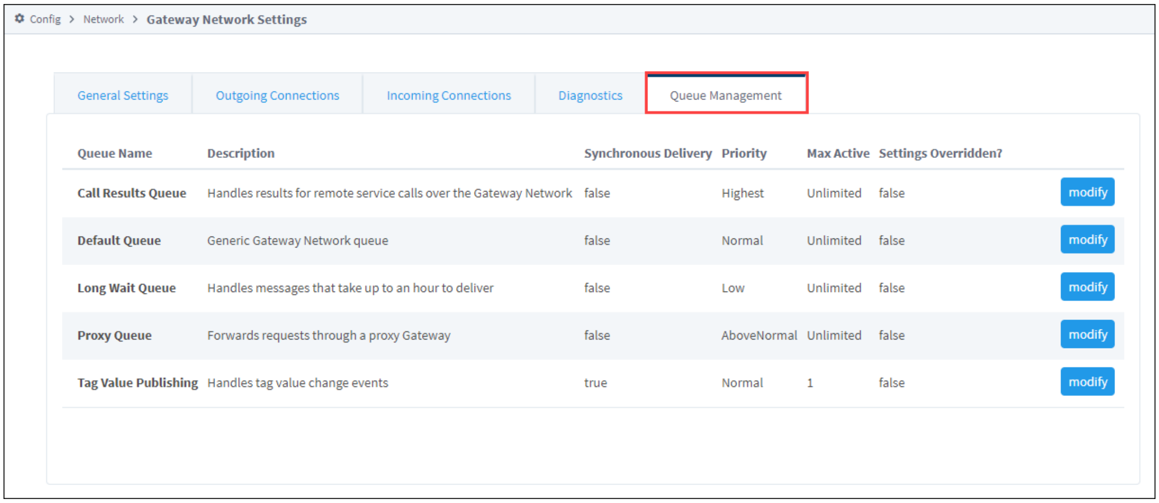
Local Certificate Information

Common Name	Serial	Issuer
ignition8-ubuntu-64bit:8060	5585666939040193040	Self

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

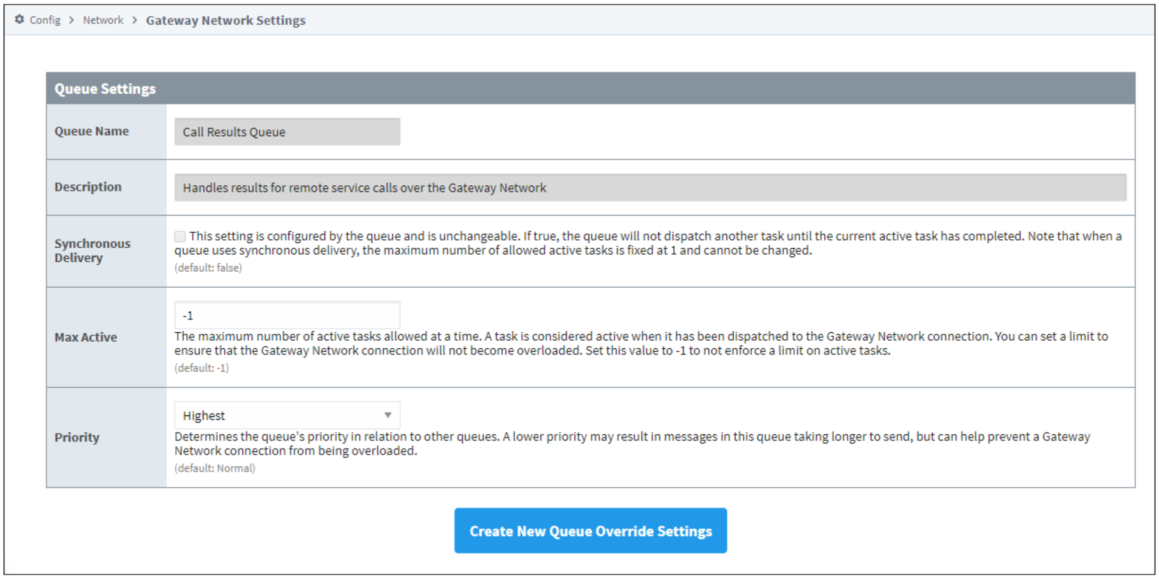
Gateway Network Queue Management

Ignition's Gateway Network system shares information across Gateways using a configurable number of send and receive threads. Ignition's Gateway Network also has a queue associated with each Ignition sub system. These queues enable Ignition to prioritize which subsystem should have access to a send or receive thread at any given time. For example, there are two Gateways, Gateway A and Gateway B, connected via Gateway Network. Gateway A is sending a lot of Tag History queries to Gateway B per the Outgoing Tasks list of Gateway A. If Gateway B takes a long time to return a query result for each query request, Gateway B could potentially starve the send or receive threads for the connection. Starving the send or receive threads for the connection could potentially affect other Ignition sub-systems aside from Tag History. A solution is to limit the Max Active setting on the Call Results Queue configuration in Gateway A to 3. This will make sure that no more than 3 send or receive threads are used for the Tag History requests coming from Gateway A. Doing this will slow down the Tag History requests and therefore the Tag History queries but it will allow for other Ignition sub systems to gracefully send and receive messages without interruption. The Queue Management tab allows uses to manage how a queue should behave for a specific Ignition subsystem. The queue settings for each type of outgoing queue are displayed on this page along with each queue's description.



Queue Name	Description	Synchronous Delivery	Priority	Max Active	Settings Overridden?	
Call Results Queue	Handles results for remote service calls over the Gateway Network	false	Highest	Unlimited	false	modify
Default Queue	Generic Gateway Network queue	false	Normal	Unlimited	false	modify
Long Wait Queue	Handles messages that take up to an hour to deliver	false	Low	Unlimited	false	modify
Proxy Queue	Forwards requests through a proxy Gateway	false	AboveNormal	Unlimited	false	modify
Tag Value Publishing	Handles tag value change events	true	Normal	1	false	modify

Clicking the Modify button for one of these queues will bring up the Queue Settings page as below:



Queue Settings

Queue Name: Call Results Queue

Description: Handles results for remote service calls over the Gateway Network

Synchronous Delivery: ☐ This setting is configured by the queue and is unchangeable. If true, the queue will not dispatch another task until the current active task has completed. Note that when a queue uses synchronous delivery, the maximum number of allowed active tasks is fixed at 1 and cannot be changed.
(default: false)

Max Active: -1
The maximum number of active tasks allowed at a time. A task is considered active when it has been dispatched to the Gateway Network connection. You can set a limit to ensure that the Gateway Network connection will not become overloaded. Set this value to -1 to not enforce a limit on active tasks.
(default: -1)

Priority: Highest
Determines the queue's priority in relation to other queues. A lower priority may result in messages in this queue taking longer to send, but can help prevent a Gateway Network connection from being overloaded.
(default: Normal)

Queue Settings

Settings	Description
Queue Name	Name of the queue you are modifying (read only).
Description	Description for the queue you are modifying (read only).
Synchronous Delivery	This setting is configured by the queue and is unchangeable. If true, the queue will not dispatch another task until the current active task has completed. Note that when a queue uses synchronous delivery, the maximum number of allowed active tasks is fixed at 1 and cannot be changed. Default is false. Note

	Some queues are hard-coded as “Synchronous Delivery” queues, for example the Tag Value Update queue. For these queues, the Max Active setting is fixed at 1 and cannot be changed by the user. The user can only change the priority of the queue.
Max Active	The maximum number of active tasks allowed at a time. A task is considered active when it has been dispatched to the Gateway Network connection. You can set a limit to ensure that the Gateway Network connection will not become overloaded. Set this value to -1 to not enforce a limit on active tasks. Default is -1.
Priority	Determines the queue's priority in relation to other queues. A lower priority may result in messages in this queue taking longer to send, but can help prevent a Gateway Network connection from being overloaded.

Database Connections

How Are Databases Used in Ignition?

While connecting to a database is not required for basic status and control functionality, it can dramatically increase the possibilities that the system offers. There are a few places where databases are used in Ignition, such as historical data logging, reporting, storing alarm logs, and as your Tags storage.

Historical Data Logging

Logging data for historical analysis, either through Tags Historian or with the SQL Bridge module, r equires a database connection. Databases are great at handling historical data, and by using a standard relational database your data is stored in an open format that can be used in many ways.

Reports, Graphs, and Charts

The Vision module makes it easy to present data stored in databases in a variety of ways. You can quickly create charts that show performance over time, locate anomalies, and detect trends. Furthermore, it's important to remember that it is possible to pull data from any database that Ignition is connected to, even if the data wasn't placed there by Ignition. This means you can tie in data from other sources or areas of your company, such as pulling in inventory and staff information, as well.

Storing Alarm Logs

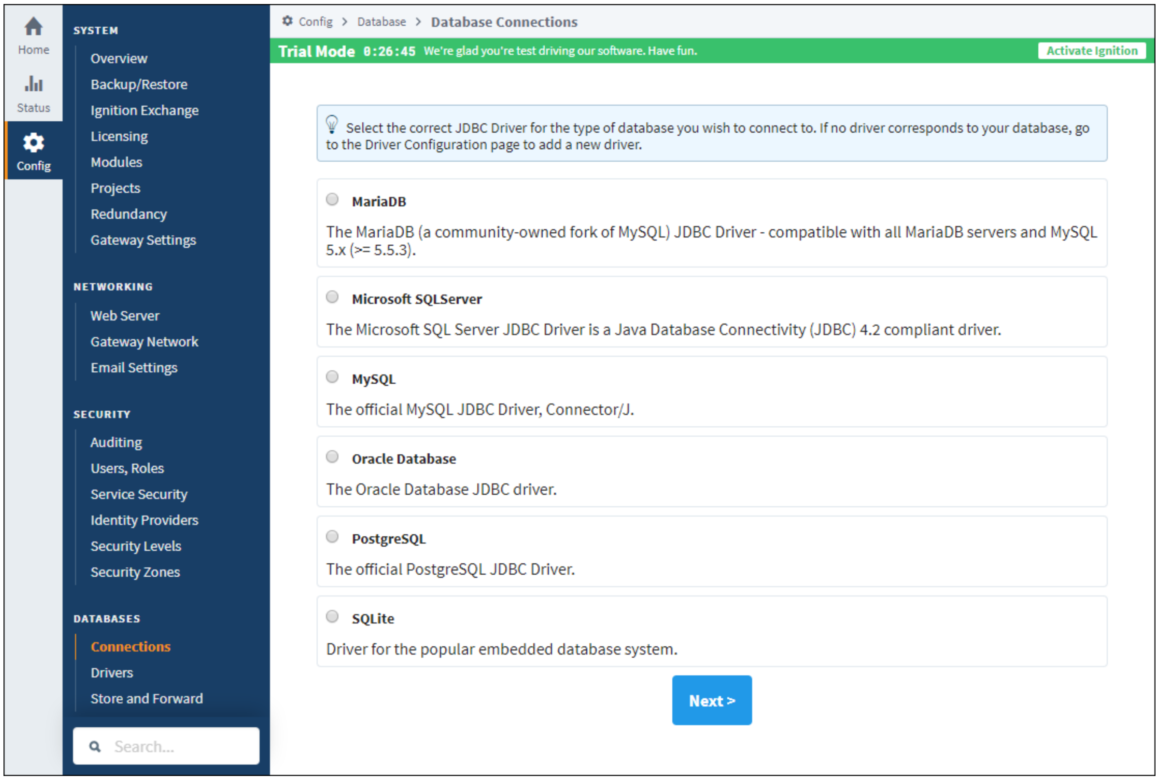
Store alarm information historically and examine it later for patterns or trouble spots.

On this page ...

Getting Started with Databases

The first step in using a database with Ignition is to identify a database server. Many companies already have database servers maintained by their IT departments. If you do not, or wish to set up your own database server for Ignition, the Supported Databases section below offers some advice on choosing a database vendor.

Once you've identified a server, all you need to do is create a connection to that server to get up and running.



Select the correct JDBC Driver for the type of database you wish to connect to. If no driver corresponds to your database, go to the Driver Configuration page to add a new driver.

○ **MariaDB**

The MariaDB (a community-owned fork of MySQL) JDBC Driver - compatible with all MariaDB servers and MySQL 5.x (>= 5.5.3).

○ **Microsoft SQLServer**

The Microsoft SQL Server JDBC Driver is a Java Database Connectivity (JDBC) 4.2 compliant driver.

○ **MySQL**

The official MySQL JDBC Driver, Connector/J.

○ **Oracle Database**

The Oracle Database JDBC driver.

○ **PostgreSQL**

The official PostgreSQL JDBC Driver.

○ **SQLite**

Driver for the popular embedded database system.

Supported Databases in Ignition

Ignition has been tested with the following databases, and can connect to them directly after installation. You can connect to other databases by installing additional JDBC drivers (the Java database connection specification), which are often provided by database vendors.

Database	Version
Full Support	
MySQL	5.0+ for full support. Ignition can connect to 4.x, but many features such as Tags are not tested.
Microsoft SQL Server	2005, 2008, 2012, 2014, 2016, 2017 (full and express editions). Ignition can connect to 2000, but has not been fully tested.
Oracle	10g, 11g, 12c (full and express). The letters stand for "grid" and "cloud"
PostgreSQL	8.0+
Firebird	All versions.
IBM DB2	9.5+
SQLite	The following feature is new in Ignition version **8.0.11** Click here to check out the other new features Added a driver for the popular embedded database system. This can be used to connect to an existing SQLite database, or create a new database: setting the connect URL property to a file that doesn't exist will result in the driver attempting to create the database.
Limited support	
Other JDBC drivers	Due to variances in databases, some features may not work fully through other non-tested JDBC drivers. However, it is usually possible to get full functionality though the careful use of the database translator feature. For example, the JDBC driver for MariaDB could be downloaded and added to Ignition.

Installing and Connecting to a Database

Once you've identified a server, all you need to do is create a connection to that server to get up and running. See the Installing Databases and Connecting to Databases sections for details about how to install and connect to different databases through Ignition.

If we don't already have a connector for your database type, you can simply add it in yourself.

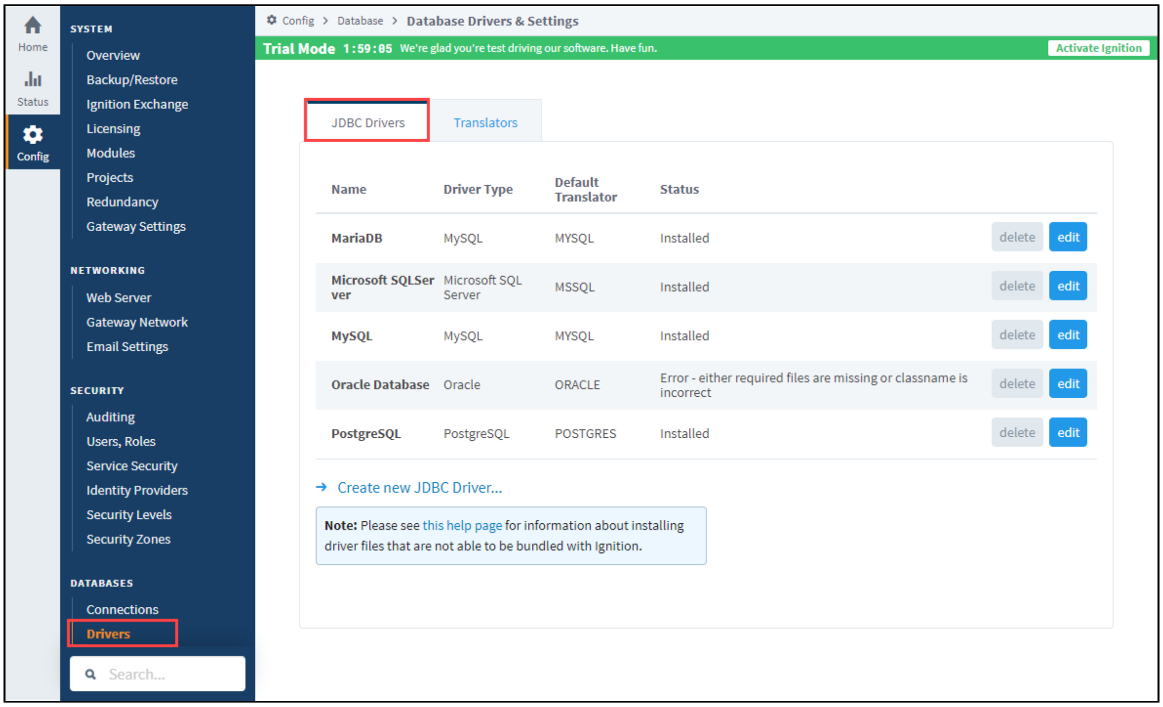
Database Drivers and Translators

What Is JDBC?

JDBC stands for the Java DataBase Connectivity API. It is a standardized way for Java-based applications to interact with a wide range of databases and data sources. A JDBC Driver enables Ignition to connect to, and use data from, a particular database system.

JDBC in Ignition

Ignition, being a Java-based application, leverages JDBC in order to connect to a variety of data sources. This enables Ignition to offer a standardized set of functionality on a wide range of different systems and databases. This includes not only commonly-used databases such as MySQL, Microsoft SQL Server, and Oracle, but additionally other lesser-known systems as well, provided the manufacturer offers a JDBC driver for the system.



Trial Mode 1:59:05 We're glad you're test driving our software. Have fun.

Name	Driver Type	Default Translator	Status
MariaDB	MySQL	MYSQL	Installed
Microsoft SQLServer	Microsoft SQL Server	MSSQL	Installed
MySQL	MySQL	MYSQL	Installed
Oracle Database	Oracle	ORACLE	Error - either required files are missing or classname is incorrect
PostgreSQL	PostgreSQL	POSTGRES	Installed

Note: Please see this help page for information about installing driver files that are not able to be bundled with Ignition.

JDBC vs. ODBC

JDBC differs from ODBC (Microsoft's OpenDataBase Connectivity standard) primarily in the fact that JDBC is written in Java, and thus can be used without modification in cross-platform environments. Additionally, whereas ODBC is a complex standard that is becoming technically out-dated, JDBC is a modern, clean specification for cross-vendor database access.

Can I Connect Using ODBC?

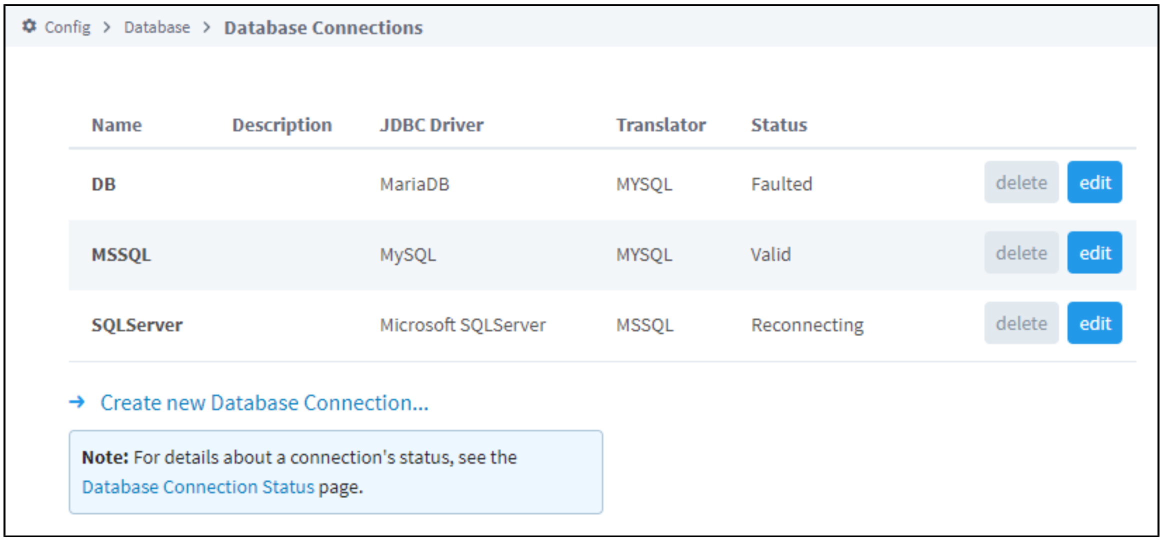
While it is indeed possible to connect to an ODBC data source using the JDBC-ODBC bridge, this is generally not advised. The bridge is designed to offer a minimal amount of functionality, and is considered a "transitional solution", meaning that it should only be used when JDBC is not available. In other words, if a JDBC option is available, ODBC should not be used.

Since most commercial databases offer JDBC drivers, transition is usually as simple as recreating your database connections inside of Ignition. The lack of a JDBC connection inside of Ignition does not necessarily indicate that JDBC isn't available for your particular database. Licensing restrictions sometime prevent the inclusion of drivers with 3rd-party software. Therefore, before using ODBC, due diligence should be taken to verify that no JDBC solution is available.

Monitoring Connection Status

The state or status of a database can be monitored from the **Status** section of the Gateway Webpage, under **Connections > Databases**. Th e status panels show the current state and a fault message, if applicable, or throughput statistics if the connection is active.

When a connection is not available, it is re-tested every 10 seconds, and the status is updated.



Config > Database > Database Connections

Name	Description	JDBC Driver	Translator	Status	
DB		MariaDB	MYSQL	Faulted	delete edit
MSSQL		MySQL	MYSQL	Valid	delete edit
SQLServer		Microsoft SQLServer	MSSQL	Reconnecting	delete edit

→ Create new Database Connection...

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

Related Topics ...

- SQL in Ignition
- JDBC Drivers and Translators

In This Section ...

Installing Databases

Why Install a Database?

On this page

...

A lot of additional functionality becomes simple or is only accessible when Ignition is connected to a database. Storing Historical data, storing notes or files, and creating dynamic lists to name a few. It is important to note that Ignition does not install any databases for you. There are many types that you can connect to, but you need to choose the database that is best for you. Installing your own database means you have complete control over it, anything Ignition adds to it can be accessed by another program easily.

You can install as many database systems as you like, and each of them allow your to create as many schemas (or groups of data) as you want. You get to decide where your database is installed, or where you want to install all of them.

Which Database Should You Use?

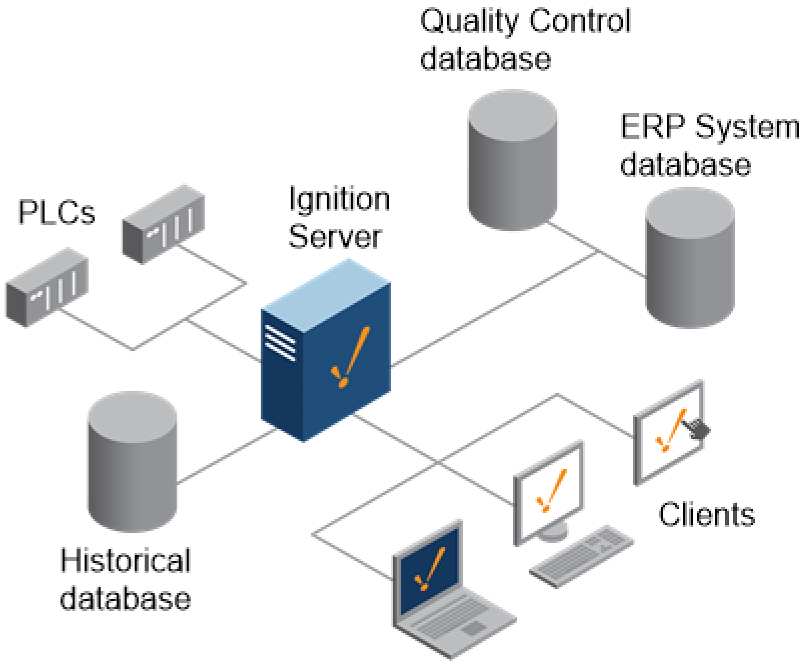
There is a lot of debate on this subject but the short answer to this question is 'whichever database your IT department already uses.' Modern relational database all have the same basic functionality, but slightly different ways of doing things. If your IT department already supports Microsoft SQL Server (MSSQL), then they already have the tools and knowledge to manage those databases. Because they are so similar (and Ignition takes care of so much for you), it is almost always easier to learn to use an existing database than to add IT support for a second type. Not to mention that adding a second type might mean hiring new personnel in the IT department.

If your company does not already have a database preference, then it's up to you to decide which is best based on your needs. They all have a free version, but different limitations. For example: MSSQL has a cap on how much data can be stored in their free version. MySQL does not have a cap like this, but also does not allow phone support for their free version.

Where to Install a Database

You can install your database anywhere that Ignition has access to through the network. There are two main options for installing your database: one is on the computer that Ignition is installed on, and the other is installed on a different server in your network. Technically, there is a third option to connect Ignition to a database that is in a remote location using a VPN or some other way to access it. This third option will work, but because of latency and the data being physically very far away, it is not recommended for storing data that will be accessed often like Tag History.

For production systems, we recommend that your database is on its own server, not installed on the computer with Ignition. This is helpful for many reasons, but mostly because databases can potentially take up a lot of resources on a computer. If the database is on its own computer, you don't have to worry about other programs starving for memory or CPU. If you do this and install your database on another computer, just make sure to adjust your firewalls and pay attention to the database connection security. Most databases don't allow the default username to connect remotely.



In This Section ...

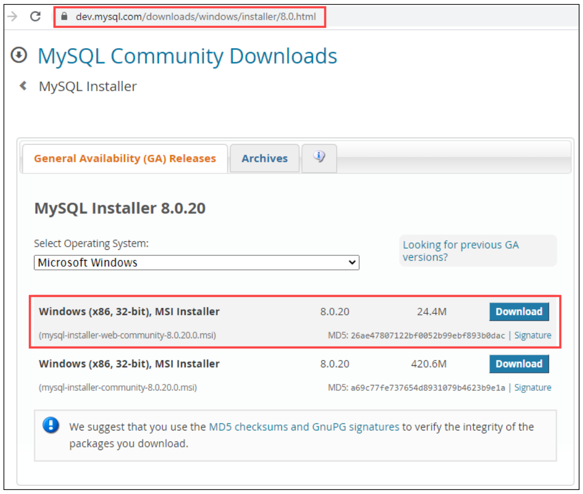
Installing MySQL

Install MySQL Server and MySQL Workbench

The goal of this page is to demonstrate how to install MySQL Server, and a helpful tool called MySQL workbench. This guide is not an exhaustive listing of all of the various installation steps or scenarios for MySQL. For more information, take a look at MySQL's documentation: MySQL Documentation.

1. Go to the MySQL website at https://dev.mysql.com/downloads/mysql/
2. Scroll-down to **Windows (x86, 32-bit), MSI Installer.** You will notice multiple download options. Both allow you to install MySQL

> ⚠ MySQL Installer is 32-bit, but will allow you to install the 64-bit version of MySQL.



dev.mysql.com/downloads/windows/installer/8.0.html

MySQL Community Downloads

MySQL Installer

General Availability (GA) Releases | Archives

MySQL Installer 8.0.20

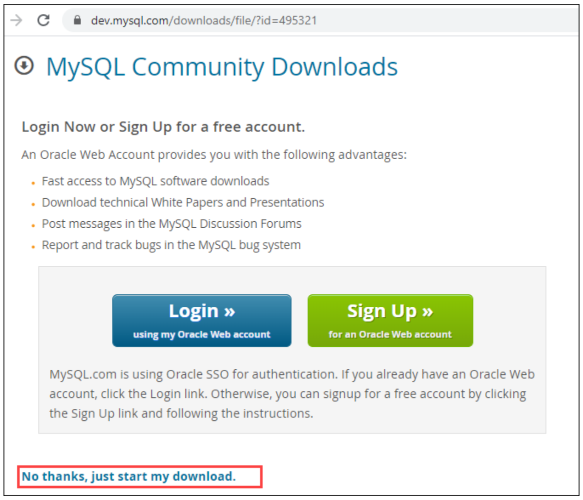
Select Operating System:
Microsoft Windows

Looking for previous GA versions?

Windows (x86, 32-bit), MSI Installer (mysql-installer-web-community-8.0.20.0.msi)	8.0.20	24.4M	Download MD5: 26ae47807122bf0052b99ebf893b0dac | Signature
Windows (x86, 32-bit), MSI Installer (mysql-installer-community-8.0.20.0.msi)	8.0.20	420.6M	Download MD5: a69c77fe737654d8931079b4623b9e1a | Signature

We suggest that you use the MD5 checksums and GnuPG signatures to verify the integrity of the packages you download.

3. Click on the **Download** button.
4. On the next page, you can login or create an account if you'd like. Otherwise click **No thanks, just start my download**.



dev.mysql.com/downloads/file/?id=495321

MySQL Community Downloads

Login Now or Sign Up for a free account.

An Oracle Web Account provides you with the following advantages:

- Fast access to MySQL software downloads
- Download technical White Papers and Presentations
- Post messages in the MySQL Discussion Forums
- Report and track bugs in the MySQL bug system

Login » using my Oracle Web account

Sign Up » for an Oracle Web account

MySQL.com is using Oracle SSO for authentication. If you already have an Oracle Web account, click the Login link. Otherwise, you can signup for a free account by clicking the Sign Up link and following the instructions.

No thanks, just start my download.

Running the Installer

On this page

...

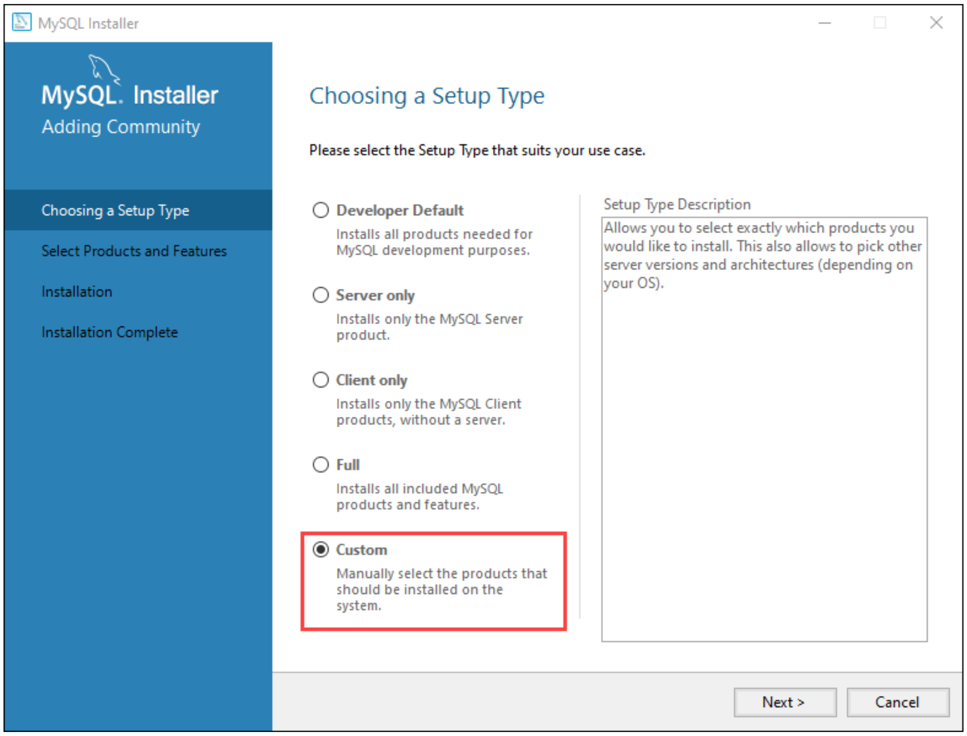
INDUCTIVE UNIVERSITY

Installing MySQL

Watch the Video

1. Once the **.msi** file is downloaded, run the file to begin the installation process.
2. The **Welcome** window is displayed. Select the **Install MySQL Products** action.
3. On the Choosing a Setup Type page, select **Custom** and click **Next**. While you can select one of the other options, at minimum you'll want to install both the server (the actual database) and MySQL Workbench (an application that allows you to quickly and easily interact with the database, without using a command-line client). Any other items beyond these two are generally unnecessary in most environments.

   If you want to learn more, MySQl does have some additional information on their documentation if you're curious about the other options: MySQL Docs. This example will continue with a Custom installation.



MySQL Installer

MySQL. Installer
Adding Community

- Choosing a Setup Type
- Select Products and Features
- Installation
- Installation Complete

Choosing a Setup Type

Please select the Setup Type that suits your use case.

○ **Developer Default**
Installs all products needed for MySQL development purposes.

○ **Server only**
Installs only the MySQL Server product.

○ **Client only**
Installs only the MySQL Client products, without a server.

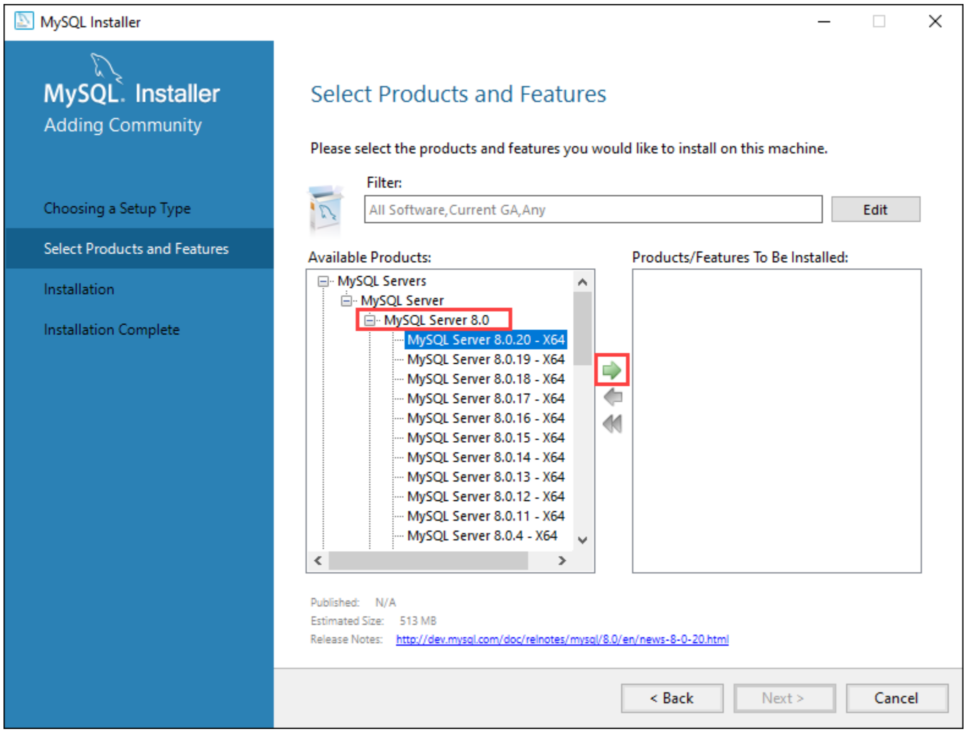
○ **Full**
Installs all included MySQL products and features.

● **Custom**
Manually select the products that should be installed on the system.

Setup Type Description
Allows you to select exactly which products you would like to install. This also allows to pick other server versions and architectures (depending on your OS).

Next > | Cancel

4. On the Select Products and Features page, scroll down to choose **MySQL Server 8.0**. Click the right arrow to move it to the "Products/Features To Be Installed" column.



MySQL Installer

MySQL. Installer
Adding Community

- Choosing a Setup Type
- Select Products and Features
- Installation
- Installation Complete

Select Products and Features

Please select the products and features you would like to install on this machine.

Filter:
All Software,Current GA,Any | Edit

Available Products:

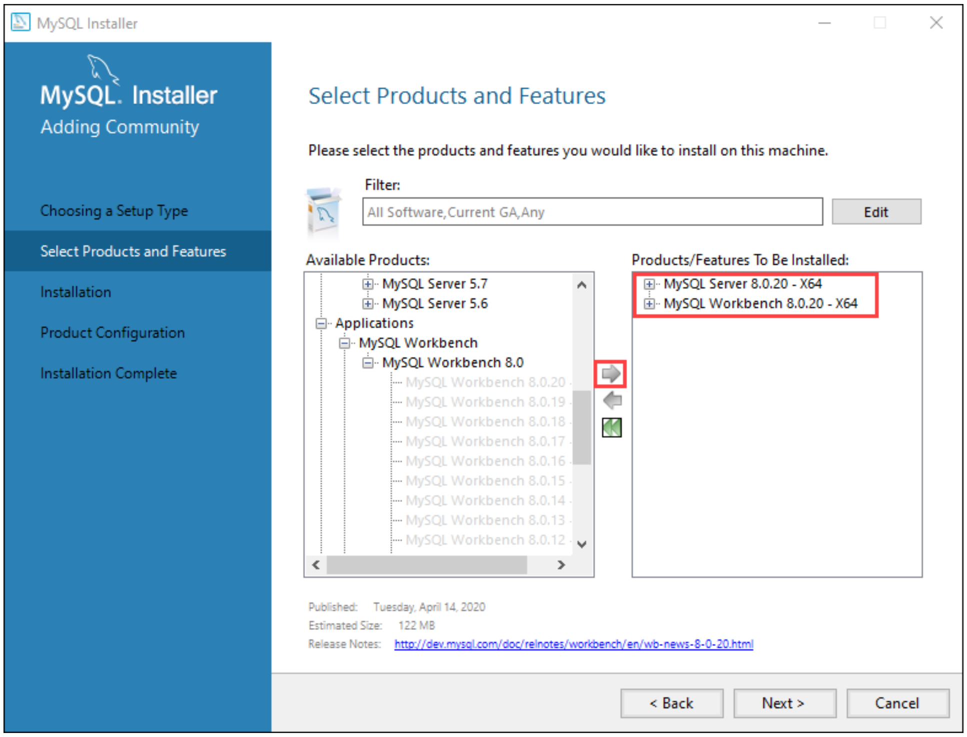
```
⊟ MySQL Servers
   ⊟ MySQL Server
      ⊟ MySQL Server 8.0
           MySQL Server 8.0.20 - X64
           MySQL Server 8.0.19 - X64
           MySQL Server 8.0.18 - X64
           MySQL Server 8.0.17 - X64
           MySQL Server 8.0.16 - X64
           MySQL Server 8.0.15 - X64
           MySQL Server 8.0.14 - X64
           MySQL Server 8.0.13 - X64
           MySQL Server 8.0.12 - X64
           MySQL Server 8.0.11 - X64
           MySQL Server 8.0.4 - X64
```

Products/Features To Be Installed:

Published: N/A
Estimated Size: 513 MB
Release Notes: http://dev.mysql.com/doc/relnotes/mysql/8.0/en/news-8-0-20.html

< Back | Next > | Cancel

5. Scroll down to MySQL Workbench, select a version, and click the right arrow to move it to the "Products/Features To Be Installed" column.
6. Click **Next**.



Select Products and Features

Please select the products and features you would like to install on this machine.

Filter: All Software,Current GA,Any

Available Products:

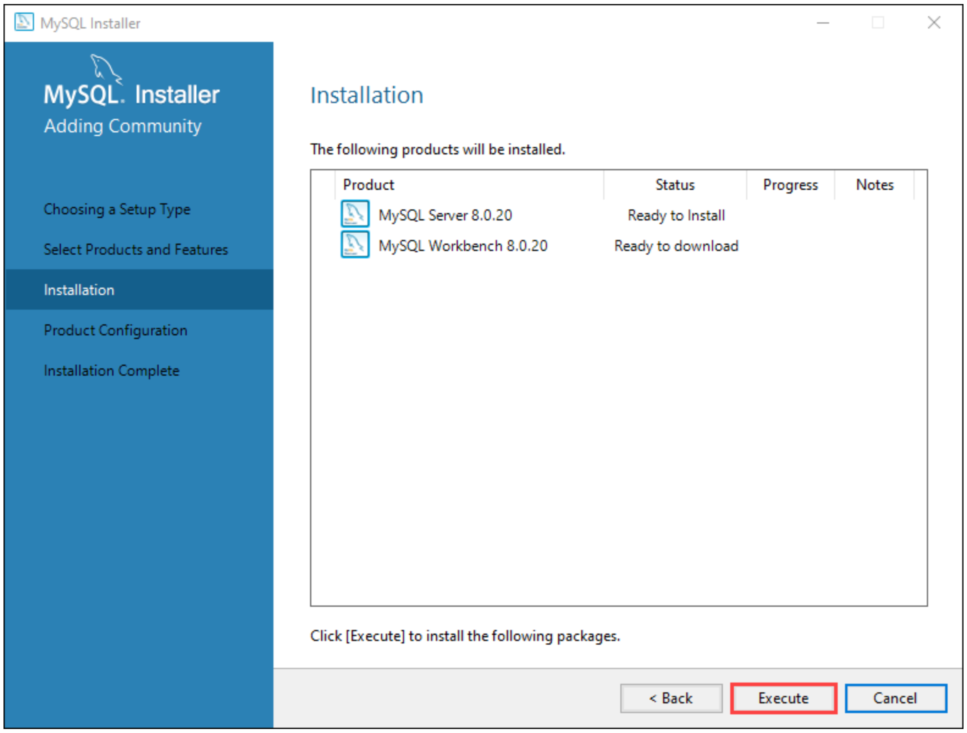
```
  ⊞ MySQL Server 5.7
  ⊞ MySQL Server 5.6
⊟ Applications
  ⊟ MySQL Workbench
    ⊟ MySQL Workbench 8.0
        MySQL Workbench 8.0.20
        MySQL Workbench 8.0.19
        MySQL Workbench 8.0.18
        MySQL Workbench 8.0.17
        MySQL Workbench 8.0.16
        MySQL Workbench 8.0.15
        MySQL Workbench 8.0.14
        MySQL Workbench 8.0.13
        MySQL Workbench 8.0.12
```

Products/Features To Be Installed:

```
⊞ MySQL Server 8.0.20 - X64
⊞ MySQL Workbench 8.0.20 - X64
```

Published: Tuesday, April 14, 2020
Estimated Size: 122 MB
Release Notes: http://dev.mysql.com/doc/relnotes/workbench/en/wb-news-8-0-20.html

7. On the Installation screen, click **Execute**.



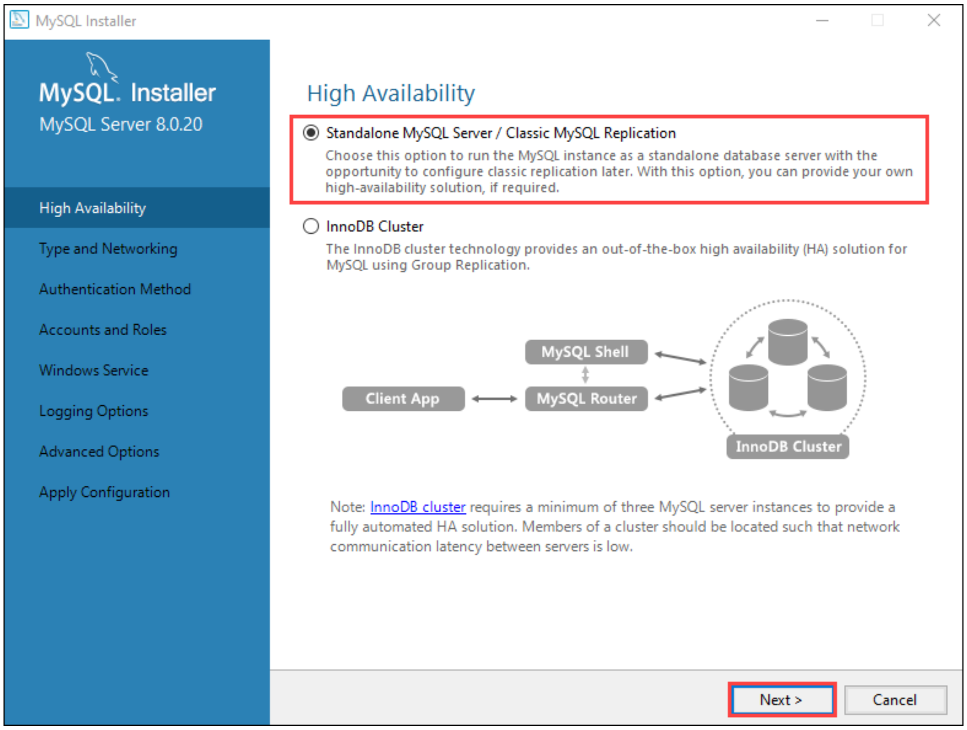
Installation

The following products will be installed.

Product	Status	Progress	Notes
MySQL Server 8.0.20	Ready to Install		
MySQL Workbench 8.0.20	Ready to download		

Click [Execute] to install the following packages.

8. After these are downloaded and installed, you see the Product Configuration page. Click **Next**.
9. In this guide, we're going to use a standalone server. On the High Availability page, select **Standalone MySQL Server / Classic MySQL Replication**, and click **Next.**



High Availability

● Standalone MySQL Server / Classic MySQL Replication

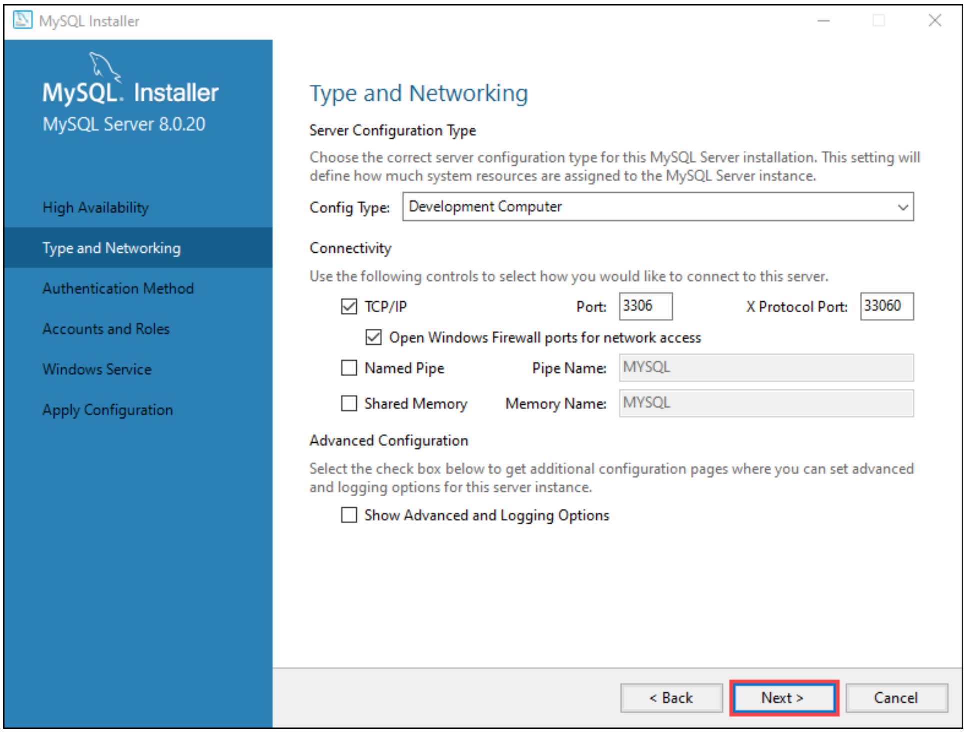
Choose this option to run the MySQL instance as a standalone database server with the opportunity to configure classic replication later. With this option, you can provide your own high-availability solution, if required.

○ InnoDB Cluster

The InnoDB cluster technology provides an out-of-the-box high availability (HA) solution for MySQL using Group Replication.

Note: InnoDB cluster requires a minimum of three MySQL server instances to provide a fully automated HA solution. Members of a cluster should be located such that network communication latency between servers is low.

10. Leave the default settings on the Type and Networking page. Click **Next**.



Type and Networking

Server Configuration Type

Choose the correct server configuration type for this MySQL Server installation. This setting will define how much system resources are assigned to the MySQL Server instance.

Config Type: Development Computer

Connectivity

Use the following controls to select how you would like to connect to this server.

☑ TCP/IP Port: 3306 X Protocol Port: 33060

☑ Open Windows Firewall ports for network access

☐ Named Pipe Pipe Name: MYSQL

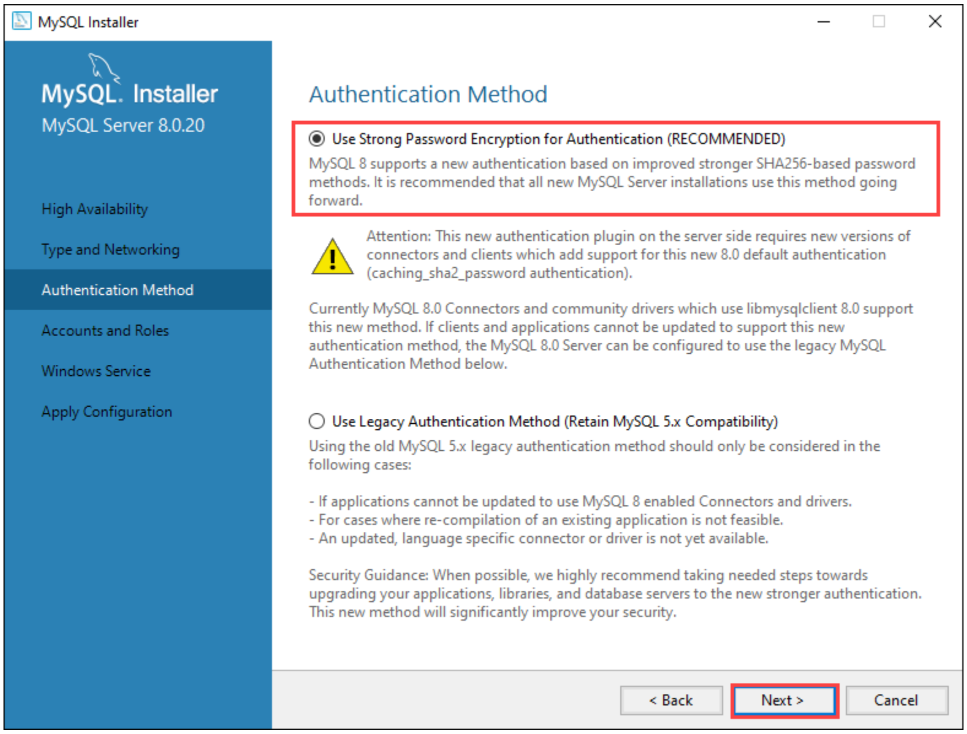
☐ Shared Memory Memory Name: MYSQL

Advanced Configuration

Select the check box below to get additional configuration pages where you can set advanced and logging options for this server instance.

☐ Show Advanced and Logging Options

11. On the Authentication Method page, choose **Use Strong Password Encryption for Authentication**. Click **Next**.


MySQL Installer

MySQL. Installer
MySQL Server 8.0.20

High Availability
Type and Networking
Authentication Method
Accounts and Roles
Windows Service
Apply Configuration

Authentication Method

● Use Strong Password Encryption for Authentication (RECOMMENDED)

MySQL 8 supports a new authentication based on improved stronger SHA256-based password methods. It is recommended that all new MySQL Server installations use this method going forward.

Attention: This new authentication plugin on the server side requires new versions of connectors and clients which add support for this new 8.0 default authentication (caching_sha2_password authentication).

Currently MySQL 8.0 Connectors and community drivers which use libmysqlclient 8.0 support this new method. If clients and applications cannot be updated to support this new authentication method, the MySQL 8.0 Server can be configured to use the legacy MySQL Authentication Method below.

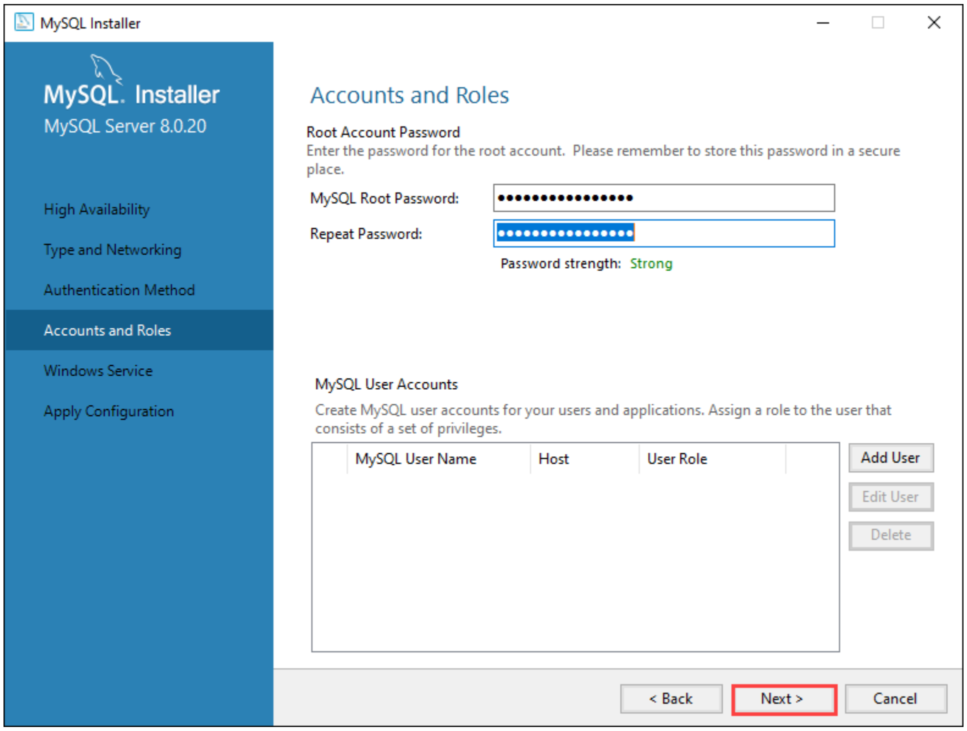
○ Use Legacy Authentication Method (Retain MySQL 5.x Compatibility)

Using the old MySQL 5.x legacy authentication method should only be considered in the following cases:

- If applications cannot be updated to use MySQL 8 enabled Connectors and drivers.
- For cases where re-compilation of an existing application is not feasible.
- An updated, language specific connector or driver is not yet available.

Security Guidance: When possible, we highly recommend taking needed steps towards upgrading your applications, libraries, and database servers to the new stronger authentication. This new method will significantly improve your security.

12. Create a strong password and click **Next**.


MySQL Installer

MySQL. Installer
MySQL Server 8.0.20

High Availability
Type and Networking
Authentication Method
Accounts and Roles
Windows Service
Apply Configuration

Accounts and Roles

Root Account Password
Enter the password for the root account. Please remember to store this password in a secure place.

MySQL Root Password: ●●●●●●●●●●●●●●●●

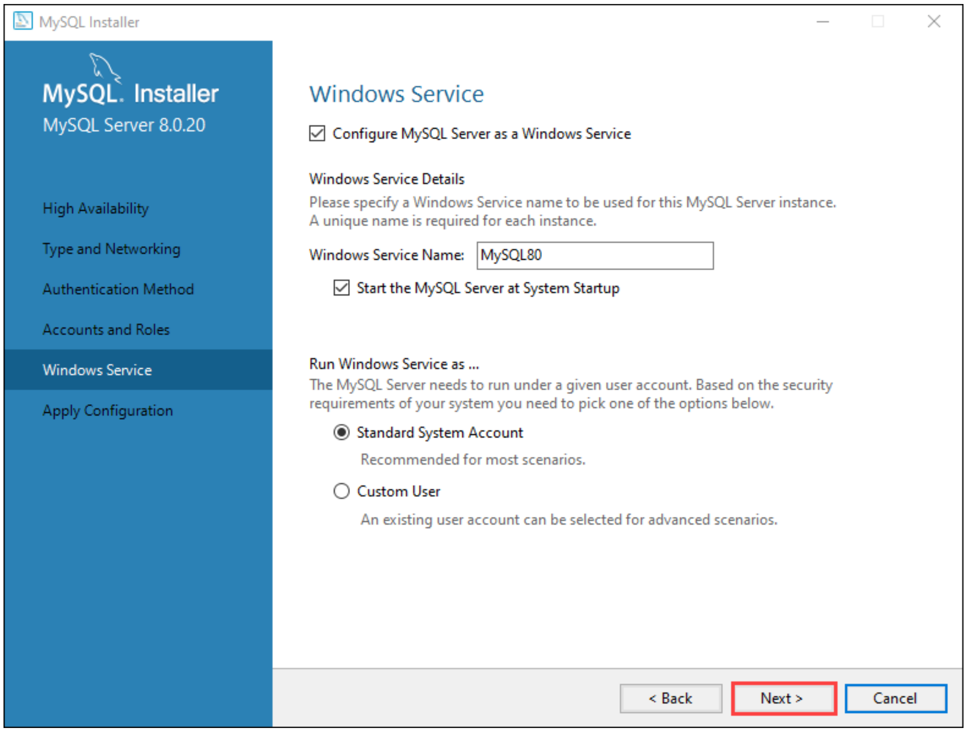
Repeat Password: ●●●●●●●●●●●●●●●●

Password strength: Strong

MySQL User Accounts
Create MySQL user accounts for your users and applications. Assign a role to the user that consists of a set of privileges.

MySQL User Name	Host	User Role

13. On the Windows Service page, leave the default settings and click **Next**.



Windows Service

☑ Configure MySQL Server as a Windows Service

Windows Service Details

Please specify a Windows Service name to be used for this MySQL Server instance.
A unique name is required for each instance.

Windows Service Name: MySQL80

☑ Start the MySQL Server at System Startup

Run Windows Service as ...

The MySQL Server needs to run under a given user account. Based on the security requirements of your system you need to pick one of the options below.

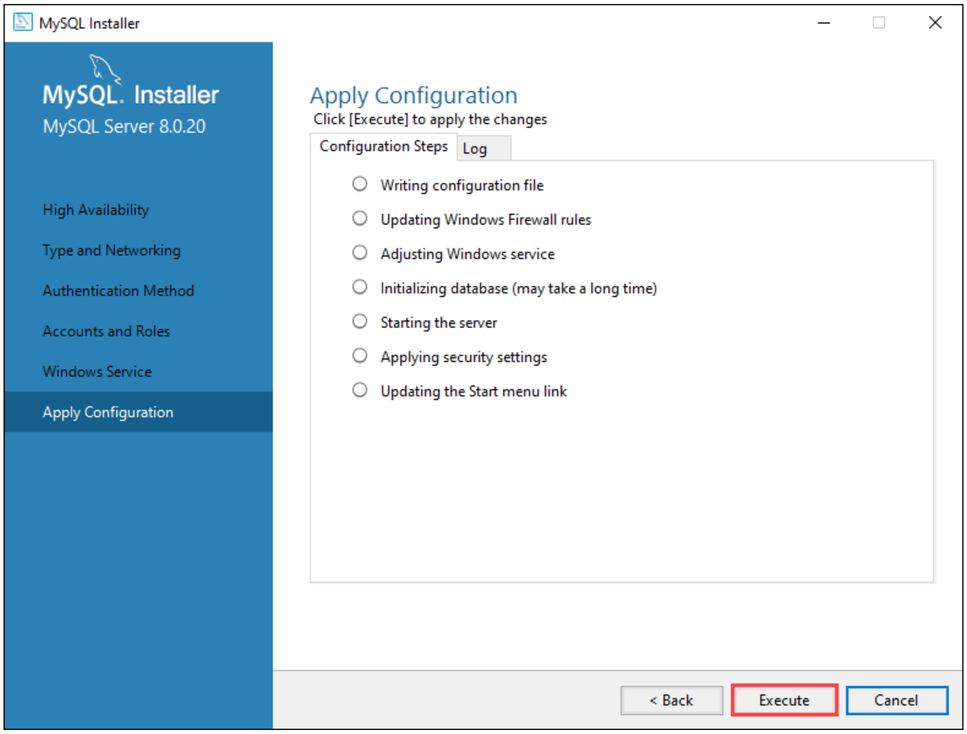
● Standard System Account

Recommended for most scenarios.

○ Custom User

An existing user account can be selected for advanced scenarios.

14. On the Apply Configuration page, click Execute.

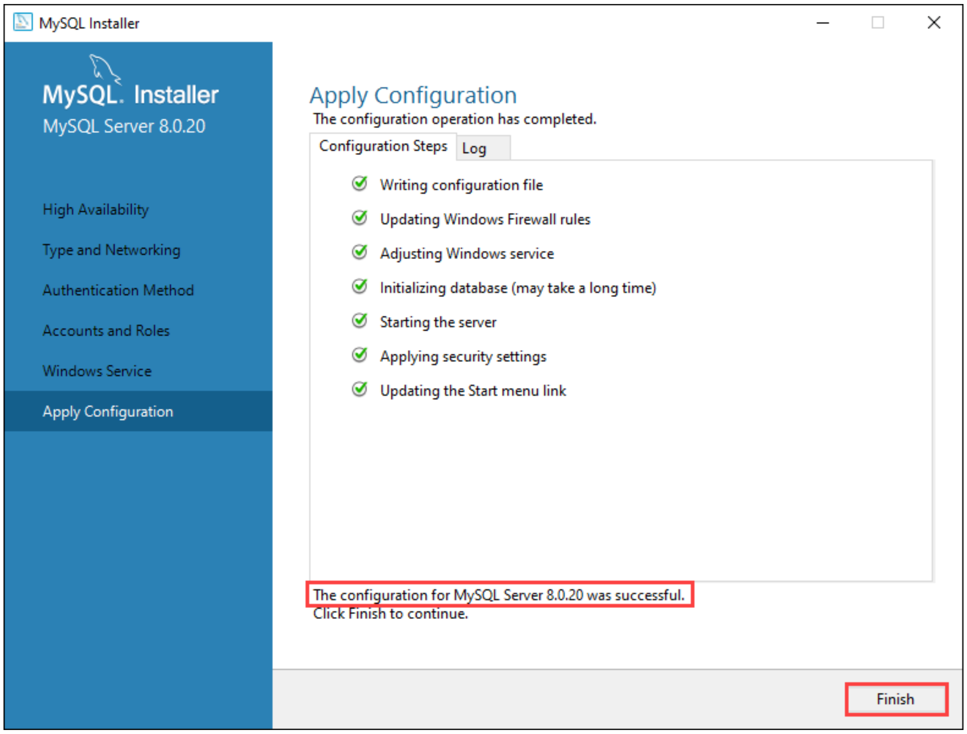


Apply Configuration

Click [Execute] to apply the changes

○ Writing configuration file
○ Updating Windows Firewall rules
○ Adjusting Windows service
○ Initializing database (may take a long time)
○ Starting the server
○ Applying security settings
○ Updating the Start menu link

15. Once the configuration is applied and the database is initialized, you'll see a confirmation message. Click **Finish** to complete the install process.



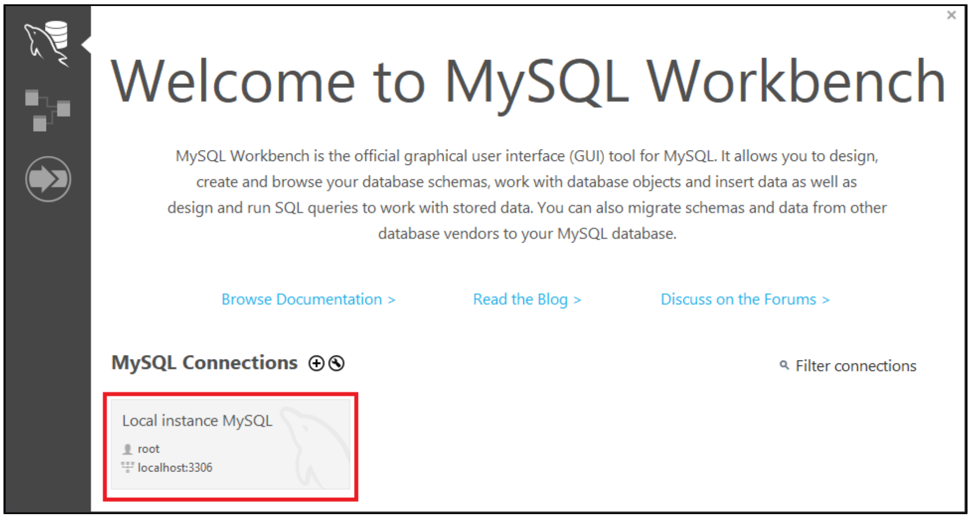
Apply Configuration

The configuration operation has completed.

- ✓ Writing configuration file
- ✓ Updating Windows Firewall rules
- ✓ Adjusting Windows service
- ✓ Initializing database (may take a long time)
- ✓ Starting the server
- ✓ Applying security settings
- ✓ Updating the Start menu link

The configuration for MySQL Server 8.0.20 was successful.
Click Finish to continue.

16. The **MySQL Workbench** window is displayed. MySQL Workbench lets you to administrate the MySQL server.
17. Click on **Local instance MySQL** to connect to the newly installed MySQL server.

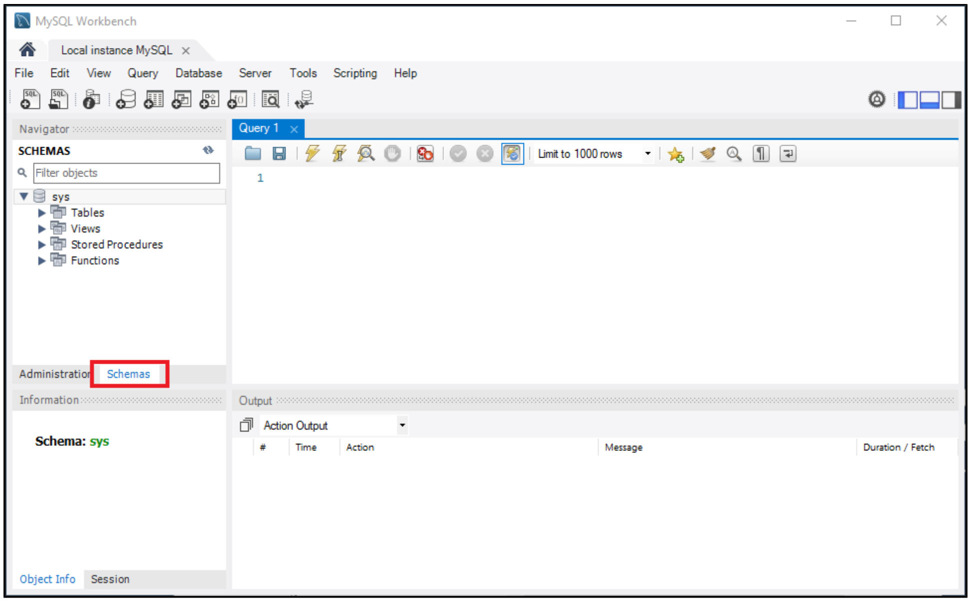


Welcome to MySQL Workbench

MySQL Workbench is the official graphical user interface (GUI) tool for MySQL. It allows you to design, create and browse your database schemas, work with database objects and insert data as well as design and run SQL queries to work with stored data. You can also migrate schemas and data from other database vendors to your MySQL database.

18. Enter the root **password** you earlier entered into the installer and click **OK**.

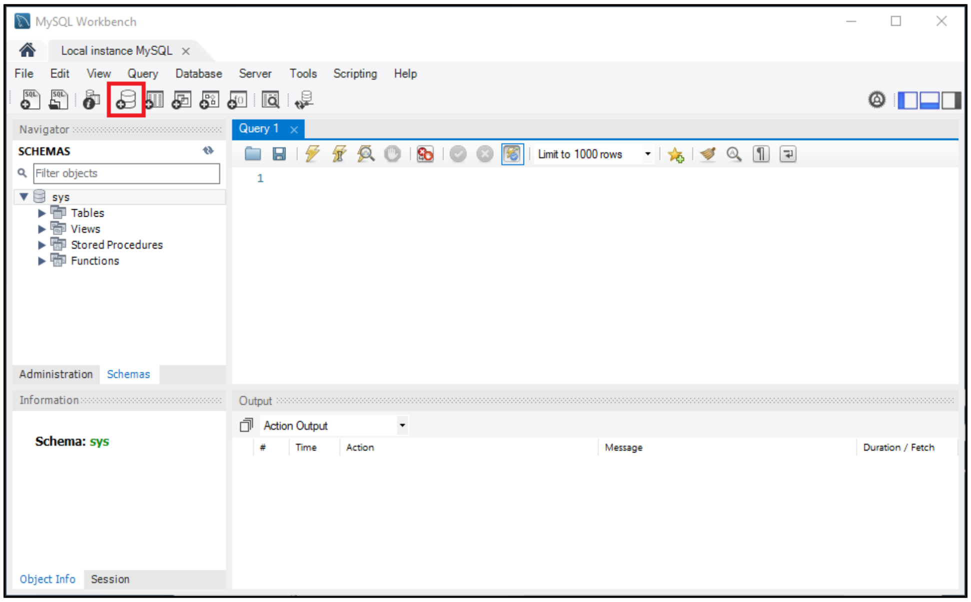
19. You are now connected and can see, in the Navigator, the default **sys** schema. Click on the **Schemas** tab to see a listing of schemas



Schema: sys

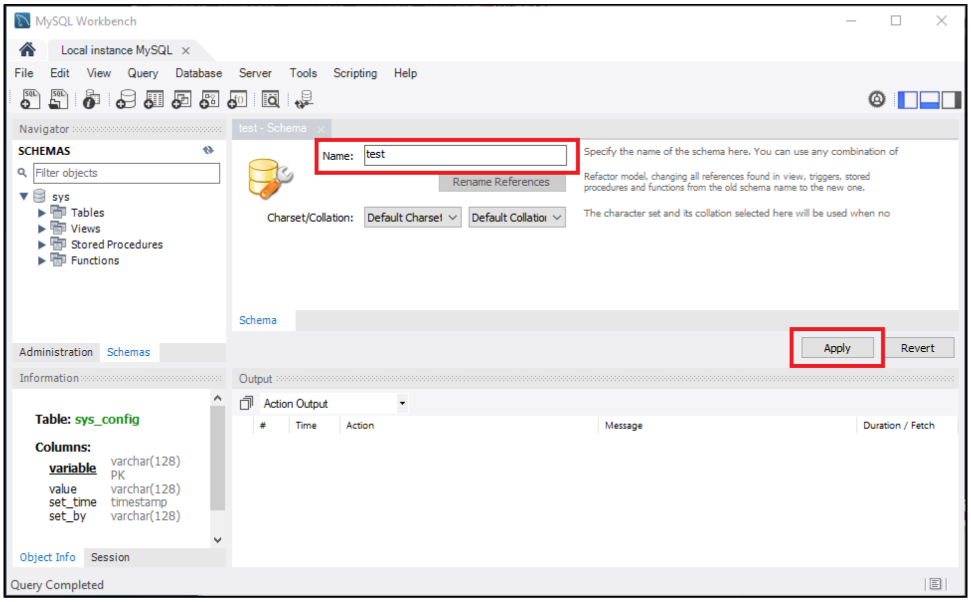
20. A "schema" is a collection of tables and other database objects. Ignition needs a schema to connect to. Instead of using the **sys** schema, we can create a scheme dedicated to Ignition.

    To create the schema, click the **Create New Schema** icon.



Schema: sys

21. You'll see the Schema creation tab. This allows you to make a new **Schema.** Type in **test** as the name, and click **Apply**. Technically you can call the schema anything the database allows, but the default MySQL connection in Ignition assumes a schema named "test". If you name the schema something else here, you'll need to remember the name when creating the database connection in Ignition later.



Name: test

Charset/Collation: Default Charset Default Collation

Specify the name of the schema here. You can use any combination of

Refactor model, changing all references found in view, triggers, stored procedures and functions from the old schema name to the new one.

The character set and its collation selected here will be used when no

Table: sys_config

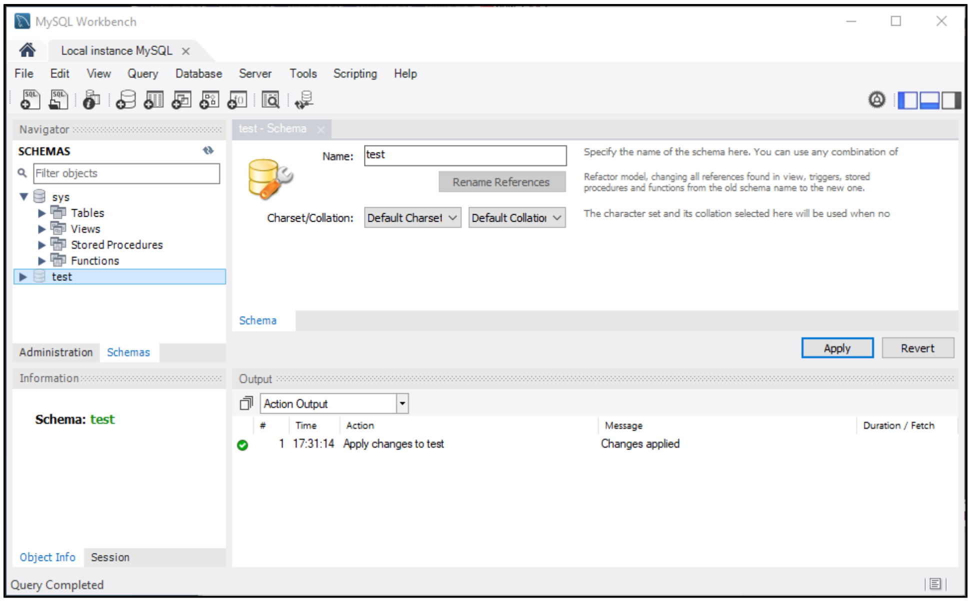
Columns:

variable	varchar(128) PK
value	varchar(128)
set_time	timestamp
set_by	varchar(128)

#	Time	Action	Message	Duration / Fetch

Query Completed

22. You'll see the **Apply SQL Script to Database** window. Click **Apply**.
23. If there was an issue, the following window will state the issue. Go back and address the issue. Otherwise, click the **Finish** button. You should now see the **test** schema in the Navigator.



Name: test

Charset/Collation: Default Charset Default Collation

Specify the name of the schema here. You can use any combination of

Refactor model, changing all references found in view, triggers, stored procedures and functions from the old schema name to the new one.

The character set and its collation selected here will be used when no

Schema: test

	#	Time	Action	Message	Duration / Fetch
✓	1	17:31:14	Apply changes to test	Changes applied	

Query Completed

Now that the database is installed, you can connect Ignition to it. Learn more here: Connecting to MySQL.

Installing Microsoft SQL Server Express

You need to download and install both the **SQL Server Express** and the **SQL Management Studio** before you connect to the database.

On this page ...

INDUCTIVE UNIVERSITY

Installing Microsoft SQL Server Express

Watch the Video

To Download SQL Server Express

This section walks through the process of installing a new instance of SQL Server Express.

1. Go to https://www.microsoft.com/en-us/sql-server/sql-server-downloads
2. Look for the link/button to download the **Express Edition**.
3. Run the installer
4. When given a choice between installation type, select **Custom**.
5. Select an installation directory. If you don't have a preference, simply use the default.
6. The installer will unpack and download required files.
7. Once the installer is ready, the **SQL Server Installation Center** window will appear. Select the **Installation** heading on the side bar, and click on the **New SQL Server stand-alone installation or add features to an existing installation** link.
8. On the **License Terms** window, choose **I accept the license terms** and click **Next**.
9. On the **Product Updates** window, click **Next** to start installing the SQL Server.
10. On the **Feature Selection** window, stay with the default selections and click **Next**. Technically you can uncheck many of the options, like the optional R and Python installations. The Server and SQL Server Management Studio are the main tools you'll need when interacting with an Ignition installation. You can always remove additional components from the SQL Server installer later if you choose so.
11. The **Instance Configuration** window shows **Named instance: SQL Express**, you can keep it or change it if you like. Click **Next**.
12. When asked about a JRE to use, you can use the provided JRE. Click **Next**.
13. On the **Server Configuration** window, choose **Automatic** from the dropdown under **Start up Type** for the **SQL Server Browser** service, and click **Next**.
14. On the **Database Engine Configuration** window, for **Authentication Mode** choose **Mixed Mode**, enter a **password** for the SA account. Note that you can use a Windows Authentication Mode with Ignition, but it does require some additional configuration when connecting later on. In either case, click **Next**.
15. On the next few windows, continue clicking **Next** until it shows installation is **Complete**. **SQL Server Express** is now installed.

To Download SQL Management Studio

1. Go to http://www.microsoft.com/en-us/download/details.aspx?id=8961
2. Click on the **Download** button.
3. Click on the **SQLManagementStudio_x64_ENU.exe** file to run the executable.
4. On the **SQL Server Installation Center** window, click on the **New SQL Server stand-alone installation or add features to an existing installation** link.
5. On the **Installation Type**, keep the defaults and click **Next**.

6. On the **License Terms** window, choose **I accept the license terms** and click **Next**.
7. On the **Product Updates** window, click **Next** to start installing the SQL Server.
8. On the **Feature Selection** window, stay with the default selection of **Management Tools** and click **Next**.
9. On the **Database Engine Configuration** window, for **Authentication Mode** choose **Mixed Mode**, enter a **password**, and click **Next**.
10. On the next few windows, continue clicking **Next** until it shows installation is **Complete**.
    The **Management Tools** is now installed.
11. To run the program, go to **Start > Programs > Microsoft SQL Server > SQL Server Management Studio**.
12. Click on **Connect** to connect to the Microsoft SQL Server.
13. In **Object Explorer**, you can now see some databases under **Databases > System Databases**.
14. Right-click on **Databases** and select **New Databases...**.
    The **New Databases** window is displayed.
15. In **Database name**, enter **test**, click **Add,** and then **OK**.
    Now you can see the **test** database in the **Databases** folder and can connect Ignition to it, see Connecting to Microsoft SQL Server Express.

Installing PostgreSQL

To Install the PostreSQL Database

1. Go to the **Postgre SQL** website at http://www.postgresql.org
2. Click on **Downloads**, look for and click on the **Windows** link.
3. On the **Windows installers** page, find the **Download** link and click on it, and on the next page select the installer you are interested in.
   For example, you can select the **Wins x86-32**.
4. From you Download folder on your computer, click on the Postgre **.exe** file to install the database.
5. Go through all the windows of the **Install Wizard** until installation is complete.
6. From the Windows **Start** menu, open the **Postgre pgAdmin** database.
7. In the **Object browser** of the **pgAdmin** window, right-click on **PostgreSQL**, select **Connect**, enter your **password** to connect to the Server, and click **OK**.
   You will now see the **Databases** folder in the Object browser.

On this page ...

Installing PostgreSQL

Watch the Video

Connecting to Databases

On this page

...

Connect Once

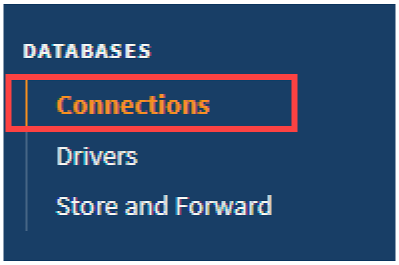
Many of the advanced features of Ignition, such as the Transaction Groups and Tags Historian require a connection to an external database and most databases require special permissions for each computer that wants to connect. Fortunately, Ignition takes care of all of this for us. You can create a connection to your database once and every system in Ignition will use that central connection. There's no need to worry about updating your database settings to add another client.

This central database connection also makes it easy to swap between databases or schemas. You can tell every query to use the default connection, then just change the default to update everything. Alternatively, you can force specific queries or systems to use a particular connection. Create as many database connections as you want and start designing using all of them.

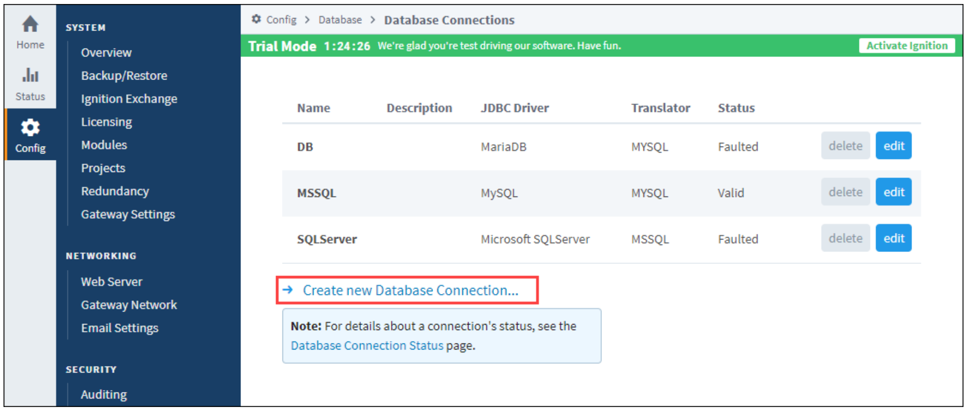
Add a Database Connection

Now that we've installed your database, let's connect to it. You can find detailed descriptions for many database connections in this User Manual, however, they all include the same steps:

1. On the Gateway Webpage, go to the Gateway **Config** tab. Scroll down to the **Databases > Connections** section.



2. On at the Database Connections page, click on the **Create new Database Connection...** link at the bottom of the table.



Name	Description	JDBC Driver	Translator	Status	
DB		MariaDB	MYSQL	Faulted	delete edit
MSSQL		MySQL	MYSQL	Valid	delete edit
SQLServer		Microsoft SQLServer	MSSQL	Faulted	delete edit

→ Create new Database Connection...

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

3. The next step is to choose a JDBC Driver.
   Ignition connects to databases using JDBC drivers that are unique to each database. Drivers for the most popular databases are included so there is usually no need to install the JDBC driver manually.

   Ignition ships with drivers for Microsoft SQL Server, MySQL, Oracle, and PostgreSQL. Pick the JDBC driver for your database, and click on the **Next** button.
   If a suitable driver is not available in the list, you need to add a new JDBC driver for other databases, like IBM DB2, which is not very difficult to do, see Adding a JDBC Driver.

4. **Configure the Connection**
   After selecting the driver, you'll configure the settings for the connection. Some settings, such as the Connect URL are specific to the

driver that you're using.

Main Database Connection Properties

Name	Each database connection needs a unique name, which consists of letters, numbers and underscores.
Description	A brief description of the database.
JDBC Driver	The JDBC driver dictates the type of database that this connection can connect to. It cannot be changed once created.
Connect URL	A string that instructs the driver how to connect to the database. This string is the server address, and may include the port, instance name, database name, and so on. The format and parameters depend on the driver being used.
Username	The username to use when connecting. Some databases support other authentication methods, such as Windows authentication, in which case this field is not used.
Change Password?	Check the box to change the existing password.
Password	Enter password.
Password	Re-type password for verification.
Extra Connection Properties	Depending on which database you are connecting to, there will be different default values placed in this box. MS SQL Server requires you to place your database name here, but for other databases you can usually leave this at its default values. Each database has its own set of available extra connection properties so you must refer to your Database documentation to determine what is valid here.
Enabled	Lets you to enable or disable a database connection.
Failover Datasource	The connection that is automatically used when this connection is not available.
Failover Mode	Lets you select how to handle the database connection failing and recovering. Database connections support *failover.* This means that the objects which use a database connection will use a different connection if the one they are using becomes unavailable. The **Failover Datasource** property determines which connection is used, and the **Failover Mode** determines when, if ever, the connection is switch back to the primary connection. There are two failover modes: • **STANDARD** mode means that this datasource will fail over when a connection cannot be retrieved, but when connectivity is restored, connections will again come from this datasource. • **STICKY** mode means that once this datasource fails over, connections will continue coming from the failover datasource until the failover datasource itself fails or the Gateway is restarted.
Slow Query Log Threshold	Queries that take longer than this amount of time, in milliseconds, will be logged. This helps to find queries that are not performing well. (default: 60,000)
Validation Timeout	The time in milliseconds between database validation checks. (default: 10,000)

Advanced Settings

There are many advanced settings that you don't need to change under normal circumstances. See the description for each property on the settings page.

In This Section ...

Connecting to MySQL

Connecting to a MySQL Database

In order to get connected to MySQL, you must have a Translator, a Driver, and a Connection. The Translator and Driver only needs to be installed once, and after that you can make as many connections as you want to any MySQL databases.

MySQL 5.7 and prior can use the built-in MariaDB connection type, as long as you are not using an the updated encryption from the new version.

> ⚠ When you Upgrade Ignition, any existing drivers are carried over. This means only a fresh install of Ignition will not have a MySQL Connector.
>
> You can find the Drivers you need on the JDBC Drivers and Translators page.

On this page

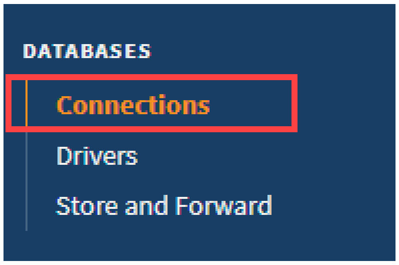
...

Connecting to MySQL

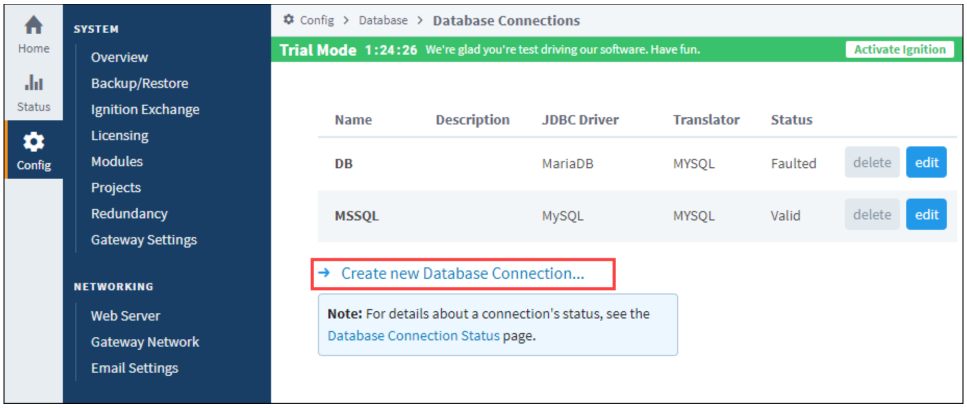
Watch the Video

Connect Ignition to MySQL Database

1. On the Gateway Webpage, go to the **Config** section.
2. Scroll down to **Databases > Connections**.



3. The Database Connections page is displayed. Click on **Create new Database Connection...**.



Name	Description	JDBC Driver	Translator	Status	
DB		MariaDB	MYSQL	Faulted	delete edit
MSSQL		MySQL	MYSQL	Valid	delete edit

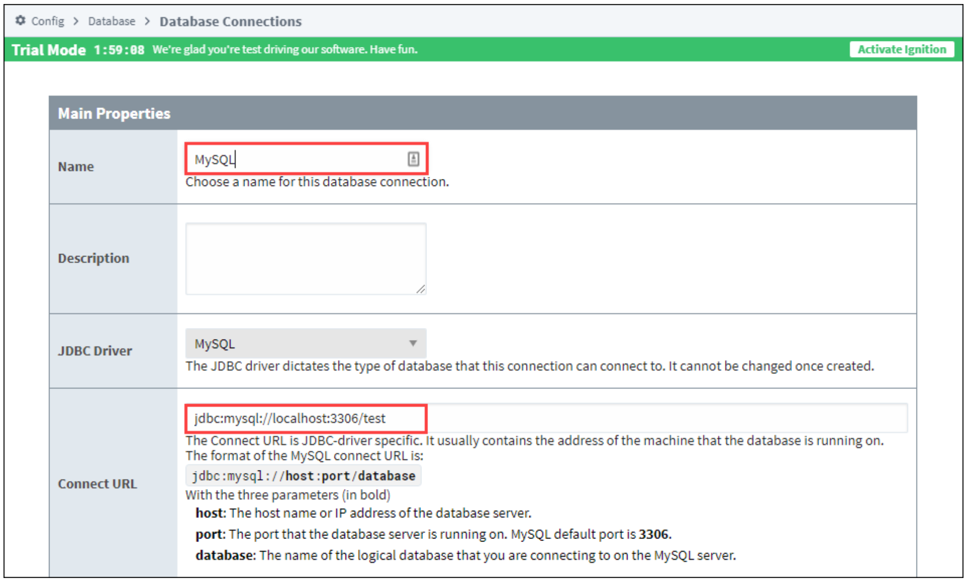
→ Create new Database Connection...

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

4. Select the **MySQL** driver and click **Next**.
5. On the **New Database Connection** page, enter the following information:

   Name: **MySQL** (use a meaningful name such as **MySQL**)

   Connect URL: **jdbc:mysql://localhost:3306/test** (By default, MySQL creates an empty database called **test**)


Config > Database > Database Connections
Trial Mode 1:59:08 We're glad you're test driving our software. Have fun.
Activate Ignition

Main Properties

Name	MySQL Choose a name for this database connection.
Description	
JDBC Driver	MySQL The JDBC driver dictates the type of database that this connection can connect to. It cannot be changed once created.
Connect URL	jdbc:mysql://localhost:3306/test The Connect URL is JDBC-driver specific. It usually contains the address of the machine that the database is running on. The format of the MySQL connect URL is: `jdbc:mysql://host:port/database` With the three parameters (in bold) **host:** The host name or IP address of the database server. **port:** The port that the database server is running on. MySQL default port is **3306**. **database:** The name of the logical database that you are connecting to on the MySQL server.

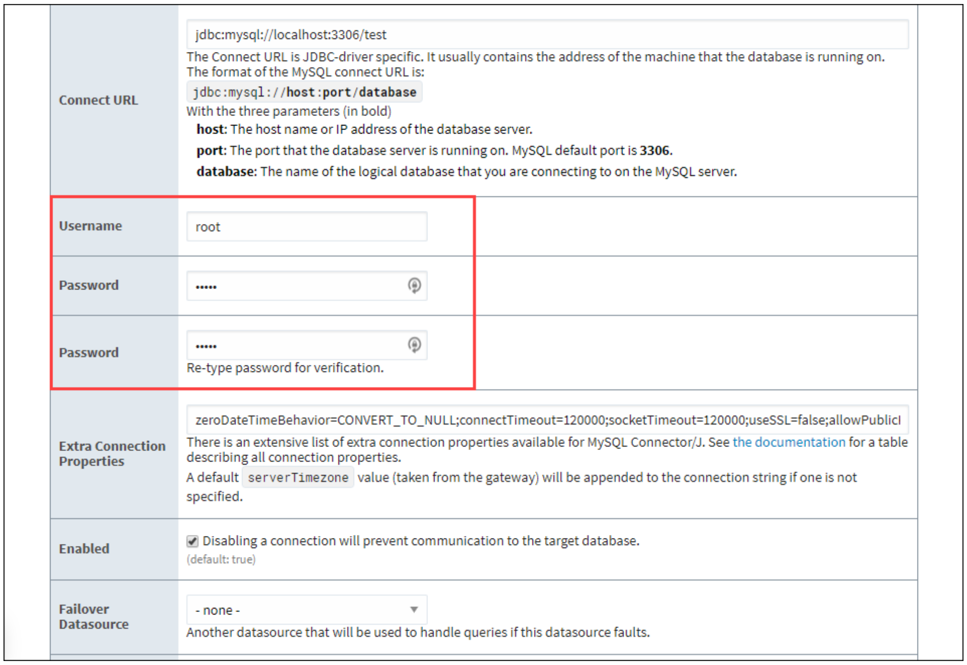
   As you see in the example above, MySQL uses the following **Connect URL** format: **jdbc:mysql://hostaddress:3306/database**

   Where **hostaddress** is the address of the machine with MySQL installed, for example: localhost, 192.168.1.1, db-server, etc., and **database** specifies the database schema the connection will target. It's important to understand that a MySQL server can host many database files. The connection will target one schema (a collection of tables and other objects) in the database.

6. To configure the connection, enter the following information:

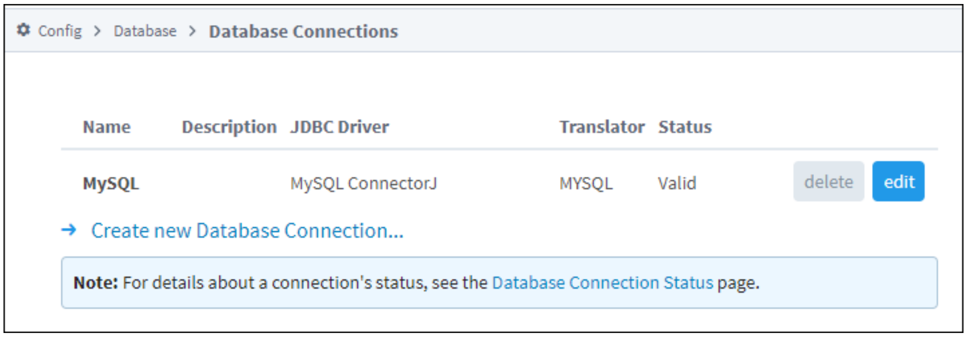
   Username: **root**
   Password: **mysql** (password is what you entered during MySQL server installation, **mysql** is the password for this example)

   Extra Connection Properties: Leave at the default value. By default, the **zeroDateTimeBehavior** parameter is defined and it is usually not necessary to add more parameters unless you are planning to store text outside of the range of 7-bit ASCII (a-z). If you are, add **useUnicode=yes;characterEncoding=utf8;** to the end of the extra connection properties, including the semi-colons.



Connect URL	jdbc:mysql://localhost:3306/test The Connect URL is JDBC-driver specific. It usually contains the address of the machine that the database is running on. The format of the MySQL connect URL is: `jdbc:mysql://host:port/database` With the three parameters (in bold) **host:** The host name or IP address of the database server. **port:** The port that the database server is running on. MySQL default port is **3306**. **database:** The name of the logical database that you are connecting to on the MySQL server.
Username	root
Password	•••••
Password	••••• Re-type password for verification.
Extra Connection Properties	zeroDateTimeBehavior=CONVERT_TO_NULL;connectTimeout=120000;socketTimeout=120000;useSSL=false;allowPublicl There is an extensive list of extra connection properties available for MySQL Connector/J. See the documentation for a table describing all connection properties. A default `serverTimezone` value (taken from the gateway) will be appended to the connection string if one is not specified.
Enabled	☑ Disabling a connection will prevent communication to the target database. (default: true)
Failover Datasource	- none - Another datasource that will be used to handle queries if this datasource faults.

7. Click on **Create New Database Connection** at the bottom of the form. Your connection is now created and the Database Connections page is displayed showing the **Status** of your connection as **Valid**.



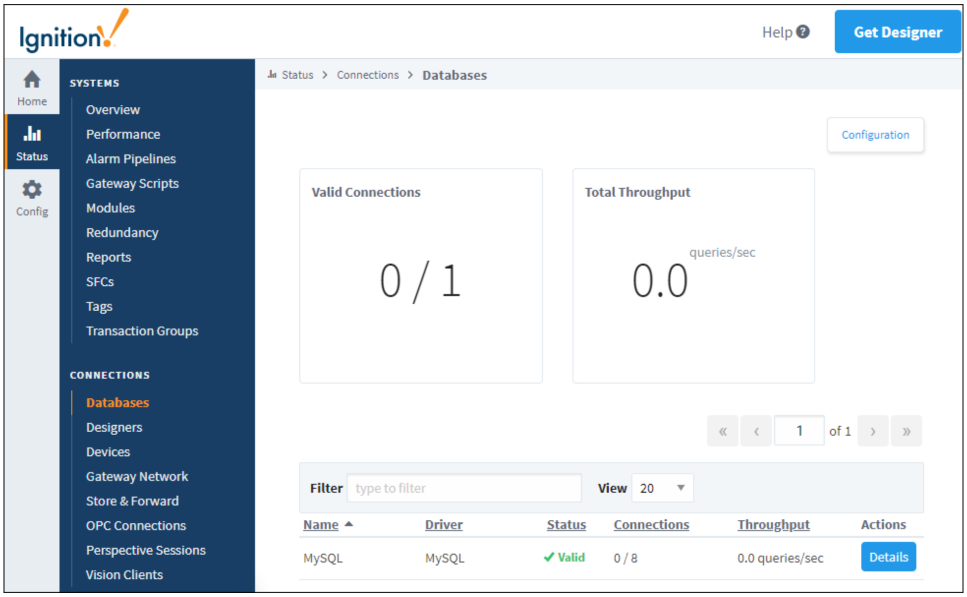
Config > Database > **Database Connections**

Name	Description	JDBC Driver	Translator	Status	
MySQL		MySQL ConnectorJ	MYSQL	Valid	delete edit

→ Create new Database Connection...

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

8. To display the details about the status of your database connection, see the **Note** on the above window and click on the **Database Connection Status** link. This will display any errors if your status is Faulted, in this example it shows the status as being Valid.



Status > Connections > **Databases**

Valid Connections

0 / 1

Total Throughput

0.0 queries/sec

1 of 1

Filter type to filter View 20

Name ▲	Driver	Status	Connections	Throughput	Actions
MySQL	MySQL	✓ Valid	0 / 8	0.0 queries/sec	Details

JDBC Drivers and Translators

MySQL connections can often use the included MariaDB translator, but we recommend using a separate MySQL Translator.

More information on configuring a JDBC driver and Translator can be found on the JDBC Drivers and Translators page. However, you may need to check the JDBC driver's documentation for information on how to configure them.

Related Topics ...

- Connecting to Microsoft SQL Server Express
- Store and Forward
- OPC UA
- Designer

Connecting to Microsoft SQL Server Express

SQL Server Connection Requirements

In order to get connected to SQL Server, you must have a Translator, a Driver, and a Connection. The Translator and Driver only needs to be installed once, and after that you can make as many connections as you want to any compatible SQL Server databases.

> ⚠ When you Upgrade Ignition, any existing drivers are carried over. This means only a fresh install of Ignition will not have a SQL Server Connector.

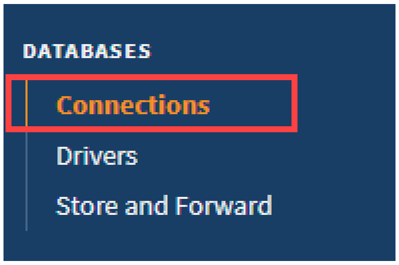
On this page...

Connecting to Microsoft SQL Server Express

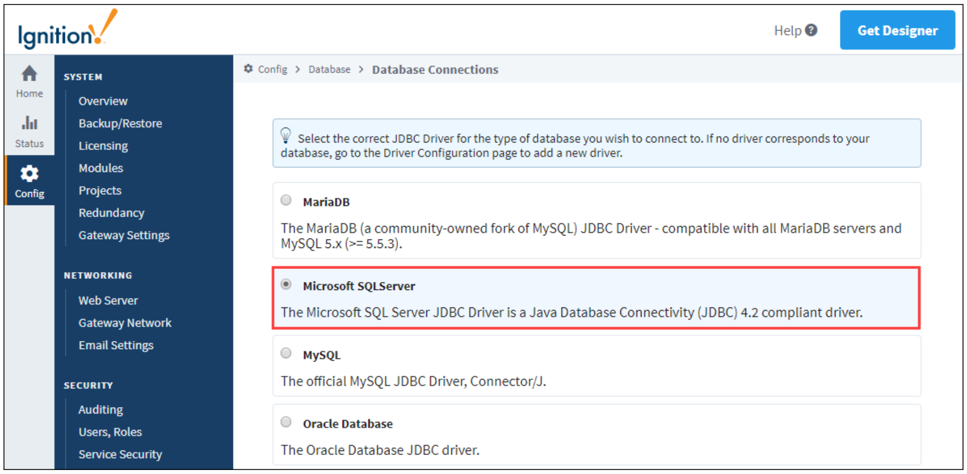
Watch the Video

Connect to Microsoft SQL Server

1. On the Gateway Webpage, go to the **Config** section.
2. Scroll down to **Databases > Connections**.



3. The Database Connections page is displayed. Click on **Create new Database Connection...**.
4. Select **Microsoft SQLServer JDBC Driver,** and click **Next**.



Select the correct JDBC Driver for the type of database you wish to connect to. If no driver corresponds to your database, go to the Driver Configuration page to add a new driver.

○ **MariaDB**

The MariaDB (a community-owned fork of MySQL) JDBC Driver - compatible with all MariaDB servers and MySQL 5.x (>= 5.5.3).

● **Microsoft SQLServer**

The Microsoft SQL Server JDBC Driver is a Java Database Connectivity (JDBC) 4.2 compliant driver.

○ **MySQL**

The official MySQL JDBC Driver, Connector/J.

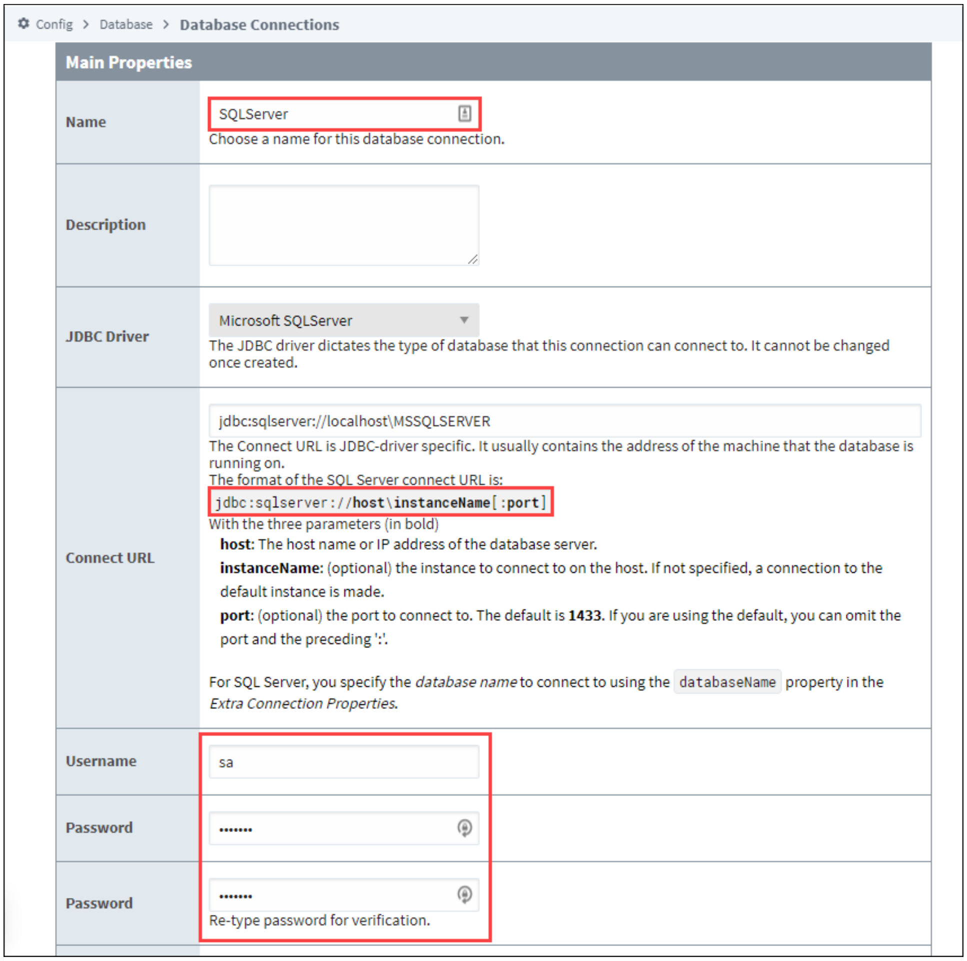
○ **Oracle Database**

The Oracle Database JDBC driver.

5. On the **Database Connections** page, enter the following information:
   Name: **SQLServer**
   Connect URL: **jdbc:sqlserver://localhost\SQLEXPRESS**

> ⚠ We are connecting to the express edition of SQL Server using the default instance name. If you have the full SQL Server with default settings, replace SQLEXPRESS with MSSQLSERVER.

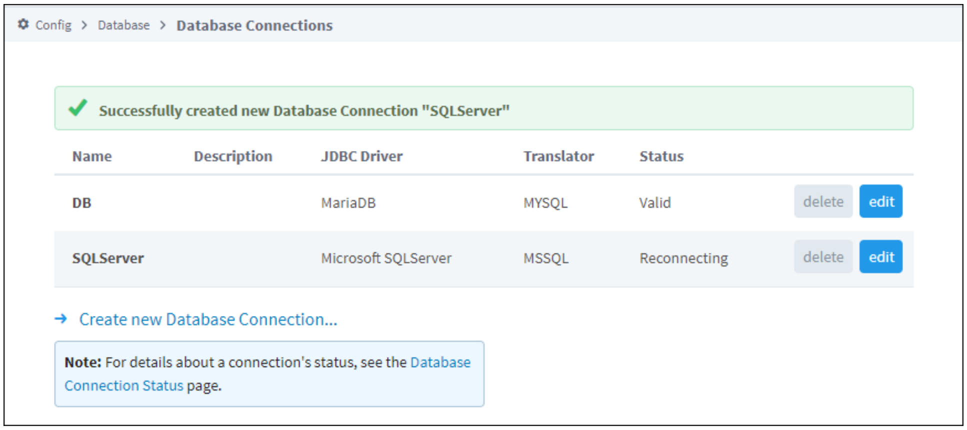
username: **sa**
password: **sqlserver** (password is what you entered during the SQL Server installation. For this example, password is **sqlserver**)



Main Properties

Name	SQLServer Choose a name for this database connection.
Description	
JDBC Driver	Microsoft SQLServer ▾ The JDBC driver dictates the type of database that this connection can connect to. It cannot be changed once created.
Connect URL	jdbc:sqlserver://localhost\MSSQLSERVER The Connect URL is JDBC-driver specific. It usually contains the address of the machine that the database is running on. The format of the SQL Server connect URL is: `jdbc:sqlserver://host\instanceName[:port]` With the three parameters (in bold) **host:** The host name or IP address of the database server. **instanceName:** (optional) the instance to connect to on the host. If not specified, a connection to the default instance is made. **port:** (optional) the port to connect to. The default is **1433**. If you are using the default, you can omit the port and the preceding ':'. For SQL Server, you specify the *database name* to connect to using the `databaseName` property in the *Extra Connection Properties*.
Username	sa
Password	•••••••
Password	••••••• Re-type password for verification.

6. At the bottom of the form, click on **Create New Database Connection**.
   Your connection is now created. The Database Connections page is displayed and will show the status of Reconnecting, then Valid.



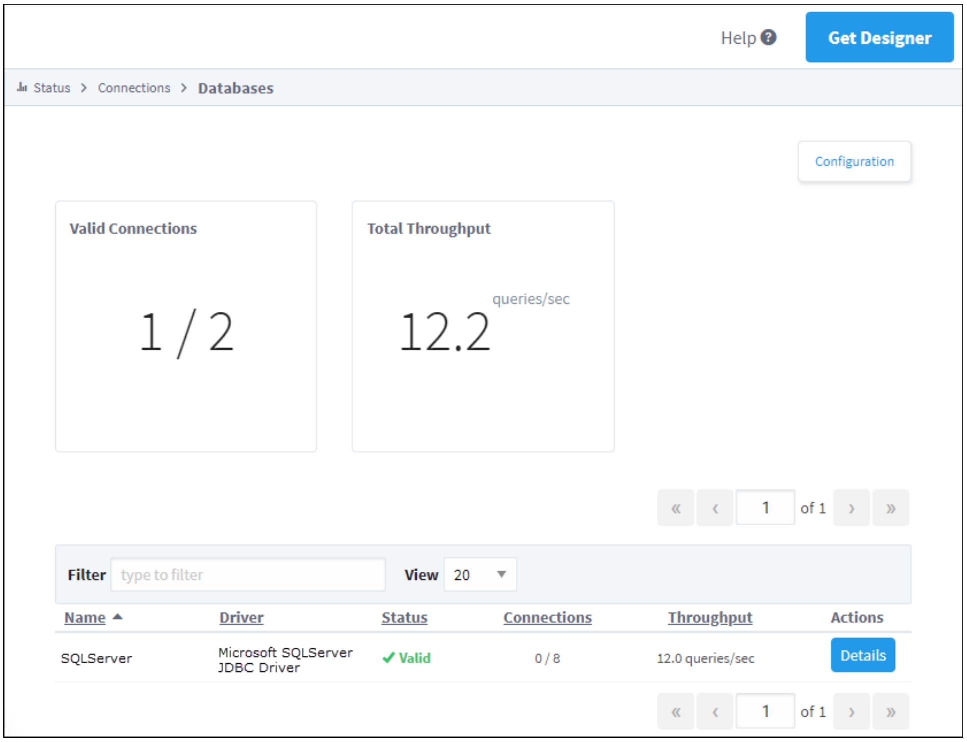
✓ Successfully created new Database Connection "SQLServer"

Name	Description	JDBC Driver	Translator	Status	
DB		MariaDB	MYSQL	Valid	delete edit
SQLServer		Microsoft SQLServer	MSSQL	Reconnecting	delete edit

→ Create new Database Connection...

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

7. To display the details about the status of your database connection, see the **Note** on the above window and click on the **Database Connection Status** link.



Valid Connections

1 / 2

Total Throughput

12.2 queries/sec

Name ▲	Driver	Status	Connections	Throughput	Actions
SQLServer	Microsoft SQLServer JDBC Driver	✓ Valid	0 / 8	12.0 queries/sec	Details

Microsoft SQL Server Connection Guide

This guide helps you with any difficulties you may have in getting the correct settings and parameters when connecting Ignition to Microsoft SQL Server, a popular and robust relational database.

Multiple Instances of Database

Microsoft SQL Server supports multiple instances of the database running concurrently on the same computer. Each instance has its own name and set of system and user databases that are not shared between instances. Applications, such as Ignition, can connect to each instance on a computer in much the same way they connect to databases running on different computers. By default, each instance gets assigned a dynamic TCP/IP port on startup that listens for any incoming requests. Since the port is dynamic and the application does not know what the new port is, it must connect using the instance name.

So if the communication is over TCP/IP and the application knows the instance name, how does the application find which port to communicate to?

The answer is the **Microsoft SQL Server Browser** service. The Microsoft SQL Server Browser program runs as a Windows service and listens for all incoming requests for resources and provides information, such as the TCP/IP port, about each instance installed on the computer. Microsoft SQL Server Browser also contributes to these two actions: browsing a list of available servers and connecting to the correct server instance.

If the Microsoft SQL Server Browser service is not running, you can still connect to SQL Server if you provide the correct port number. For example, you can connect to the default instance of SQL Server with TCP/IP if it is running on port 1433.

Check 1: Make Sure the Database has TCP/IP Enabled

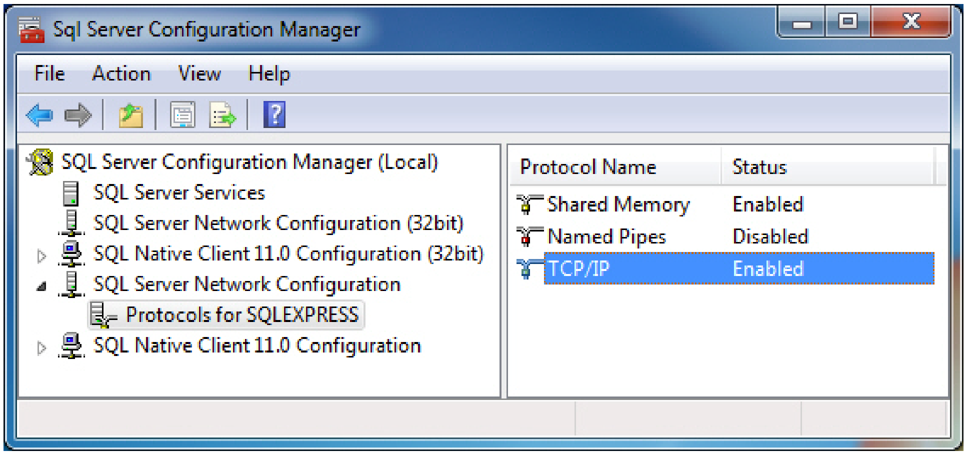
Ignition connects using TCP/IP, therefore make sure your database has TCP/IP enabled.

1. Open the **SQL Server Configuration Manager** from **Start > All Programs > Microsoft SQL Server Version # > Configuration Tools > SQL Server Configuration Manager.**

   The **Sql Server Configuration Manager** window is displayed.

2. To see all the instances setup on that machine, expand **SQL Server Network Configuration**.

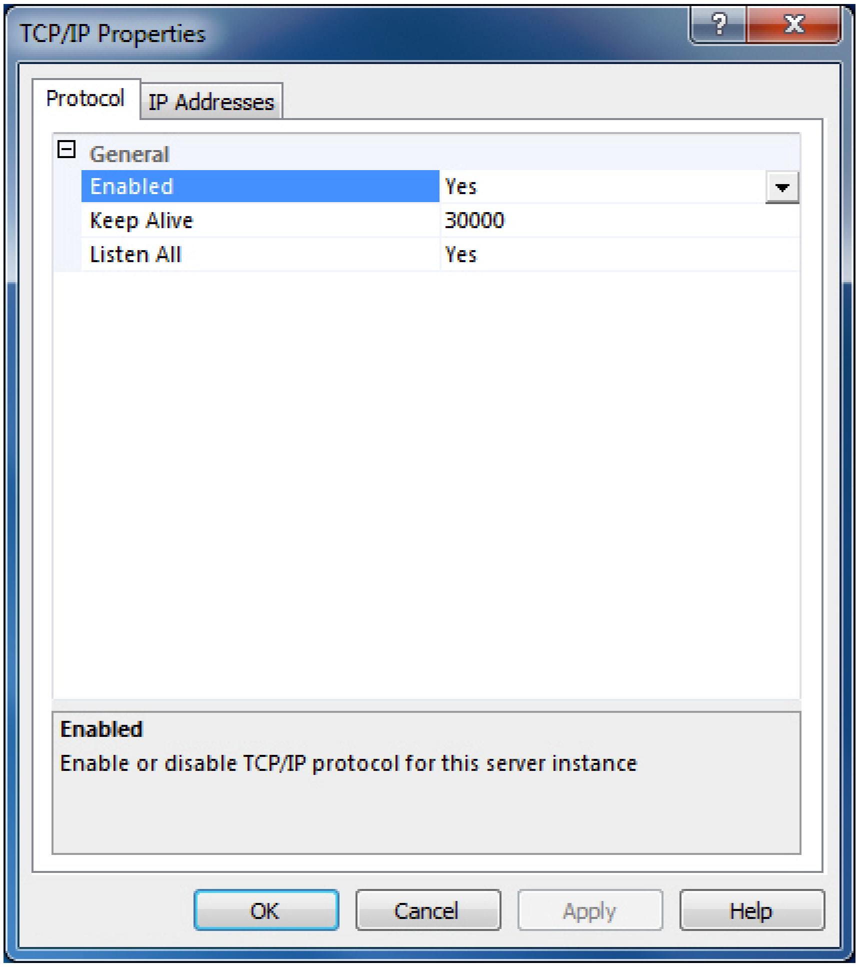
3. Find the database (or instance) you plan on using. To the right, all of the protocols the database supports are shown. Find the **TCP /IP** protocol and select it.



SQL Server Configuration Manager (Local)
SQL Server Services
SQL Server Network Configuration (32bit)
SQL Native Client 11.0 Configuration (32bit)
SQL Server Network Configuration
Protocols for SQLEXPRESS
SQL Native Client 11.0 Configuration

Protocol Name	Status
Shared Memory	Enabled
Named Pipes	Disabled
TCP/IP	Enabled

4. Make sure the **Status** next to TCP/IP is **Enabled**. If not, double-click **TCP/IP** and choose **Yes** from the drop-down next to Enabled and click **OK**.



General

Enabled	Yes
Keep Alive	30000
Listen All	Yes

Enabled

Enable or disable TCP/IP protocol for this server instance

Check 2: Make Sure Microsoft SQL Server Browser is Running

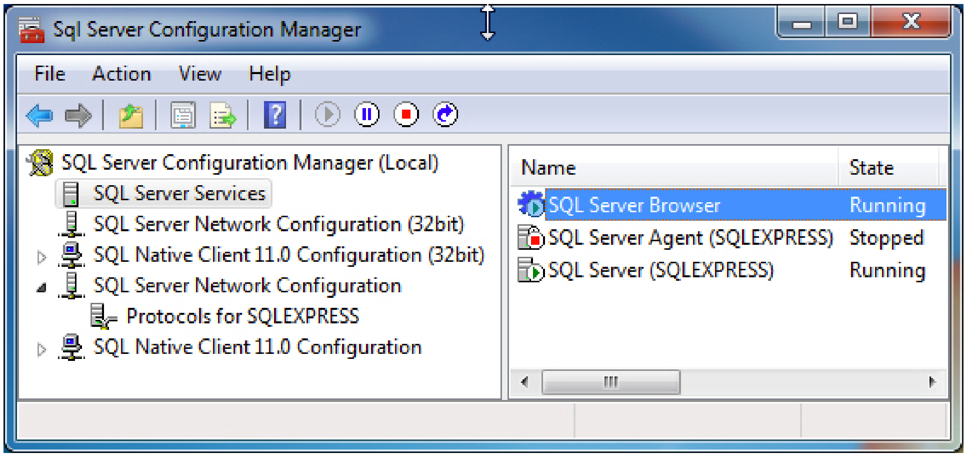
If you ARE connecting to your database using a NAMED INSTANCE, you must make sure that the Microsoft SQL Server Browser is running. As mentioned earlier, the Microsoft SQL Server Browser translates the instance name to a TCP/IP port in order for Ignition to connect to it.

1. Open the **SQL Server Configuration Manager** from **Start > All Programs > Microsoft SQL Server Version # > Configuration Tools > SQL Server Configuration Manager.**

2. Select the **SQL Server Services** section.

3. On the right, see all of the services installed. One of the services is **SQL Server Browser**. Make sure this service is in fact running. If the service is not running, right-click and select **Start**.

   Note: The service could be disabled, so you may need to double-click it to enable the service before starting it up.


Sql Server Configuration Manager
File
Action
View
Help
SQL Server Configuration Manager (Local)
SQL Server Services
SQL Server Network Configuration (32bit)
SQL Native Client 11.0 Configuration (32bit)
SQL Server Network Configuration
Protocols for SQLEXPRESS
SQL Native Client 11.0 Configuration

Name	State
SQL Server Browser	Running
SQL Server Agent (SQLEXPRESS)	Stopped
SQL Server (SQLEXPRESS)	Running

Different Ways of Connecting to SQL Server

Now that you have ensured that TCP/IP is enables and the Microsoft SQL Server Browser is running, you can connect to Microsoft SQL Server in four different ways (all using TCP/IP communication) as follows:

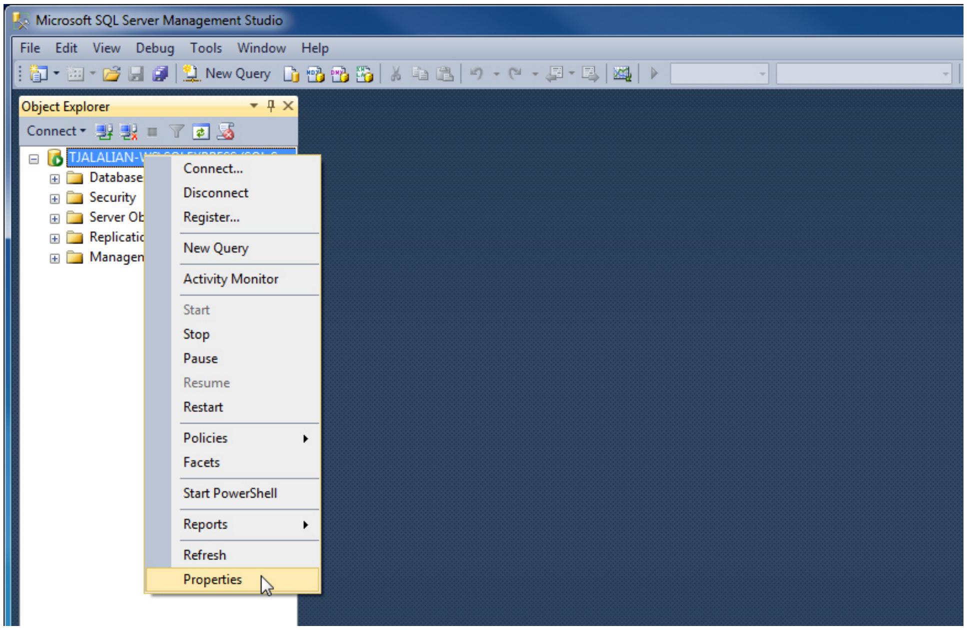
1. Connect using an Instance Name and SQL Authentication.
2. Connect using an Instance Name and Windows Authentication (this is the most common method).
3. Connect using a Port and SQL Authentication.
4. Connect using a Port and Windows Authentication.

Scenario 1: Connect By Using an Instance Name and SQL Authentication

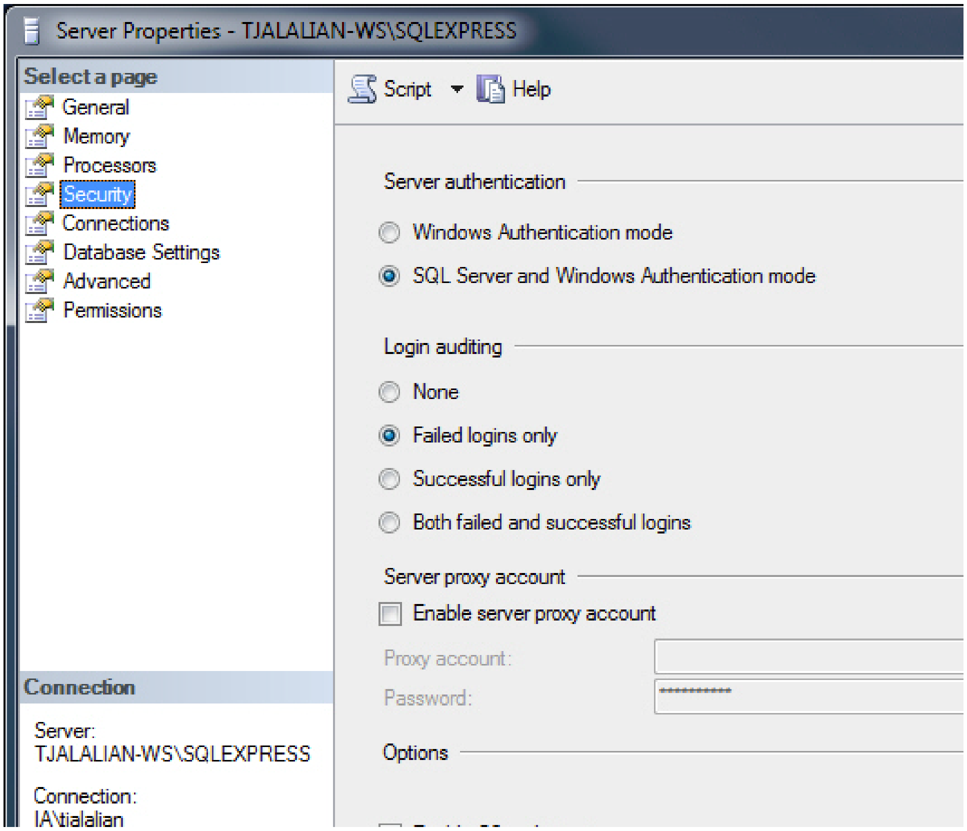
By default, Microsoft SQL Server only allows Windows authentication since it is more secure. But because we are using SQL authentication, we must enable Microsoft SQL to allow this type of authentication.

Enable SQL Authentication

1. Open the **Microsoft SQL Server Management Studio** window from **Start > All Programs > Microsoft SQL Server Version # > SQL Server Management Studio.**
   The window is displayed showing connections to your database.
2. Right-click the top-level database in the **Object Explorer** and select **Properties**.



3. From the Server Properties window, on the left side, select **Security**.



Server Properties - TJALALIAN-WS\SQLEXPRESS

Select a page
General
Memory
Processors
Security
Connections
Database Settings
Advanced
Permissions

Script
Help

Server authentication

○ Windows Authentication mode
● SQL Server and Windows Authentication mode

Login auditing

○ None
● Failed logins only
○ Successful logins only
○ Both failed and successful logins

Server proxy account

□ Enable server proxy account

Proxy account:
Password: ***********

Options

Connection

Server:
TJALALIAN-WS\SQLEXPRESS

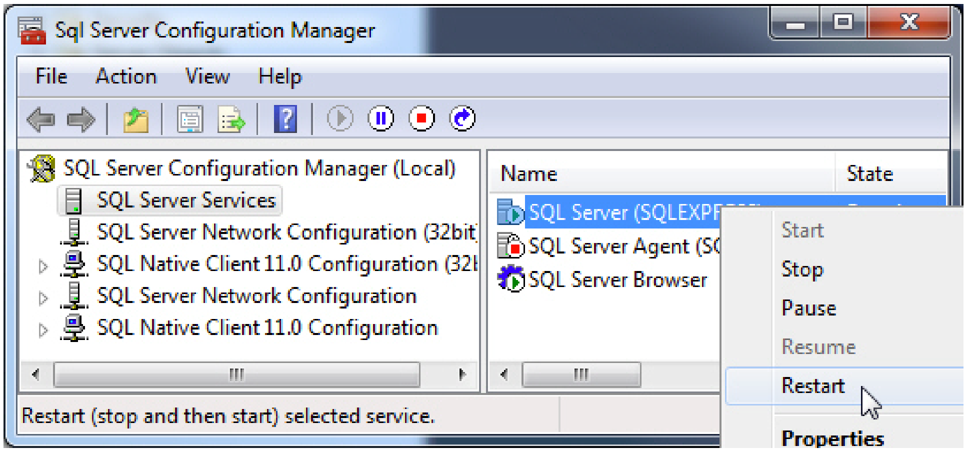
Connection:
IA\tjalalian

4. Verify that **SQL Server and Windows Authentication** mode is selected.
If not, select it and click **OK**.

Now you need to restart the **SQL Server Windows** service so that this setting takes effect.

5. Open the **SQL Server Configuration Manager** at
Start > All Programs > Microsoft SQL Server Version # > Configuration Tools > SQL Server Configuration Manager.

6. Select the **SQL Server Services** section and restart the **SQL Server (Instance Name)** item.

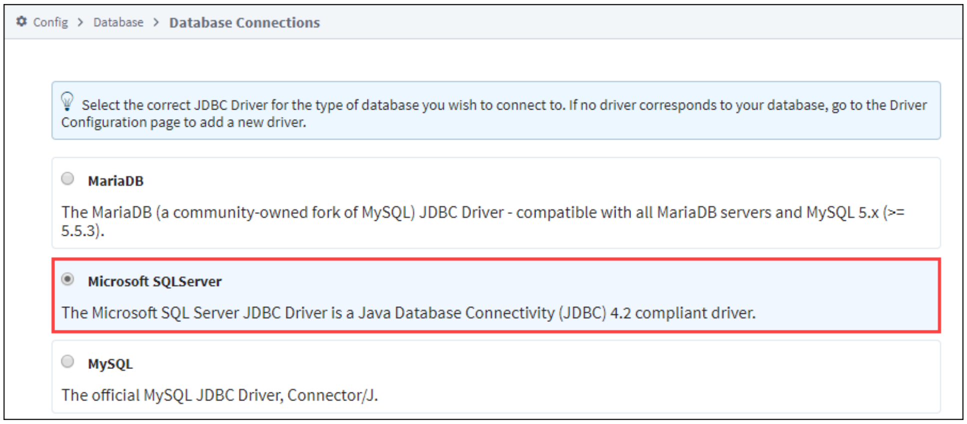

Sql Server Configuration Manager
File
Action
View
Help
SQL Server Configuration Manager (Local)
SQL Server Services
SQL Server Network Configuration (32bit
SQL Native Client 11.0 Configuration (32
SQL Server Network Configuration
SQL Native Client 11.0 Configuration
Name
State
SQL Server (SQLEXPR
SQL Server Agent (SQ
SQL Server Browser
Start
Stop
Pause
Resume
Restart
Properties

Restart (stop and then start) selected service.

Now that Microsoft SQL Server accepts SQL authentication, we can configure Ignition.

Configure the Database Connection in Ignition

1. Go to and login to the Ignition Gateway Config page from your webbrowser at **http://hostname:8088/main/web/config/**
2. Select **Databases > Connections** from the menu.
3. Click on **Create new Database Connection**.
4. Select **Microsoft SQL Server JDBC Driver** and click **Next**.



Select the correct JDBC Driver for the type of database you wish to connect to. If no driver corresponds to your database, go to the Driver Configuration page to add a new driver.

○ **MariaDB**

The MariaDB (a community-owned fork of MySQL) JDBC Driver - compatible with all MariaDB servers and MySQL 5.x (>= 5.5.3).

● **Microsoft SQLServer**

The Microsoft SQL Server JDBC Driver is a Java Database Connectivity (JDBC) 4.2 compliant driver.

○ **MySQL**

The official MySQL JDBC Driver, Connector/J.

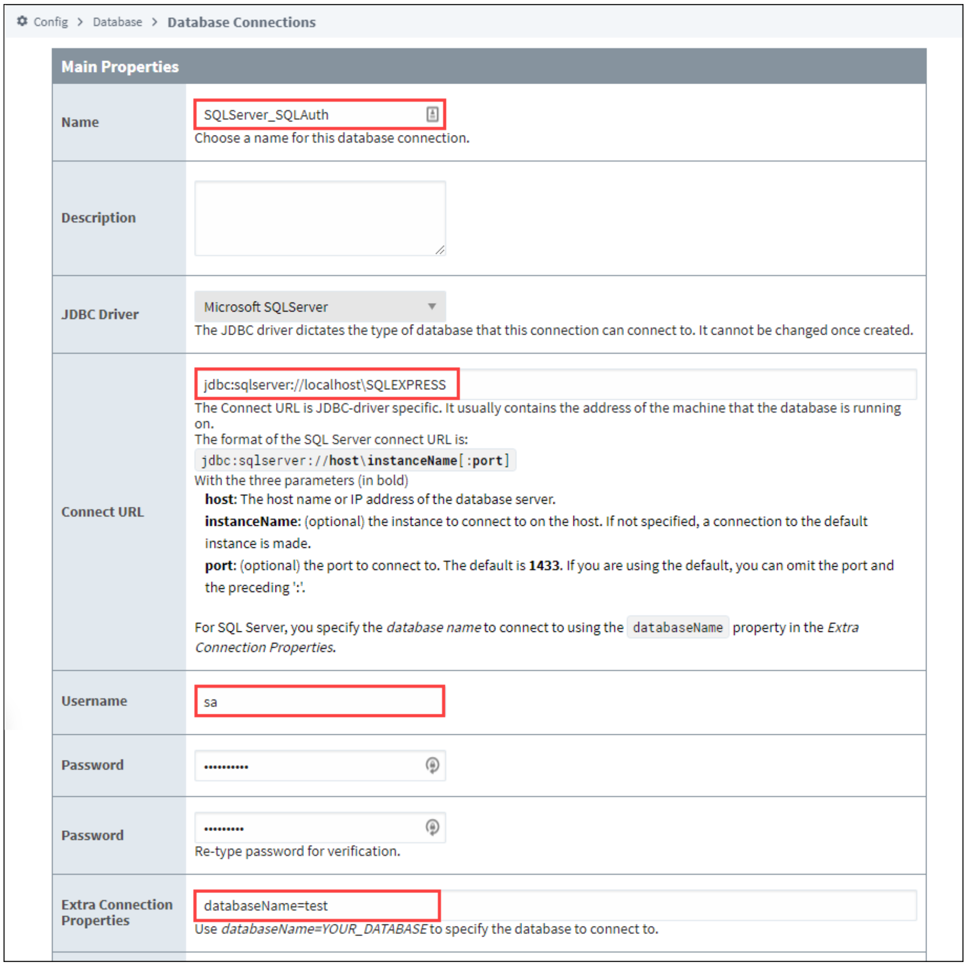
5. In the **New Database Connection** window, enter the following information:

   Name: **SQLServer_SQLAuth** (no spaces)

   Connect URL: **jdbc:sqlserver://Hostname\InstanceName**

   where **Hostname** is your databases IP address or hostname and **InstanceName** is your databases instance name, for example:
   jdbc:sqlserver://localhost\SQLEXPRESS
   jdbc:sqlserver://10.10.1.5\MSSQLSERVER

6. Set the username and password to a valid SQL authentication user. For example, **sa** is the default administrator account you can use.



Config > Database > **Database Connections**

Main Properties

Name	SQLServer_SQLAuth Choose a name for this database connection.
Description	
JDBC Driver	Microsoft SQLServer The JDBC driver dictates the type of database that this connection can connect to. It cannot be changed once created.
Connect URL	jdbc:sqlserver://localhost\SQLEXPRESS The Connect URL is JDBC-driver specific. It usually contains the address of the machine that the database is running on. The format of the SQL Server connect URL is: `jdbc:sqlserver://host\instanceName[:port]` With the three parameters (in bold) **host**: The host name or IP address of the database server. **instanceName**: (optional) the instance to connect to on the host. If not specified, a connection to the default instance is made. **port**: (optional) the port to connect to. The default is **1433**. If you are using the default, you can omit the port and the preceding ':'. For SQL Server, you specify the *database name* to connect to using the `databaseName` property in the *Extra Connection Properties.*
Username	sa
Password	••••••••••
Password	••••••••• Re-type password for verification.
Extra Connection Properties	databaseName=test Use *databaseName=YOUR_DATABASE* to specify the database to connect to.

7. To add your own user account, open the SQL Server Management Studio and expand the **Security > Logins** folder.
   You will see all the current logins including **sa** and you can add a new login.

   a. To add a new login, right-click on the **Logins** folder and click **New Login...**.
      The **Login** window is displayed.

   b. Choose the **SQL Server authentication** mode and type in a Login name and password.

   > ⚠ You will also have to add permissions to your database by mapping db_datareader and `db_datawriter` to the new user in the User Mapping section of the Login window. If you want Ignition to be able to create tables (ie: for Tag History), you also need to give table creation access such as db_owner.

8. Go back to the **New Database Connection** page in the Gateway, enter the name of your database, for example, in the **Extra Connection Properties** enter: **databaseName=test** (replace **test** with your database name, not the instance name).

9. Click **Create New Database Connection**.
   The Database Connection page is displayed showing the Status as **Valid** after a couple of seconds. If the connection is **Faulted**, click on the Database Connection Status link to find out why. Typically, the username/password is incorrect or the user doesn't have the right permissions.

Scenario 2: Connect By Using Instance Name and Windows Authentication

In Windows authentication mode, the username and password used to connect comes from the Ignition Windows Service logon. By default, the Ignition Windows Service is set to local system account which usually doesn't have privileges to connect.

Set Up the Service to Use Windows Authentication

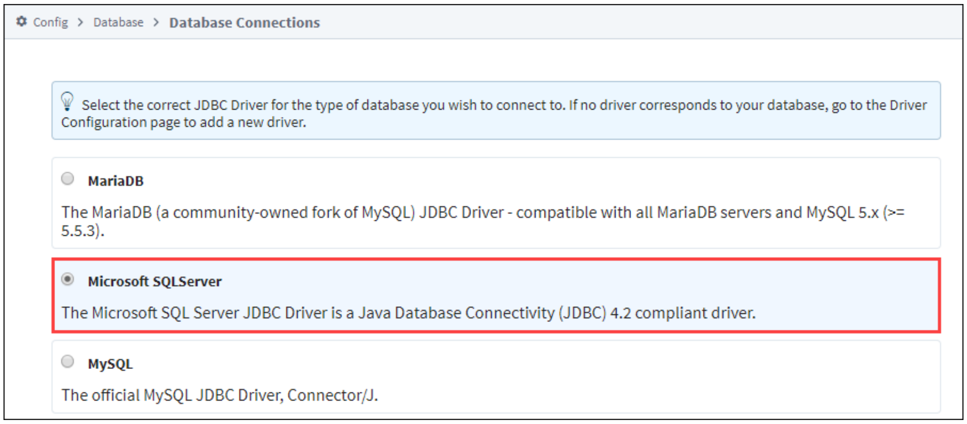
1. Download a copy of the SQL Server JDBC driver. Specifically, download a ZIP or tar.gz file (NOT an installer), as you will need to extract a specific file and relocate it to the gateway's installation directory. The exact version required depends on the version of Java your gateway is using. Ignition 8.0 uses uses Java 11, but future updates could change the java version. Regardless, it's recommended you use the most recent driver available for your system.
2. Locate the `DLL` file from the correct architecture folder ("x64" for 64-bit JDBC) inside of the `enu/auth` folders in the zip file.
3. Copy the DLL file to the **lib** folder in your install directory. If you have the default install directory, it's in the following location:
   C:\Program Files\Inductive Automation\Ignition\lib\

> **Note:** Older version of the JDBC driver (such as version 7.2.1) needed to be renamed to `sqljdbc_auth.dll` before Ignition could utilize the file. However, in more modern version of the driver, this is no longer the case.

4. The account used to connect will be the account that Ignition is running under in the services menu. To setup Ignition to logon using the right Windows account, open the **Services Control Panel** from Start > Control Panel > Administrative Tools > Services
5. Right-click the **Ignition** service and choose **Properties**.
6. Select the **Log On** tab.
7. Choose the **This account** radio button and enter in your Windows username and password.
8. Click **OK** to save.
9. Now restart the Ignition service to make this change take effect. Click the **Action > Restart** button in the menubar to restart the Ignition service (or your can stop and start from the right-click menu).

Configure the Database Connection in Ignition

1. Go to and login to the Ignition Gateway Config page from your webbrowser at http://hostname:8088/main/web/config/
2. Select **Databases > Connections** from the menu.
3. Click on **Create new Database Connection**.
4. Select **Microsoft SQL Serve**r JDBC Driver and click **Next**.



Config > Database > **Database Connections**

Select the correct JDBC Driver for the type of database you wish to connect to. If no driver corresponds to your database, go to the Driver Configuration page to add a new driver.

○ **MariaDB**

The MariaDB (a community-owned fork of MySQL) JDBC Driver - compatible with all MariaDB servers and MySQL 5.x (>= 5.5.3).

● **Microsoft SQLServer**

The Microsoft SQL Server JDBC Driver is a Java Database Connectivity (JDBC) 4.2 compliant driver.

○ **MySQL**

The official MySQL JDBC Driver, Connector/J.

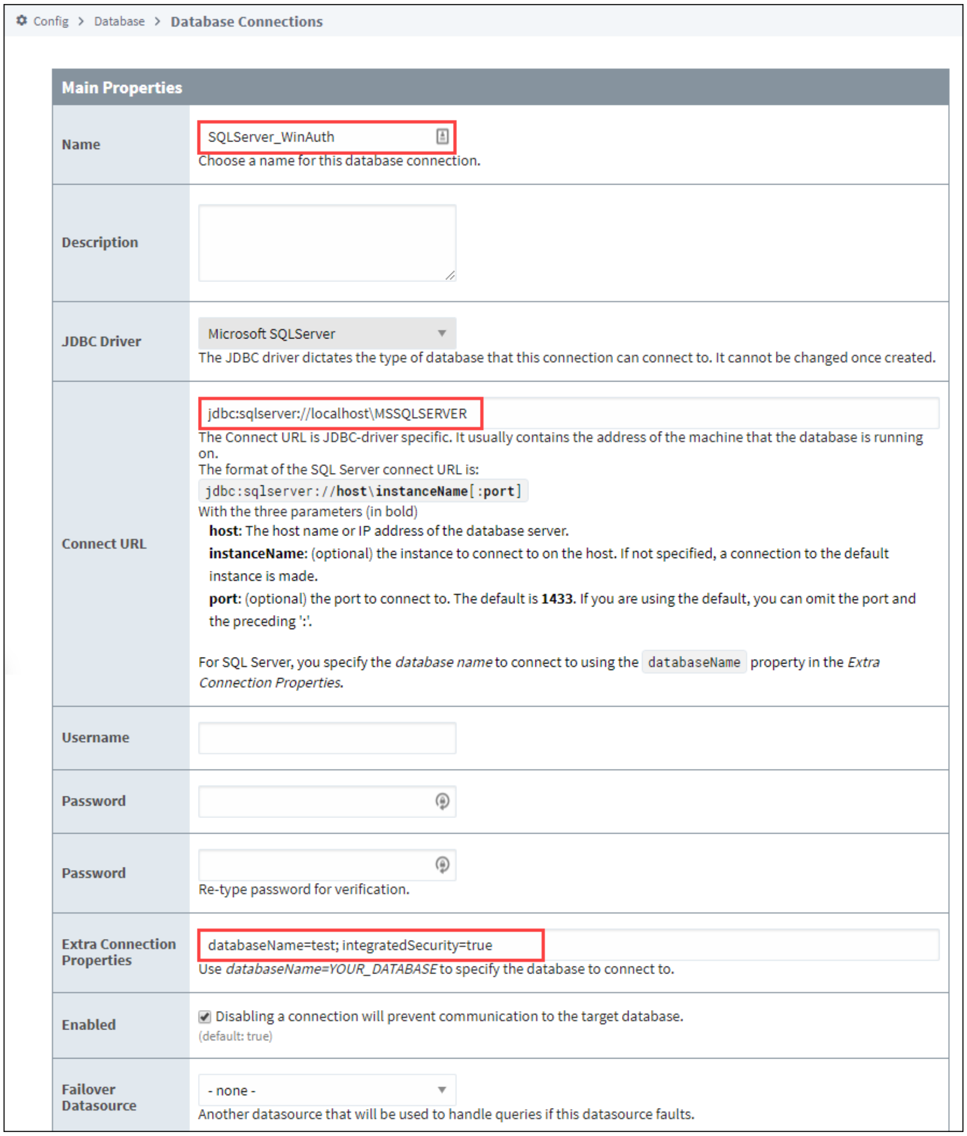
5. On the **New Database Connection** page, enter the following information:
   Name: **SQLServer_WinAuth** (no spaces)

   Connect URL: **jdbc:sqlserver://Hostname\InstanceName**

   where **Hostname** is your databases IP address or hostname and **InstanceName** is your databases instance name, for example:
   jdbc:sqlserver://localhost\SQLEXPRESS
   jdbc:sqlserver://10.10.1.5\MSSQLSERVER

   username: leave blank
   password: leave blank

   Extra Connection Properties:
   databaseName=test; integratedSecurity=true; (replace test with your database name)



Config > Database > **Database Connections**

Main Properties

Name	SQLServer_WinAuth Choose a name for this database connection.
Description	
JDBC Driver	Microsoft SQLServer The JDBC driver dictates the type of database that this connection can connect to. It cannot be changed once created.
Connect URL	jdbc:sqlserver://localhost\MSSQLSERVER The Connect URL is JDBC-driver specific. It usually contains the address of the machine that the database is running on. The format of the SQL Server connect URL is: `jdbc:sqlserver://host\instanceName[:port]` With the three parameters (in bold) **host**: The host name or IP address of the database server. **instanceName**: (optional) the instance to connect to on the host. If not specified, a connection to the default instance is made. **port**: (optional) the port to connect to. The default is **1433**. If you are using the default, you can omit the port and the preceding ':'. For SQL Server, you specify the *database name* to connect to using the `databaseName` property in the *Extra Connection Properties*.
Username	
Password	
Password	Re-type password for verification.
Extra Connection Properties	databaseName=test; integratedSecurity=true Use *databaseName=YOUR_DATABASE* to specify the database to connect to.
Enabled	☑ Disabling a connection will prevent communication to the target database. (default: true)
Failover Datasource	- none - Another datasource that will be used to handle queries if this datasource faults.

6. Click on **Create New Database Connection**.
   The Status should be Valid after a couple of seconds. Again, if the connection is Faulted, click the Database Connection Status link to find out why.

Scenario 3: Connect By Using Port and SQL Authentication

1. Connecting by using a port and SQL authentication is just like scenario 1 above except you specify a port instead of the instance name in the **New Database Connection** page.
2. Enter the following:

   Connect URL: **jdbc:sqlserver://Hostname:Port**

   where **Hostname** is your databases IP address or hostname and **Port** is your databases TCP/IP port (SQLSERVER default port is 1433), for example:
   jdbc:sqlserver://localhost:1433
   jdbc:sqlserver://10.10.1.5:1433

Scenario 4: Connect By Using Port and Windows Authentication

Connecting by using a port and Windows authentication is just like scenario 2 above except you specify a port instead of the instance name in the **New Database Connection** page. Don't forget to download the `sqljdbc_auth.dll` file if you need it.

Enter the following:

Connect URL: **jdbc:sqlserver://Hostname:Port**

where **Hostname** is your databases IP address or hostname and **Port** is your databases TCP/IP port (SQLSERVER default port is 1433), for example:
jdbc:sqlserver://localhost:1433
jdbc:sqlserver://10.10.1.5:1433

Troubleshooting

TCP/IP Communication Not Enabled

SQL Server requires that you explicitly turn on TCP connectivity. To do this, use the SQL Server Configuration Manager, located in the **Start** menu under **Microsoft SQL Server > Configuration Tools**. Under **SQL Server Network Configuration**, select your instance, and then enable TCP/IP in the panel to the right. You need to restart the server for the change to take affect.

Window Firewall

When connecting remotely, make sure that Windows Firewall is disabled, or set up to allow the necessary ports. Normally ports 1434 and 1433 must be open for TCP traffic, but other ports may be required based on configuration.

SQL Server Browser Process Not Running

To connect to a named instance, the **SQL Server Browser** service must be running. It is occasionally disabled by default, so you need to verify that the service is not only running, but set to
start automatically on bootup. The service can be found in the Windows Service Manager (**Control Panel > Administrative Tools > Services**).

Mixed Mode Authentication Not Enabled

Unless selected during setup, **mixed mode** or **SQL authentication** is not enabled by default. This mode of authentication is the **username /password** scheme that most users are used to. When not enabled, SQL Server only allows connections using Windows Authentication. Due to the ease of using SQL Authentication over Windows Authentication, we recommend enabling this option and defining a user account for Ignition.

1. To enable this, open the SQL Server Management Studio.
2. Connect to the server.
3. Right click on the instance and select **Properties**.
4. Under **Security**, select **SQL Sever and Windows Authentication mode**.

JDBC Drivers and Translators

In some cases, you may need to add your own JDBC Driver, or configure a Translator. More information on configuring these can be found on the JDBC Drivers and Translators page. However, you may need to check the JDBC driver's documentation for information on how to configure them.

Related Topics ...

- Connecting to Oracle Express
- Store and Forward
- OPC UA
- Designer

Connecting to Oracle Express

This page documents how to configure a database connection to an Oracle Express instance.

Oracle User Grants

When using an Oracle Express database connection, it is required to provide user credentials that have grants for "CREATE TRIGGER" and "CREATE SEQUENCE". Some of Ignition's subsystems, such as the Tag Historian, will fail to work properly if the user defined in the Database Connection does not have these grants. In addition, any manual queries (for example, those called by system.db.runNamedQuery) that need to insert records or create sequences may fail if the gateway does not have the grants.

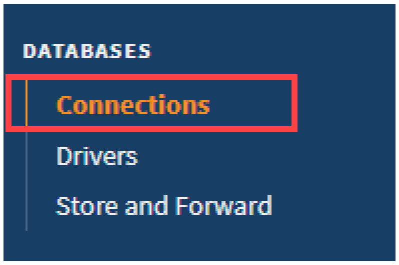
On this page ...

Connecting to Oracle Express

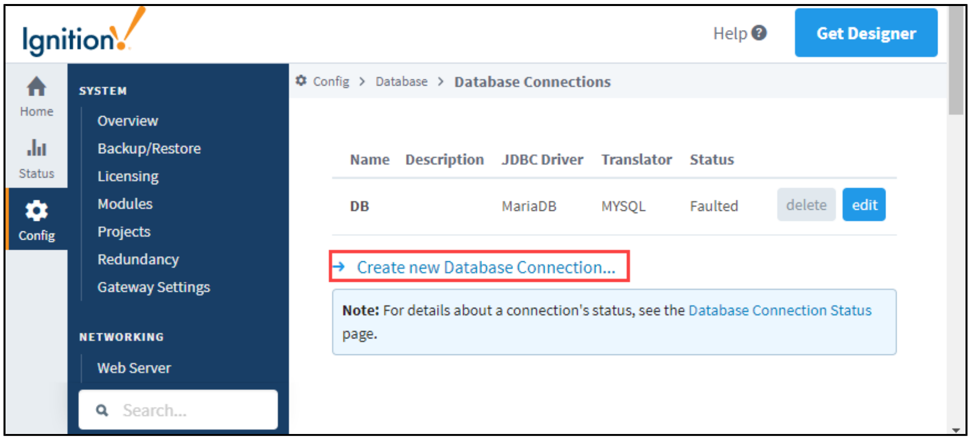
Watch the Video

Connect Ignition to the Oracle Express Database

1. On the Gateway Webpage, go to the **Config** section.
2. Scroll down to **Databases > Connections**.



3. The Database Connections page is displayed. Click on **Create new Database Connection...**.

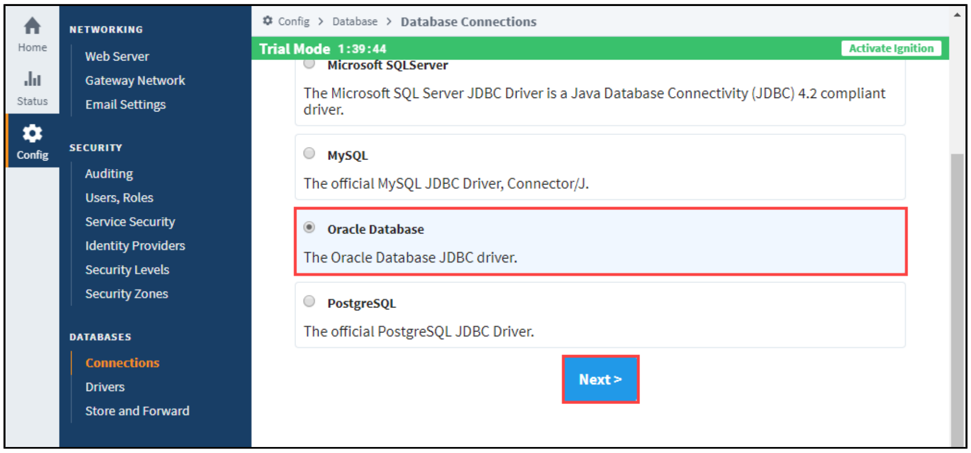


Name	Description	JDBC Driver	Translator	Status
DB		MariaDB	MYSQL	Faulted

→ Create new Database Connection...

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

4. Select the **Oracle JDBC Driver** and click **Next**.



○ Microsoft SQLServer

The Microsoft SQL Server JDBC Driver is a Java Database Connectivity (JDBC) 4.2 compliant driver.

○ MySQL

The official MySQL JDBC Driver, Connector/J.

● Oracle Database

The Oracle Database JDBC driver.

○ PostgreSQL

The official PostgreSQL JDBC Driver.

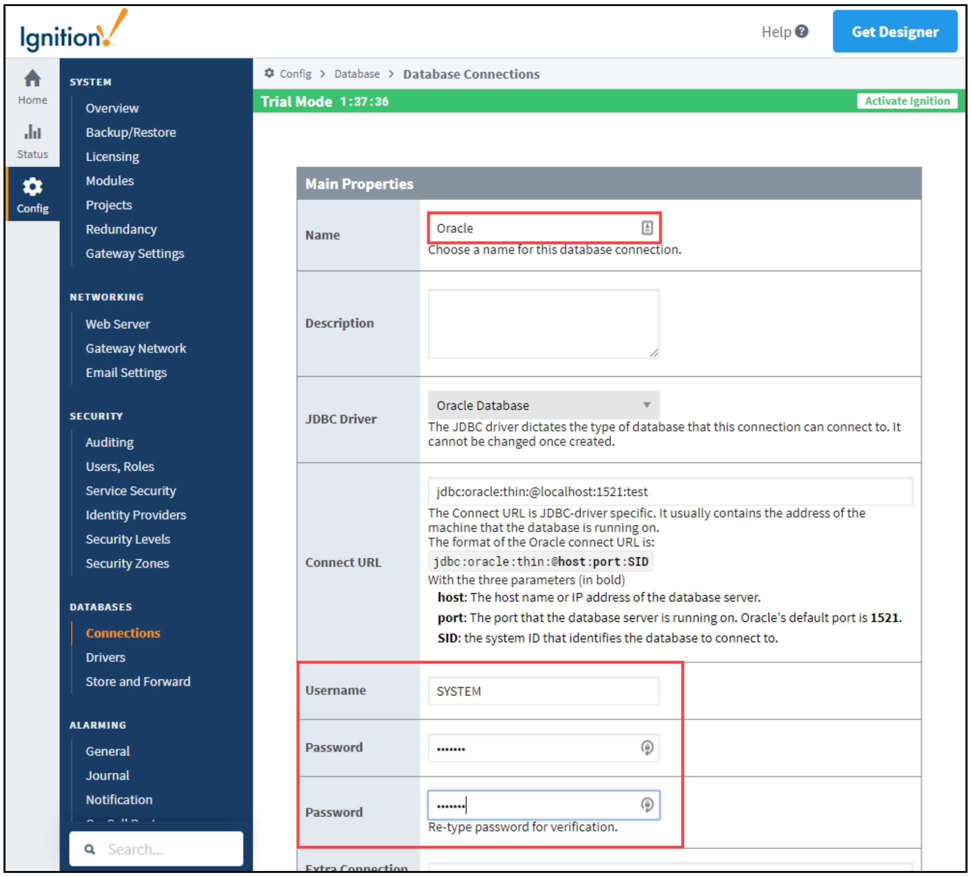
5. Enter the following information:

Property Name	How should it be configured
Name	The name of the database connection. Other features in Ignition will reference this connection by the name specified here.

Connect URL	A URL that describes where on the network the database is, as well as information about which schema to connect to. The initial value should look something like the following
	`jdbc:oracle:thin:@localhost:1521:XE`
	However, you may need to make some changes. If we examine the key pieces of the URL, it would look like this:
	`jdbc:oracle:thin://1:2:3`
	Where:
	1 = The IP Address or hostname of the computer/sever that the database is installed on.
	2 = The port that the database is running on. The default is 1521, but this could have been changed during installation, or sometime afterwards. When in doubt, ask the person that installed the database.
	3 = The System ID for the database. More information on the System ID can be found in Oracle's documentation.

Username	The username that the Gateway will use to connect to the database.
Password	The password for the user specified under the Username property.

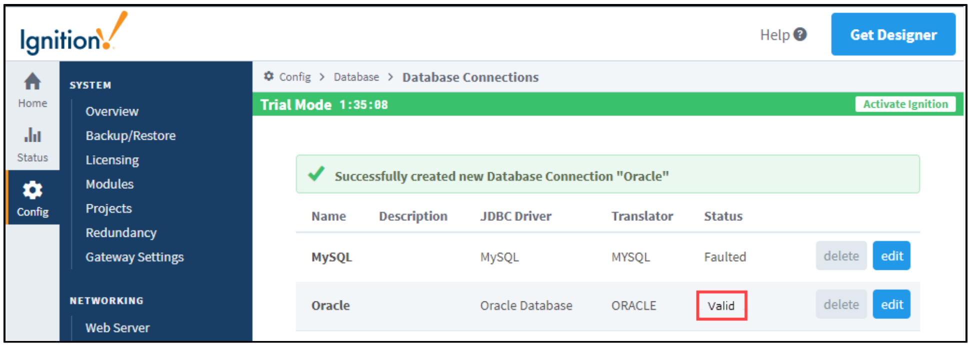
For our example, we entered information relevant to our installation. The values seen below may differ from each installation.



Main Properties

Name	Oracle Choose a name for this database connection.
Description	
JDBC Driver	Oracle Database The JDBC driver dictates the type of database that this connection can connect to. It cannot be changed once created.
Connect URL	jdbc:oracle:thin:@localhost:1521:test The Connect URL is JDBC-driver specific. It usually contains the address of the machine that the database is running on. The format of the Oracle connect URL is: `jdbc:oracle:thin:@host:port:SID` With the three parameters (in bold) **host:** The host name or IP address of the database server. **port:** The port that the database server is running on. Oracle's default port is **1521**. **SID:** the system ID that identifies the database to connect to.
Username	SYSTEM
Password	•••••••
Password	••••••• Re-type password for verification.
Extra Connection	

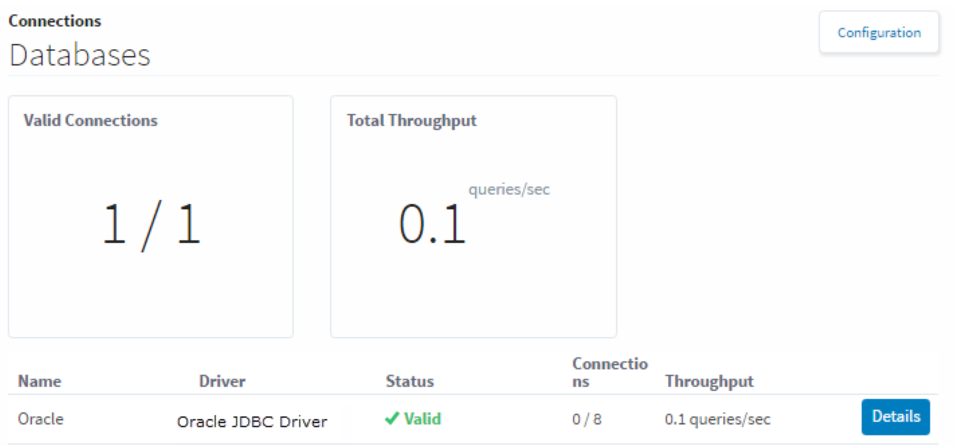
6. Click **Create New Database Connection** at the bottom of the form.
7. The Database Connections page is displayed showing the **Status** of your connection as **Valid**.



✓ Successfully created new Database Connection "Oracle"

Name	Description	JDBC Driver	Translator	Status	
MySQL		MySQL	MYSQL	Faulted	delete edit
Oracle		Oracle Database	ORACLE	Valid	delete edit

8. To display the details about the status of your database connection, see the **Note** on the above window and click on the **Database Connection Status** link.



Connections

Databases

Valid Connections

1 / 1

Total Throughput

0.1 queries/sec

Name	Driver	Status	Connections	Throughput
Oracle	Oracle JDBC Driver	✓ Valid	0 / 8	0.1 queries/sec

JDBC Drivers and Translators

In some cases, you may need to add your own JDBC Driver, or configure a Translator. More information on configuring these can be found on the JDBC Drivers and Translators page. However, you may need to check the JDBC driver's documentation for information on how to configure them.

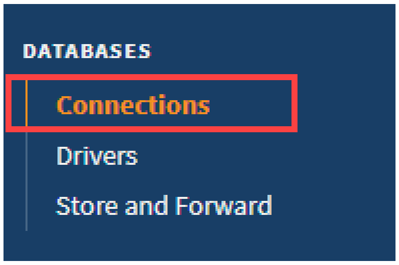
Related Topics ...

- Store and Forward
- OPC UA
- Designer

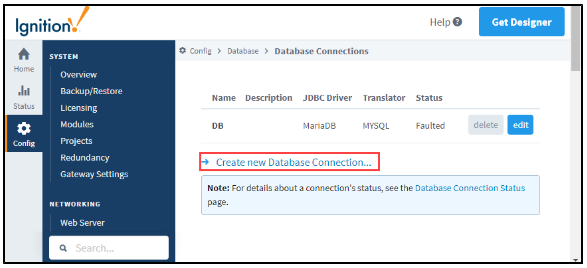
Connecting to PostgreSQL

Connect Ignition to the PostreSQL Database

1. On the Gateway Webpage, go to the **Config** section.
2. Scroll down to **Databases > Connections**.



3. The Database Connections page is displayed. Click on **Create new Database Connection...**.



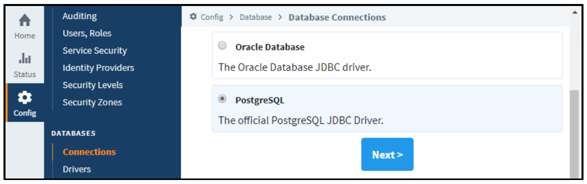
Config > Database > Database Connections

Name	Description	JDBC Driver	Translator	Status
DB		MariaDB	MYSQL	Faulted

→ Create new Database Connection...

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

4. Select the **PostgreSQL JDBC Driver** and click **Next**.



Config > Database > Database Connections

○ **Oracle Database**
The Oracle Database JDBC driver.

● **PostgreSQL**
The official PostgreSQL JDBC Driver.

5. Next, you'll need to enter information that will allow the Gateway to connect to a Postgres instance. In most cases, the following properties should be updated:

Property Name	How should it be configured
Name	The name of the database connection. Other features in Ignition will reference this connection by the name specified here.

Connect URL	A URL that describes where on the network the database is, as well as information about which schema to connect to. The initial value should look something like the following
	`jdbc:postgresql://localhost:5432/schema`
	However, you may need to make some changes. If we examine the key pieces of the URL, it would look like this:
	`jdbc:postgresql://1:2/3`
	Where:

INDUCTIVE UNIVERSITY

Connecting to PostgreSQL

Watch the Video

	1 = The IP Address or hostname of the computer/sever that Postgres is installed on 2 = The port that Postgres is running on. The default is 5432, but this could have been changed during installation, or sometime afterwards. When in doubt, ask the person that installed the database. 3 = The schema name that this connection will provide access to.
Username	The username that the Gateway will use to connect to the database.
Password	The password that the Gateway will use to connect to the database.

6. Click **Create New Database Connection** at the bottom of the form.
7. The Database Connections page is displayed showing the **Status** of your connection as **Valid**.
8. To display the details about the status of your database connection, see the **Note** on the above window and click on the **Database Connection Status** link.

JDBC Drivers and Translators

In some cases, you may need to add your own JDBC Driver, or configure a Translator. More information on configuring these can be found on the JDBC Drivers and Translators page. However, you may need to check the JDBC driver's documentation for information on how to configure them.

Related Topics ...

- Store and Forward
- OPC UA
- Designer

Connecting to SQLite

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

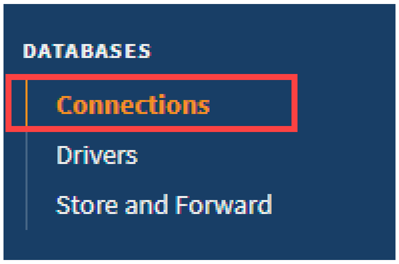
An Ignition Gateway can create a SQLite database, allowing for data collection without installing a separate SQL database. This is ideal for small scale applications, as well as testing and demonstrations.

SQLite connections offer convenience, but Relational Database Management Systems (RDBMS) generally offer better performance, especially so in cases where queries are frequently executed such as systems utilizing a historian system. As a result, the SQLite connection is **not** recommended for production systems as a historian database. Learn more about appropriate uses of SQLite in their official documentation.

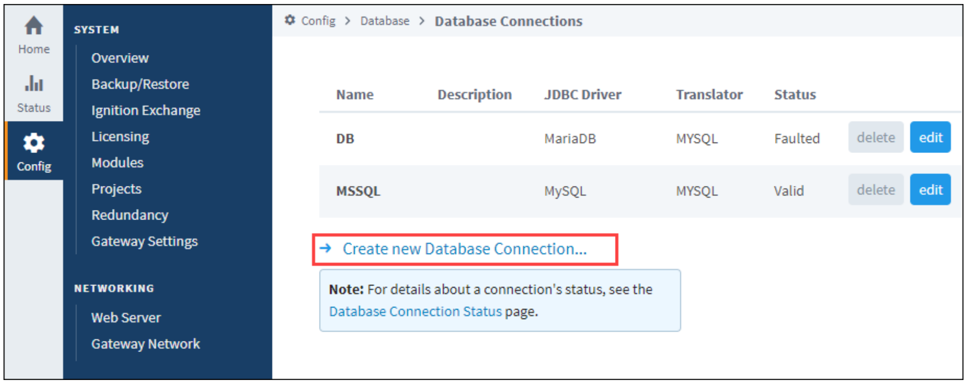
Unlike other database connections, SQLite databases only support a single connection at a time. Configuring this type of Database Connection means the Gateway will constantly be connected to the SQLite database, preventing other systems from connecting.

Connect Ignition to SQLite Database

1. On the Gateway Webpage, go to the **Config** section.
2. Scroll down to **Databases > Connections**.



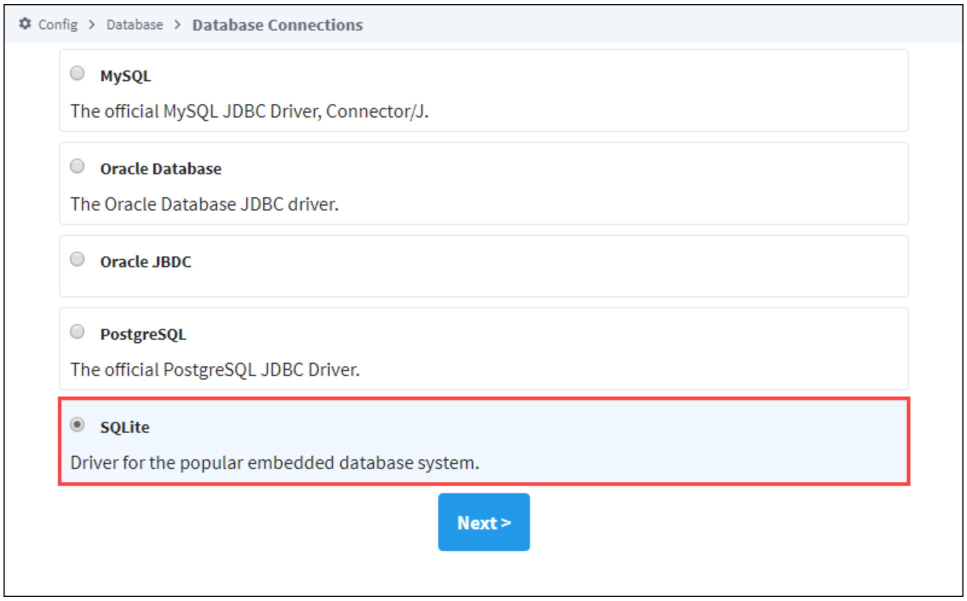
3. The Database Connections page is displayed. Click on **Create new Database Connection...**.



Name	Description	JDBC Driver	Translator	Status
DB		MariaDB	MYSQL	Faulted
MSSQL		MySQL	MYSQL	Valid

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

4. Select the **SQLite** driver and click **Next**.



Config > Database > **Database Connections**

○ **MySQL**
The official MySQL JDBC Driver, Connector/J.

○ **Oracle Database**
The Oracle Database JDBC driver.

○ **Oracle JBDC**

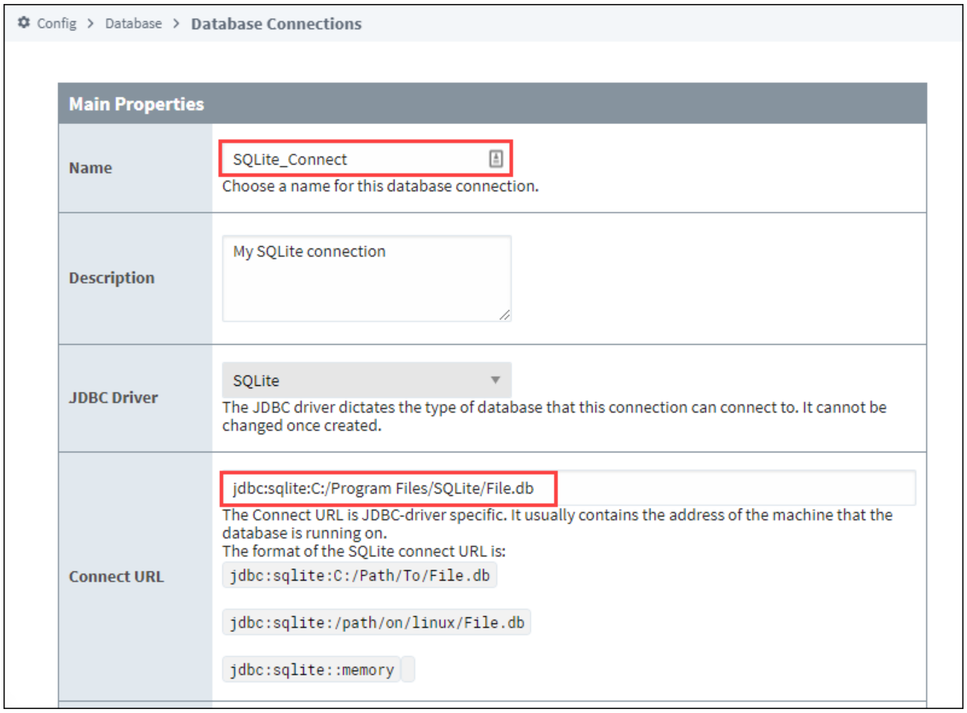
○ **PostgreSQL**
The official PostgreSQL JDBC Driver.

● **SQLite**
Driver for the popular embedded database system.

5. On the **New Database Connection** page, enter a name for your connection. We used **SQLite_Connect**.
6. Next, enter the connect URL for the database. For SQLite this setting must lead to a location on a filesystem. In our example here we used **jdbc:sqlite:C:Program Files/SQLite/File.db.** This path ultimately must lead to a local directory, or a locally mapped drive.

> ⚠ The JDBC Driver for SQLite will attempt to create a database if one does not already exist.

> ⚠ For Windows machines, either forward / or backslashes\ can be used.



Config > Database > **Database Connections**

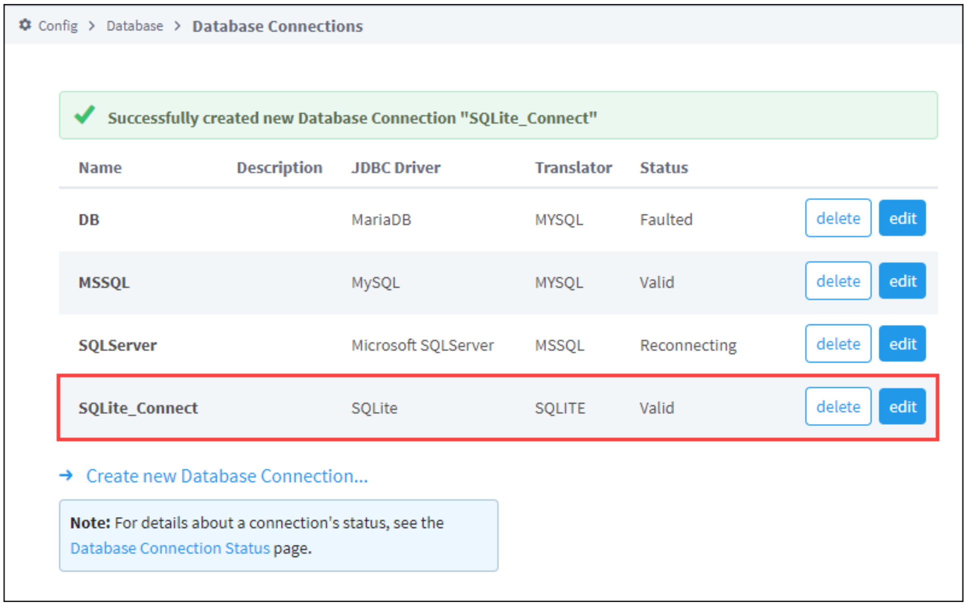
Main Properties

Name	SQLite_Connect Choose a name for this database connection.
Description	My SQLite connection
JDBC Driver	SQLite The JDBC driver dictates the type of database that this connection can connect to. It cannot be changed once created.

Connect URL	jdbc:sqlite:C:/Program Files/SQLite/File.db
	The Connect URL is JDBC-driver specific. It usually contains the address of the machine that the database is running on.
	The format of the SQLite connect URL is:
	`jdbc:sqlite:C:/Path/To/File.db`
	`jdbc:sqlite:/path/on/linux/File.db`
	`jdbc:sqlite::memory`

7. Click on **Create New Database Connection** at the bottom of the form.

   Your connection is now created and the Database Connections page is displayed showing the **Status** of your connection as **Valid**.



✓ Successfully created new Database Connection "SQLite_Connect"

Name	Description	JDBC Driver	Translator	Status
DB		MariaDB	MYSQL	Faulted
MSSQL		MySQL	MYSQL	Valid
SQLServer		Microsoft SQLServer	MSSQL	Reconnecting
SQLite_Connect		SQLite	SQLITE	Valid

→ Create new Database Connection...

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

Related Topics ...

- Connecting to Microsoft SQL Server Express
- Store and Forward
- OPC UA
- Designer

JDBC Drivers and Translators

Overview

In most cases, the default JDBC drivers and Translator settings in Ignition will not need to be modified. However, there are cases where drivers or translators may need to be modified.

JDBC Drivers and Ignition Upgrades

When upgrading Ignition, JDBC drivers are **not** modified during the upgrade process. This is to prevent database connection issues on upgrade: you should only need to upgrade your JDBC drivers when the database is updated, not when Ignition is upgraded. Only new Ignition installations will always use the JDBC drivers that are included with the installer.

Running the installer to upgrade a preexisting Ignition installation will not modify the installed JDBC drivers, even if newer drivers are included in the installer. In addition, restoring a gateway backup from an older version will replace any new versions of the drivers with the versions from the backup. In this case, you will need to manually update the JDBC drivers.

Commonly, you will have to download the official JAR file from the creator's website. We have a few links here to make it easy to find. Please note that these links could break at any time without us knowing.

On this page ...

Database	Download Link	Notes
MySQL	https://dev.mysql.com/downloads/connector/j/	1. Select Your Operating system (or **Platform Independent** if you are on Windows). 2. After the file has been downloaded, unzip the the archive. On Windows you can right-click and select the **Extract All** option. 3. The location of the JAR we need should be in the extracted folder under **mysql-connector-java-X.X.XX** where the Xs are the version number. You are looking for a file that is named like **mysql-connector-java-X.X.XX.jar**.
MSSQL	https://github.com/microsoft/mssql-jdbc/releases	1. You only need the .jar file, so you can select just the **mssql-jdbc-X.X.XX.jre11.jar** file. The Xs will be replaced with the version numbers. Ensure you download the **jre11** version of the JAR.
Oracle	https://www.oracle.com/technetwork/database/application-development/jdbc/downloads/index.html	1. You will need to create an Oracle account to download the JAR files. 2. You can select the Unzipped version of the newest JAR. 3. You are looking for a file that is named like **odbcX.jar**. Where X is the version number.

Translator and JDBC Driver Settings

In some cases, you may need to add your own JDBC Driver, or configure a Translator. We recommend checking the JDBC driver's documentation for information on how to configure them. Below are some recommended settings, but the vendor's documentation should always supersede any suggestions here.

MySQL Translator and JDBC Driver Settings

MySQL connections can often use the included MariaDB translator, but we recommend using a separate MySQL Translator.

Main Properties	
Name	MySQL
Create Table Syntax	CREATE TABLE {tablename} ({creationdef}{primarykeydef})
Create Sequence Syntax	<leave empty>
Create Trigger Syntax	<leave empty>
Create Index Syntax	CREATE INDEX {indexname} ON {tablename}({columnname})
Auto Increment Field Definition	{type} NOT NULL AUTO_INCREMENT
Alter Table Syntax	ALTER TABLE {tablename} {alterdef}
Add Column Syntax	ADD COLUMN {columnname} {type}
Primary Key Syntax	PRIMARY KEY ({columnname})
Limit Syntax	LIMIT {limit}
Limit Position	Back
Current Timestamp Query	SELECT CURRENT_TIMESTAMP
Column Quote Character	`
Supports Returning Auto-generated Keys?	True
Fetch Key Query	<leave empty>
Table List Filter	<leave empty>
Data Type Mapping	
Byte (I1)	int
Short (I2)	int
Integer (I4)	int
Long (I8)	bigint
Boolean	int
Datetime	datetime
Float (R4)	float(10)
Double (R8)	double
String	varchar(255)
Binary	varbinary
Long Text	text

Main Properties	
Name	MySQL
Description	The official MySQL JDBC Driver, Connector/J.
Classn	com.mysql.cj.jdbc.Driver

ame	
JAR File(s)	<Click on the **Choose File** button to select and upload the JAR(s). This is the part where you upload the JDBC driver. >
Driver Defaults & Instructions	
Driver Type	MySQL
URL Format	jdbc:mysql://localhost:3306/test
URL Instructions	<br/>The format of the MySQL connect URL is:<br><code>jdbc:mysql://<b>host</b>:<b>port</b>/<b>database</b></code><br>With the three parameters (in bold) <ul style="list-style-type:none;margin-left:10px;"><li><b>host</b>: The host name or IP address of the database server.</li><li><b>port</b>: The port that the database server is running on. MySQL default port is <b>3306</b>.</li><li><b>database</b>: The name of the logical database that you are connecting to on the MySQL server.</li></ul>
Default Connection Properties	zeroDateTimeBehavior=CONVERT_TO_NULL;connectTimeout=120000;socketTimeout=120000;useSSL=false;allowPublicKeyRetrieval=true;
Connection Properties Instructions	There is an extensive list of extra connection properties available for MySQL Connector/J. See <a href='the'>http://dev.mysql.com/doc/connectors/en/connector-j-reference-configuration-properties.html'>the documentation</a> for a table describing all connection properties.<br>A default <tt>serverTimezone</tt> value (taken from the gateway) will be appended to the connection string if one is not specified.
Default Validation Query	SELECT 1
SQL Language Compatibility	
Default Translator	MySQL

MSSQL Translator and JDBC Driver Settings

Main Properties	
Name	<Name of the Translator. The JDBC driver will reference the settings below by the name specified here>
Create Table Syntax	CREATE TABLE {tablename} ({creationdef}{primarykeydef})
Create Sequence Syntax	<Blank>
Create Trigger Syntax	<Blank>
Create Index Syntax	CREATE INDEX {indexname} ON {tablename}({columnname})
Auto Increment Field Definition	{type} IDENTITY(1,1)
Alter Table Syntax	ALTER TABLE {tablename} ADD {alterdef}
Add Column Syntax	{columnname} {type}
Primary Key Syntax	PRIMARY KEY CLUSTERED ({columnname})
Limit Syntax	TOP {limit}

Limit Position	Front
Current Timestamp Query	SELECT CURRENT_TIMESTAMP
Column Quote Character	"
Supports Returning Auto-generated Keys?	True
Fetch Key Query	<Blank>
Table List Filter	<Blank>
Data Type Mapping	
Byte (I1)	int
Short (I2)	int
Integer (I4)	int
Long (I8)	bigint
Boolean	int
Datetime	datetime
Float (R4)	float(10)
Double (R8)	double precision
String	varchar(255)
Binary	varbinary
Long Text	nvarchar(max)

Main Properties	
Name	<Name of the driver, as you would like it to appear on the gateway>
Description	<Enter a useful description you would like to see next to the driver>
Classname	com.microsoft.sqlserver.jdbc.SQLServerDriver
JAR File(s)	<Click on the **Choose File** button to select and upload the JAR(s). This is the part where you upload the JDBC driver. >
Driver Defaults & Instructions	
Driver Type	Microsoft SQL Server
URL Format	jdbc:sqlserver://localhost\SQLEXPRESS
URL Instructions	<br/>The format of the SQL Server connect URL is:<br/><code>jdbc:sqlserver://<b>host</b>\<b>instanceName</b>[:<b>port</b>]</code><br/>With the three parameters (in bold) <ul style="list-style-type:none;margin-left:10px;"><li><b>host</b>: The host name or IP address of the database server.</li><li><b>instanceName</b>: (optional) the instance to connect to on the host. If not specified, a connection to the default instance is made.</li><li><b>port</b>: (optional) the port to connect to. The default is <b>1433</b>. If you are using the default, you can omit the port and the preceding ':'.</li></ul><br/>For SQL Server, you specify the <i>database name</i> to connect to using the <code>databaseName</code> property in the <i>Extra Connection Properties</i>.
Default	databaseName=test

Connection Properties	
Connection Properties Instructions	Use <i>databaseName=YOUR_DATABASE</i> to specify the database to connect to.
Default Validation Query	SELECT 1
SQL Language Compatibility	
Default Translator	The Translator this driver should use. If you're adding a new Driver, then you may

Oracle Express Translator and JDBC Driver Settings

Main Properties	
Name	<Name of the Translator. The JDBC driver will reference the settings below by the name specified here>
Create Table Syntax	CREATE TABLE {tablename} ({creationdef}{primarykeydef})
Create Sequence Syntax	CREATE SEQUENCE {tablename}seq START WITH 1 INCREMENT BY 1
Create Trigger Syntax	CREATE TRIGGER {tablename}trig BEFORE INSERT ON {tablename} REFERENCING NEW AS NEW FOR EACH ROW BEGIN select {tablename}seq.nextval INTO :NEW.{columnname} FROM dual; END;
Create Index Syntax	CREATE INDEX {indexname} ON {tablename}({columnname})
Auto Increment Field Definition	{type} NOT NULL
Alter Table Syntax	ALTER TABLE {tablename} ADD ({alterdef})
Add Column Syntax	{columnname} {type}
Primary Key Syntax	PRIMARY KEY ({columnname})
Limit Syntax	rownum<={limit}
Limit Position	Where
Current Timestamp Query	SELECT CURRENT_TIMESTAMP FROM DUAL
Column Quote Character	"
Supports Returning Auto-generated Keys?	False

Fetch Key Query	SELECT {tablename}SEQ.CURRVAL FROM DUAL
Table List Filter	<leave empty>
Data Type Mapping	
Byte (I1)	int
Short (I2)	int
Integer (I4)	int
Long (I8)	int
Boolean	int
Datetime	timestamp
Float (R4)	float
Double (R8)	double precision
String	varchar2(255)
Binary	varbinary
Long Text	nclob

Main Properties	
Name	<Name of the driver, as you would like it to appear on the gateway>
Description	<Enter a useful description you would like to see next to the driver>
Classname	oracle.jdbc.driver.OracleDriver
JAR File(s)	<Click on the **Choose File** button to select and upload the JAR(s). This is the part where you upload the JDBC driver. >
Driver Defaults & Instructions	
Driver Type	Oracle
URL Format	jdbc:oracle:thin:@localhost:1521:test
URL Instructions	<br/>The format of the Oracle connect URL is:<br/><code>jdbc:oracle:thin:@<b>host</b>:<b>port</b>:<b>SID</b></code><br />With the three parameters (in bold) <ul style="list-style-type:none;margin-left:10px;"><li><b>host</b>: The host name or IP address of the database server.</li><li><b>port</b>: The port that the database server is running on. Oracle's default port is <b>1521</b>.</li><li><b>SID</b>: the system ID that identifies the database to connect to.</li></ul>
Default Connection Properties	<Blank>
Connection Properties Instructions	<Blank>
Default Validation Query	SELECT 1 FROM DUAL
SQL Language Compatibility	
Default	The Translator this driver should use. If you're adding a new Driver, then you may

Translator	

PostgreSQL Translator and JDBC Driver Settings

Main Properties	
Name	<Name of the Translator. The JDBC driver will reference the settings below by the name specified here>
Create Table Syntax	CREATE TABLE {tablename} ({creationdef}{primarykeydef})
Create Sequence Syntax	<Blank>
Create Trigger Syntax	<Blank>
Create Index Syntax	CREATE INDEX {indexname} ON {tablename}({columnname})
Auto Increment Field Definition	SERIAL NOT NULL
Alter Table Syntax	ALTER TABLE {tablename} {alterdef}
Add Column Syntax	ADD COLUMN {columnname} {type}
Primary Key Syntax	PRIMARY KEY ({columnname})
Limit Syntax	LIMIT {limit}
Limit Position	Back
Current Timestamp Query	SELECT CURRENT_TIMESTAMP
Column Quote Character	"
Supports Returning Auto-generated Keys?	True
Fetch Key Query	<Blank>
Table List Filter	<Blank>
Data Type Mapping	
Byte (I1)	int
Short (I2)	int
Integer (I4)	int
Long (I8)	bigint
Boolean	int
Datetime	timestamp
Float (R4)	float
Double (R8)	double precision
String	varchar(255)
Binary	bytea
Long Text	text

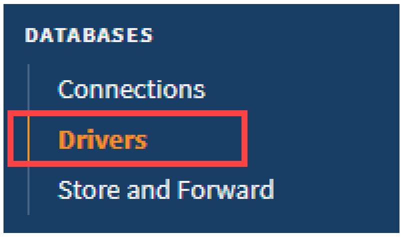
Main Properties	
Name	<Name of the driver, as you would like it to appear on the gateway>
Description	<Enter a useful description you would like to see next to the driver>
Class	org.postgresql.Driver

name	
JAR File(s)	<Click on the **Choose File** button to select and upload the JAR(s). This is the part where you upload the JDBC driver. >
Driver Defaults & Instructions	
Driver Type	PostgreSQL
URL Format	jdbc:postgresql://localhost:5432/test
URL Instructions	<br/>The format of the PostgreSQL connect URL is:<br/><code>jdbc:postgresql://<b>host</b>:<b>port</b>/<b>database</b></code><br/>With the three parameters (in bold) <ul style="list-style-type:none;margin-left:10px;"><li><b>host</b>: The host name or IP address of the database server.</li><li><b>port</b>: The port that the database server is running on. PostgreSQL default port is <b>5432</b>.</li><li><b>database</b>: The name of the logical database that you are connecting to on the PostgreSQL server.</li></ul>
Default Connection Properties	<Blank>
Connection Properties Instructions	No extra connection parameters are recommended for PostgreSQL. For possible parameter values, see the documentation at <a href='the'>http://jdbc.postgresql.org'>the PostgreSQL JDBC driver website</a>.
Default Validation Query	SELECT 1
SQL Language Compatibility	
Default Translator	The Translator this driver should use. If you're adding a new Driver, then you may

Adding a JDBC Driver

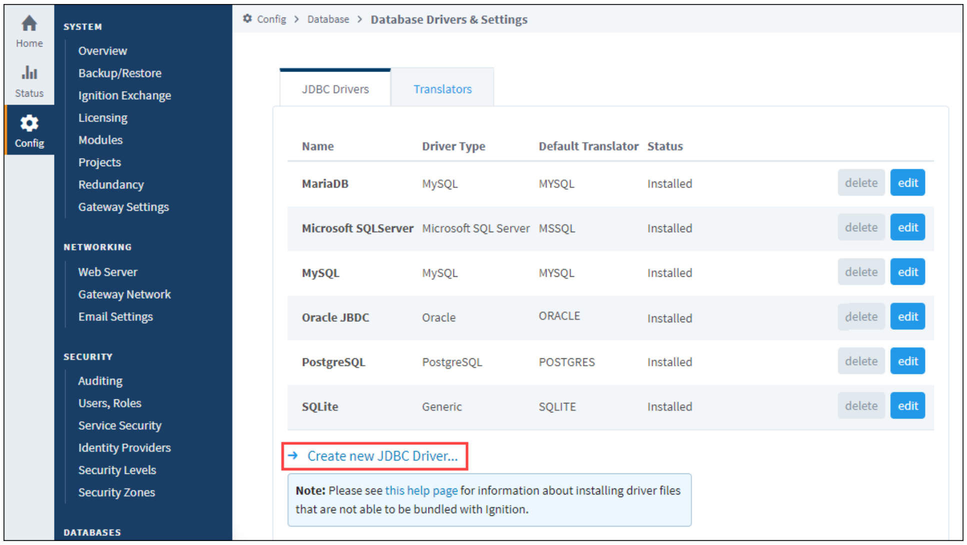
To add a new JDBC driver to Ignition, do the following steps:

1. On the Gateway Webpage **Config** section, click on **Databases > Drivers**.



The **Database Drivers & Settings** page is displayed.

2. Click on the **Create new JDBC Driver...** link at the bottom of the page.



Name	Driver Type	Default Translator	Status		
MariaDB	MySQL	MYSQL	Installed	delete	edit
Microsoft SQLServer	Microsoft SQL Server	MSSQL	Installed	delete	edit
MySQL	MySQL	MYSQL	Installed	delete	edit
Oracle JBDC	Oracle	ORACLE	Installed	delete	edit
PostgreSQL	PostgreSQL	POSTGRES	Installed	delete	edit
SQLite	Generic	SQLITE	Installed	delete	edit

→ Create new JDBC Driver...

Note: Please see this help page for information about installing driver files that are not able to be bundled with Ignition.

3. In the **Name** field, type the full name of the JDBC driver, see the manufacturer's documentation to get the name.

4. In the **JAR File(s)** field, specify the JAR file that contains the driver, as well as any other required JARs. If you do not have the JAR file needed, see above for download links.

5. Use the default settings for the following properties:

Driver Defaults and Instructions

Driver Type	Is the brand of database. This is used for optimizations in the Gateway, if in doubt, select GENERIC.
URL Format	Is a default value for the connect URL. This provides a hint to the format of the connect URL that this driver requires while adding a datasource connection. For example, the hint for the format can be, `jdbc:dbtype://host:port/database`.
URL Instructions	Free form instructions that are shown to help the user to create a connection.
Default Connection Properties	Any additional properties to add by default to the connection string.
Connection Properties Instructions	Tips about which connection properties might be useful.
Default Validation Query	The default query that is used to verify that the connection is available.

SQL Language Compatibility

Default Translator	The database translator that is used by default for connections from this driver.

6. Click the **Create New JDBC Driver** button, located at the very bottom of the page, to create the new driver.

Upgrading a JDBC Driver

In some cases you may need to upgrade a driver. The steps below detail where this would take place

1. You will need to obtain the new driver. These are typically provided by the same organization that made the database. The driver will be a JAR file.

2. Once you have the new driver, head to your Ignition Gateway's **Configure** section, click on **Databases > Drivers**.

3. The **Database Drivers & Settings** page will be displayed. These are the currently configured JDBC drivers on the Gateway, and can be modified from the **Edit** button. Click the Edit button for the driver you need to upgrade.

4. You will need to pass in the new driver to the **JAR File(s)** property. Click the **Choose File** button, navigate to the driver, and click **Open**.

5. **[Optional]** Update any other properties. In most cases, you may skip this step. However you may need to update some other properties when a new driver is in place. This step depends on the driver, and what it changes. Refer to the driver's documentation to determine if any connection properties need to be changed. For example, users upgrading to MySQL 8.0 from legacy versions will need to change the **Default Connection Properties** value from:

   ```
   zeroDateTimeBehavior=convertToNull;
   ```

   to:

   ```
   zeroDateTimeBehavior=CONVERT_TO_NULL;useSSL=false;allowPublicKeyRetrieval=true;
   ```

   Again, this step depends on the driver, and in some cases you may be able to skip it.

6. Click the **Save Changes** button at the bottom of the page.

Database Translators

Despite the presence of a SQL standard, many database system vary in how they implement or accomplish various tasks. The JDBC driver system tries to hide these differences as much as possible, but unfortunately some differences persist.

The **database translator** system in Ignition navigates these differences as they apply to the system. It provides a way to define certain key operations that are commonly different between database vendors, such as creating auto-incrementing index columns, and the keywords used for different data types.

Translator Management

Database translators are managed in the Gateway from the **Databases > Drivers > Translators** tab. Ignition comes pre-configured with translators for the major supported databases, but you can edit and remove them, as well as create new translators. It is necessary to create a new translator only when adding a new JDBC driver for a database that does not share syntax with any of the existing translators.

Creating a New Translator

To add a new database translator to Ignition, do the following steps:

1. In Gateway on the **Configure** page, click on **Databases > Drivers**.
   The **Database Drivers & Settings** page is displayed.

2. Go to the **Translators** tab, find the blue arrow, and click on the **Create new Database Translator...** link.
   The **New Database Translator** page is displayed showing a list of all the translator properties.

3. Define the tokens used with the translator properties on the **New Database Translator** page.

   For most of the properties, you need to define special token markers to indicate places where other values are placed. For example, the default **Create Table Syntax** entry looks as follows:

   `CREATE TABLE {tablename} ({creationdef}{primarykeydef})`

   Where:
   `tablename`, `creationdef`, and `primarykeydef` are all tokens that are expanded. `tablename` is replaced directly with the table, `creationdef` is a list of columns, and `primarykeydef` is the phrase created by the **Primary Key Syntax** entry in the translator.

The possible tokens are as follows:

Token	Description
`tablename`	The name of the table being created.
`indexname`	The name of the index to create, when adding a column index to the table.
`primarykeydef`	A clause that defines a primary key for a new table.
`creationdef`	The list of columns to create in the table.
`alterdef`	A list of columns to add/remove/modify in the table.
`columnname`	The name of a column.
`type`	The data type of a column.
`limit`	The value of the limit clause.
Other Properties	
Limit Position	Defines where the limit clause should be placed. **Back**, the limit is placed at the end of the query. **Front**, places it directly after the SELECT keyword.
Column Quote Character	All columns are created and accessed with the defined quote, which tells the database to use a specific casing, as well as avoiding collisions between the column name and database keywords.
Supports Returning Auto-generated Keys? / Fetch Key Query	Indicates whether the JDBC driver supports the return of generated keys. If the driver does not support this feature, the **Fetch Key Query** is used to retrieve the last key.
Date Type Mapping	
All data types	The keywords that are used when creating columns of the given types.

4. Click the **Create New Database Translator** button, located at the very bottom of the page, to create the translator.

Store and Forward

The store-and-forward system provides a reliable way for Ignition to store data to the database. In Ignition, systems such as Tag Historian and SQL Bridge (Transaction Groups) use store-and-forward to ensure that data reaches its destination in the database, and is stored in an efficient manner. The store-and-forward system can be configured in a number of ways, offering both memory buffering for performance and local disk caching for safe storage.

Note: Store-and-forward engines are automatically created for each Database Connection.

On this page ...

Primary Features and Benefits

The store-and-forward system offers a number of benefits over other systems that log directly to the database, such as:

- **Data loss prevention**
  Data is removed from the system only when the write to the database has executed successfully.

- **Guaranteed ordering**
  Data is forwarded in the same order that it arrived, even if a database connection is not currently available.

- **Enhanced performance**
  By first buffering the data in memory, the store-and-forward system can optimize writes, and prevent the originating systems from blocking. This means that the system is less likely to lose data samples in the event of system slow downs.

INDUCTIVE UNIVERSITY

Using Store and Forward

Watch the Video

Store and Forward Data Flow

Although the system offers settings that can affect the pipeline, by default the data flow occurs as follows:

1. Data is generated in some system.
2. Data is placed in a memory buffer.
3. If not removed from memory buffer in some time (the **Write Time**), or if a certain amount of data accumulates (**Write Size**), it is placed in the local cache.
4. The data sink, based on a database connection, pulls data in first from the local store, and then the memory buffer, based on the **Write Time** and **Write Size** settings under **Forward Settings**.
5. If the data fails to forward, either due to an error in the connection or in the data itself, it is returned to the buffer or cache.
6. If the data errors out too many times, it becomes quarantined.
7. Quarantined data can be managed through the Gateway, and can be deleted or un-quarantined, once the error is resolved.

Understanding the Forward Triggers

Data is forwarded from one stage to the next based on the **Write Time** and **Write Size** triggers. These settings work as an **either/or** manner, meaning that if either of them is surpassed, the data is forwarded. One important point to note is that the **Write Size** setting influences the transaction size of similar data to be forwarded, and therefore can have a big impact on performance. As a result, the **Write Time** should normally be used as the controlling factor, with the **Write Size** set to something that will provide reasonable transactions, like 100.

Store and Forward for Reliability

The store-and-forward system settings, while seemingly limited, offer a good deal of flexibility in tuning. Different types of situations and goals will likely require different configurations.

When the safety of the data is a concern, the goal is to get the data stored to disk as quickly as possible in order to minimize risk of loss due to a power outage or system failure. The local cache plays a crucial role in this, allowing the system to store data locally for any amount of time until the remote database can accept it. This protects against network failures and database failures, as well.

By setting the **write size** and **write time** of both the local cache and forwarder to low values, the data spends less time in the memory buffer. While the memory buffer can be set to 0 to bypass it completely, this is not usually recommended, as the buffer is used to create a loose coupling between the history system and other parts of Ignition that report history. This disconnect improves performance and protects against temporary system slowdowns. In fact, it is recommended that for reliable logging, this value be set to a high value, to allow the maximum possible amount of data to enter the system in the case of a storage slowdown.

In This Section ...

Using Store and Forward

The current status of the Store and Forward system can be viewed from the Status section of the Gateway Web Interface. These pages provide detailed analysis on each Store and Forward engine.

Using Store and Forward

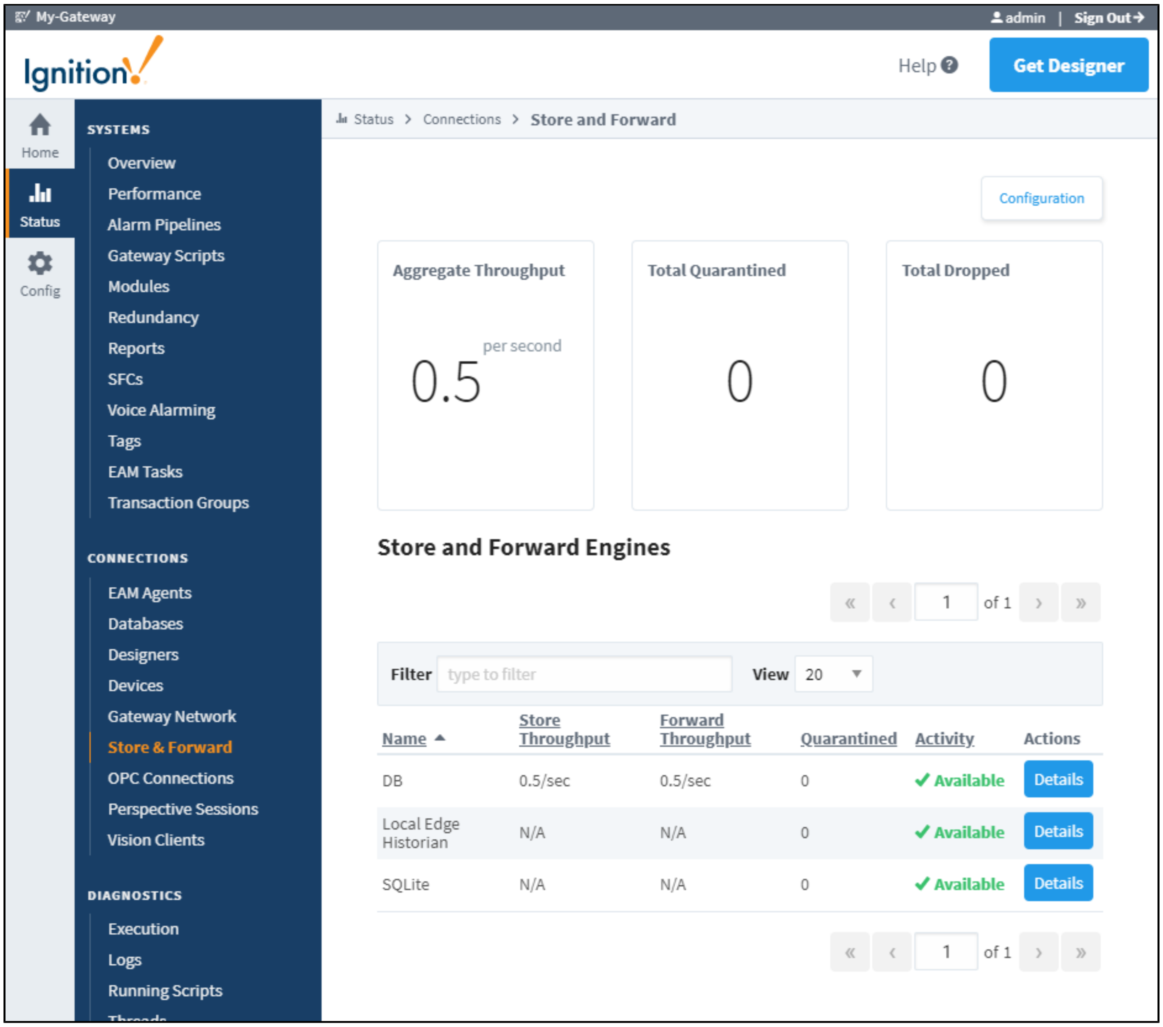
Watch the Video

To Monitor the Store and Forward Engine

1. Go to the **Status** section of the Gateway.
2. Click on **Store & Forward** from the left menus.
   The **Store & Forward Connections** page is displayed showing each store-and-forward engine along with the current throughput and capacity of its **Memory Buffer** and **Disk Cache**.

   On this page, there are several notable items:

Name	Description
Aggregate Throughput	The aggregated number of records inserted into a database from any engine, per second.
Total Quarantined	The current count of quarantined items across all engines.
Total Dropped	The number of records that have been dropped from all store and forward engines. A record is considered dropped if it can not be added to one of the buffers, such as when a buffer is full, and the engine can no longer accept new records.



Aggregate Throughput: 0.5 per second

Total Quarantined: 0

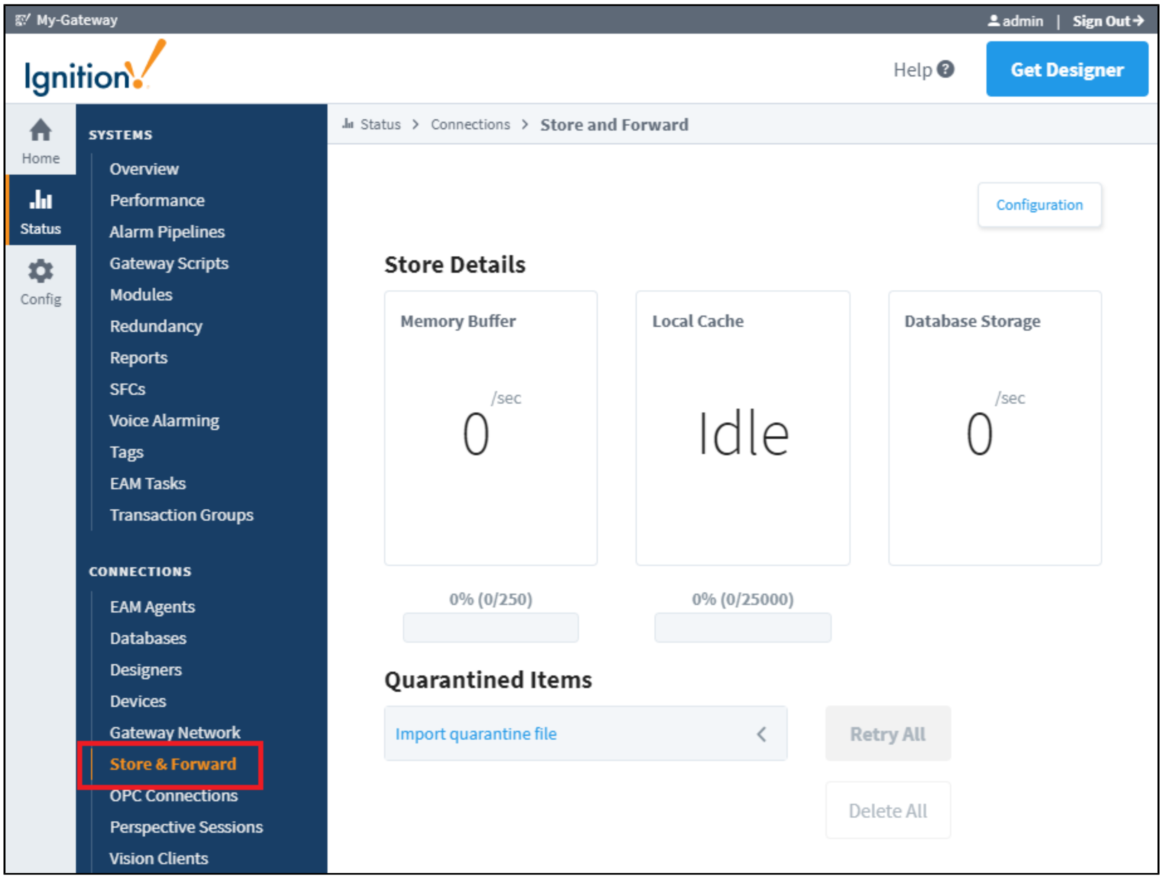
Total Dropped: 0

Store and Forward Engines

Name	Store Throughput	Forward Throughput	Quarantined	Activity	Actions
DB	0.5/sec	0.5/sec	0	✓ Available	Details
Local Edge Historian	N/A	N/A	0	✓ Available	Details
SQLite	N/A	N/A	0	✓ Available	Details

You can click on **Details** under the **Store and Forward Engines** section to refresh and update the displayed values. This page provides in-depth information on the current status of the engine.

Name	Description
Memory Buffer	The number of records entering the Memory Buffer per second. The progress bar shows the percent of the buffer being utilized, along with the current and max number of records.
Disk Buffer	The number of records entering the Disk Buffer per second. Note that a state of "idle" means the engine is able to successfully store all records into the database before the **Write Size** or **Write Time** values have been reached. The progress bar shows the percent of the buffer being utilized, along with the current and max number of records.
Database	Shows the number of records pushed from either buffer to the database per second.
Quarantined Items	Lists all quarantined items in the engine. Includes the number of occurrences, a description of the where the items originated from, and the reason why the record was placed into the quarantine. Provides an opportunity to retry, export, or delete the items.



Store Details

Memory Buffer	Local Cache	Database Storage
0 /sec	Idle	0 /sec
0% (0/250)	0% (0/25000)	

Quarantined Items

Related Topics ...

- Configuring Store and Forward

Configuring Store and Forward

Store and Forward for High-speed Buffering

When configuring the store-and-forward system for high-speed buffering, you are expecting the case that data will come in quick bursts. By buffering the data, the system can accommodate more information than would be possible going directly against the database.

The key points in configuring a buffering system is to avoid expensive operations like storing and reading from the local cache, and to set the memory buffer large enough to accommodate the expected burst sizes.

Each database connection has its own store and forward settings. Store-and-forward engines are directly correlated to database connections, and are automatically managed so that each connection has an engine defined.

> ✓ You can create multiple database connections pointing to the same database if you wish to configure multiple store-and-forward engines for different purposes.

On this page ...

INDUCTIVE UNIVERSITY

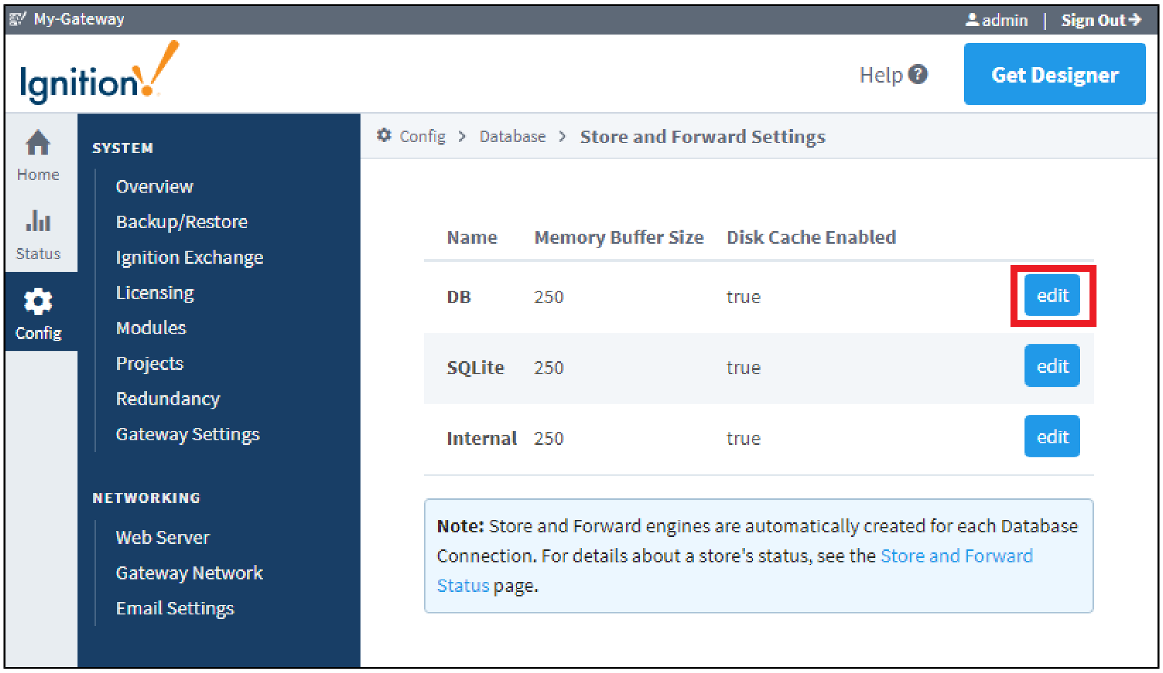
Configuring Store and Forward

Watch the Video

Configuring Store and Forward

To configure the Store and Forward engine for your database, do the following steps:

1. Go to the Gateway **Config** section and select **Databases > Store and Forward**.
   The **Store and Forward** page is displayed and you will see a store-and-forward setting for each of your database connections and you can edit these settings.
2. On the **Store and Forward** page, look for **edit** at the far right of the table and click on it to see all the store-and-forward settings.
   The settings of a store-and-forward engine define how and when data is moved through the system. You must understand these settings so that you can carefully set them according to your goals.



Name	Memory Buffer Size	Disk Cache Enabled	
DB	250	true	edit
SQLite	250	true	edit
Internal	250	true	edit

Note: Store and Forward engines are automatically created for each Database Connection. For details about a store's status, see the Store and Forward Status page.

Store and Forward Settings

Buffer Settings

Memory Buffer Size	The number of records that can be stored in the memory buffer, the first stage of the store-and-forward chain. Other settings define when the data will move from the memory buffer forward, this setting only determines the maximum size. If the max size is reached, additional data will error out and be discarded. The memory buffer cannot quarantine data, so if there are errors and the disk cache is not enabled, the data will be lost. If set to 0, the memory buffer will always be considered full, dropping records.

Store Settings

These settings apply to the local disk storage cache.

Disk Cache Enabled	Turns on the hard-disk cache. Data is stored here if it cannot be forwarded in a timely manner. The cache also stores quarantined data (that is, data with errors).
Max Records	The maximum size of the cache. After the max is reached, data is backed up into the memory buffer, and once that is full, it is dropped. A 'record' is an insert or update statement. These statements may be batches, thus it is possible for a single 'record' to impact multiple rows. The default value is 25,000. **Note:** The disk cache is intended to be a short term storage system. It is not recommend to increase the Max Records beyond 50,000.
Write Size	The number of records that should be accumulated in the memory store before written to the cache. Writing data in blocks can increase performance, but too large of a size increases the risk of data being lost in the event of a power outage or system failure.
Write Time	The max age of records in the memory buffer before they are stored to the cache. This setting is used in combination with the write size in order to give the forwarder the opportunity to retrieve data directly from the memory store and avoid the write to disk entirely.

Forward Settings

These settings govern when data is forwarded to the database. The data is pulled first from the local cache, and then from the memory store. When no data is present in the cache, it is pulled directly from the memory store.

Write Size	Same as disk cache setting above.
Write Time	Same as disk cache setting above.
Enable Schedule	If enable schedule is selected, the forward engine will only be enabled during the times specified by the pattern. The pattern can specify specific times and ranges using a simple syntax.
Schedule Pattern	The schedule is specified as a comma separated list of times or time ranges. You can use the following formats: 24-hour times, that is **8:00-15:00** (for 8am through 3pm) or **21:00-24:00** (9pm through midnight). 12-hour with am/pm (if not specified, **12** is considered noon): **8am-3pm or 9pm-12am** **Note:** When the time period is over, any queued data will remain cached until the next execution period. That is, the forward engine does not run until all data is forwarded.

Once you made the changes you want, click **Save Changes** at the bottom of the page.
This will take you back to the Store and Forward page.

Related Topics ...

- Controlling Quarantine Data

Controlling Quarantine Data

Quarantined data is data that has erred-out multiple times during attempts to forward it or data that could not be stored because of some configuration issues. It is removed from the forward queue to allow other data to pass. The most common reason for data quarantining is an invalid schema in the database for the data that is being stored. Quarantined data is held indefinitely until the issue is resolved, then you can either delete it or re-insert it into the queue.

Handle the Quarantined Data

1. From **Status** section of the Gateway, go to **Connections > Store and Forward**.

2. Click the **Details** tab next to a Store and Forward engine that you would like to see the quarantine data for. Here you will see any quarantined data, including the number of occurrences, a description of the where the items originated from, and the reason why the record was placed into the quarantine. Each set of data has the option to retry it, delete it, or export it for later use. If there are a lot of quarantined records, it may be a good idea to export and delete them so that the store and forward engine won't fill up and drop records.

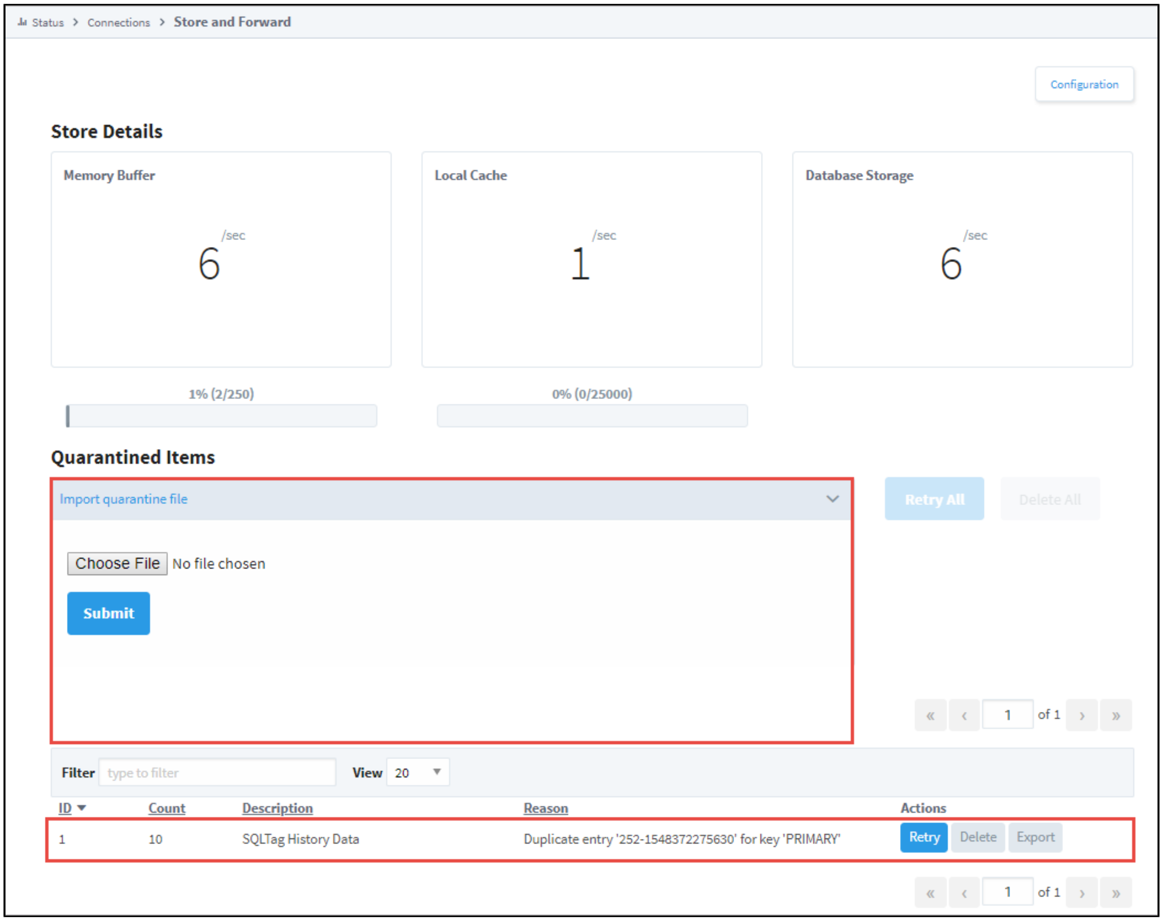
3. Fix the problem/error you found that caused the quarantine data.

4. Click on **retry**, or if the data had been exported, import the data using the import tool and then **retry**. This way, you can ensure no data gets lost.

INDUCTIVE UNIVERSITY

Controlling Quarantine Data

Watch the Video



Store Details

Memory Buffer	Local Cache	Database Storage
6 /sec	1 /sec	6 /sec
1% (2/250)	0% (0/25000)	

Quarantined Items

No file chosen

Filter: type to filter View: 20

ID	Count	Description	Reason	Actions
1	10	SQLTag History Data	Duplicate entry '252-1548372275630' for key 'PRIMARY'	Retry Delete Export

Related Topics ...

- Store and Forward
- Connections - Store & Forward

Security

Security options in Ignition provide many ways to safeguard your data and applications. You control not only who accesses your systems, but when and where they can access them. You can add as many user sources as you need. For users, you can store detailed information beyond just a user name and password, such as schedules, roles, and phone numbers. At the core of Ignition's security are users and the roles of users.

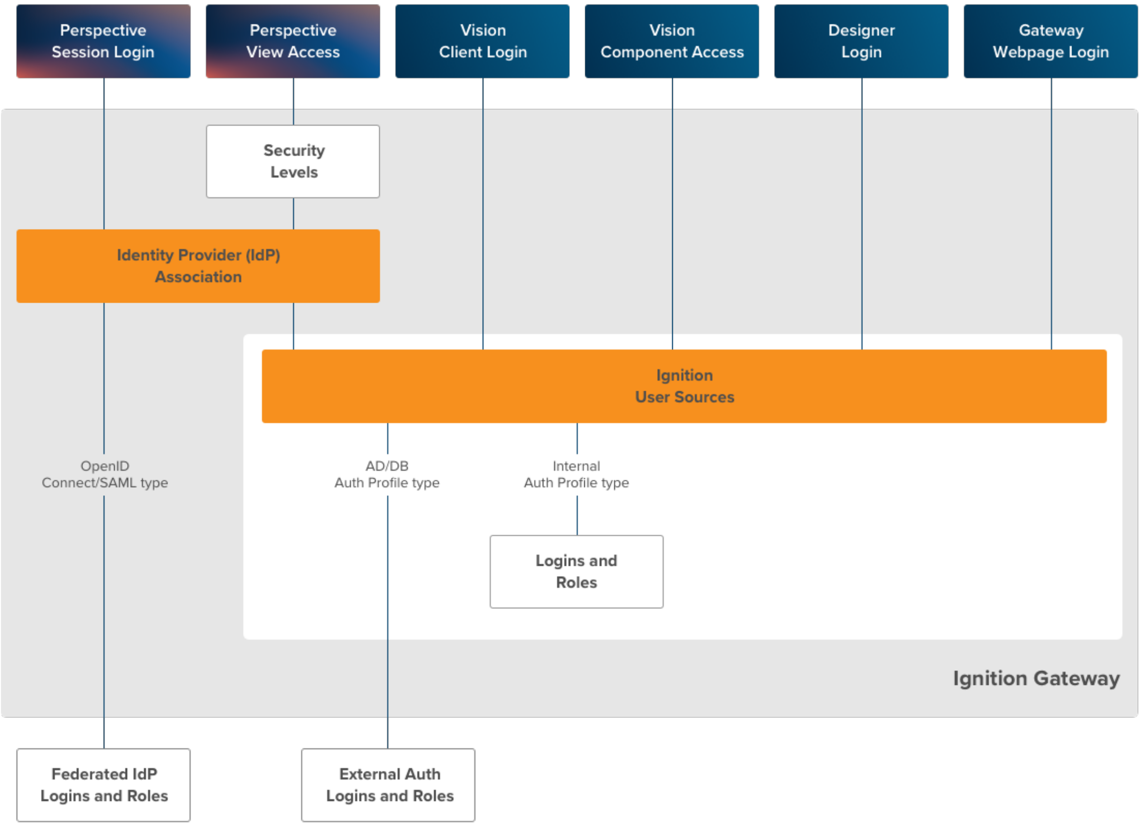
At its simplest, setting up security in Ignition follows this pattern:

1. Have a user source for logging in (one or more users).
2. Set up roles that define the types of access users may need (for example, Administrator, Supervisor, Operator, or Guest).
3. Establish security on Ignition Projects, the Gateway, the Designer, Vision components, Perspective Views, and so forth.
4. Assign roles to the individual users.

The diagram below illustrates how Perspective and Vision access user sources and shows some of the differences between Federated IdPs, external AD authorization, and internal Ignition authorization. A few notable points:

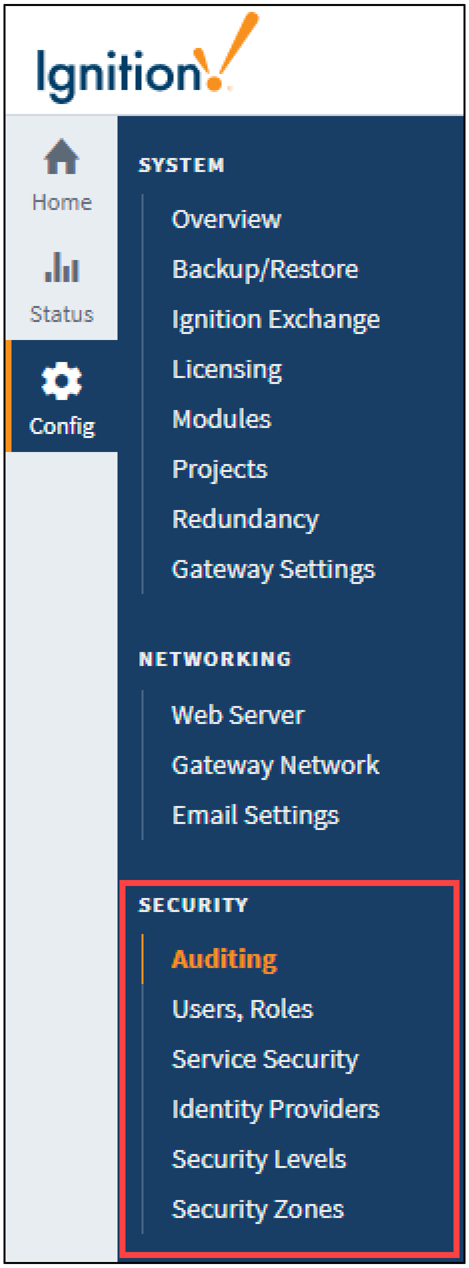
- Vision clients only use User Sources. Roles in Vision clients are automatically pushed into the Designer.
- The Perspective module gets users and roles from an IdP. Roles must be then set up in Security Levels in order to be available to the security screens in the Designer.
- Perspective sessions can be made available to all users (Security Level - Public) or set to require user authentication.

On this page ...



Security Setup

Security in Ignition falls into a few categories, and the bulk of the setup happens in the Gateway Webpage. Under Security in the Config section of the Gateway Webpage, you'll find pages for authentication, role mappings, and zones.



User Sources

- User Sources can be created and stored in Ignition, they can access user info from an external database, or from an Active Directory (AD) profile.
- In Ignition 8.0 with the Perspective module, users can also come from Ignition's internal Identity Providers (IdP) or an external IdP using trusted federated identity technologies such as OpenID Connect or Security Assertion Markup Language (SAML).

Roles and Security Levels

- Roles are created and then assigned to users. Roles can be stored in Ignition or roles can be linked to AD Groups.
- In Ignition 8.0 with the Perspective module, Security Levels are established to connect users with roles. Security Levels are assigned to users through Security Level Rules or User Grants.

Project and Component Security

- All Projects have security settings indicating what roles can publish, view, save, delete, or access Project resources.
- In the Vision module, you can set security for managing alarms, editing Tags and a host of other functions. You can also set security on Vision components and windows .

- In Ignition 8.0 with the Perspective module, you can set security on projects and views. You can also set permissions in event Actions.

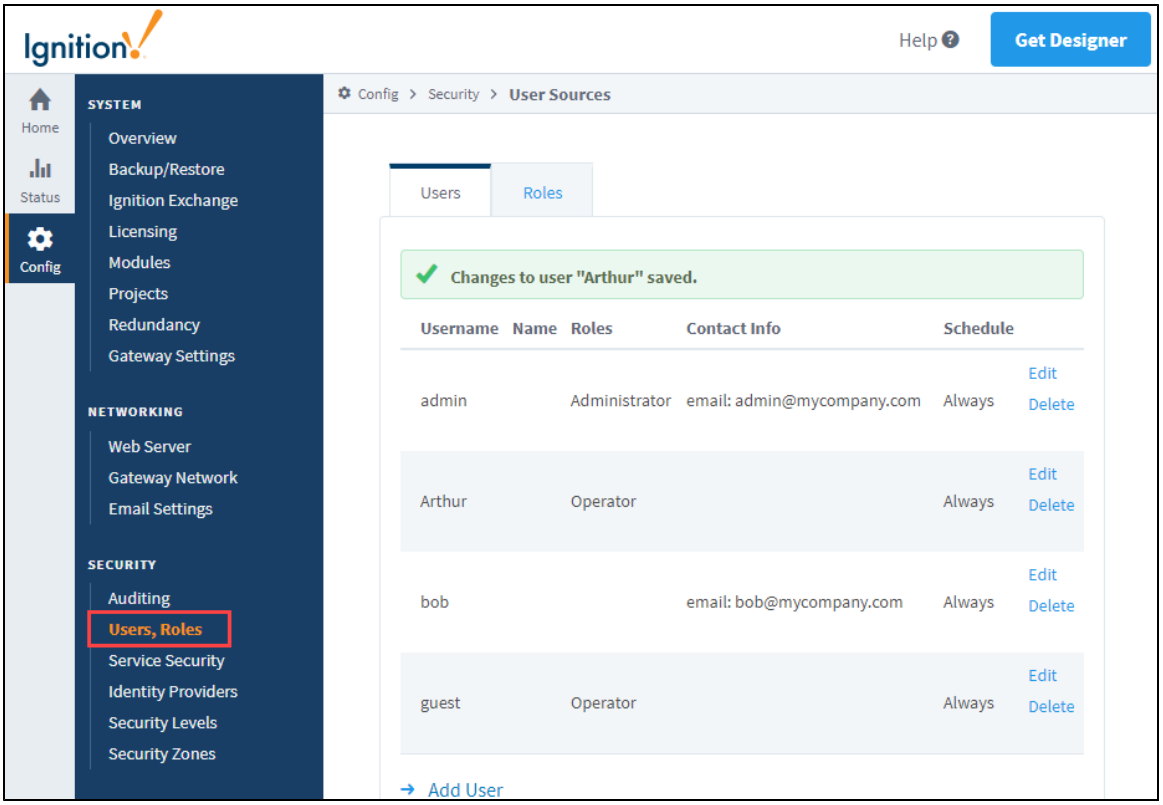
Gateway Security

- The primary purpose of Gateway security is to protect access to the two most critical areas of Ignition: the Designer and the Gateway. Many important resources are configured in these areas, so access to each Gateway section (Status and Config), as well as the Designer, can be limited by role.
- Through the Gateway security settings you can also choose to use Secure Sockets Layer (SSL), a widely adapted encryption protocol used all over the world.
- Through the Gateway you can also set up Audit Log and Profiles, which cause Ignition to record details about specific events that occurred. For more information, see Project Auditing and Alarm Notification Auditing.

Security with User Sources

Role-based security works under the concept that each user may be assigned to various roles. Security policies are then defined in terms of these roles, rather than defined for specific users. An example of roles could be an Administrator role that has access to the Designer and the client or session or an Operator roles that have access only to windows or views that pertain to their jobs. Roles allow users to be reassigned, removed, and added without affecting the logic of the security policy.

The users and their roles are stored in User Sources. An Ignition Gateway may have many different User Sources defined, each governing the security of different aspects of the Gateway. For example, logging into the Gateway might be governed by one User Source, while the security in a project is governed by another. The example below shows the Users and Roles screen after user "Arthur" has been updated.



✓ Changes to user "Arthur" saved.

Username	Name	Roles	Contact Info	Schedule	
admin		Administrator	email: admin@mycompany.com	Always	Edit Delete
Arthur		Operator		Always	Edit Delete
bob			email: bob@mycompany.com	Always	Edit Delete
guest		Operator		Always	Edit Delete

There are several types of User Sources that offer various features. For example, the Internal User Source offers the ultimate in ease-of-use: you simply define the users, their passwords, and the roles within the Ignition Gateway configuration web interface. In contrast, the Active-Directory User Source offers the power of integrating Ignition with a corporate security infrastructure. Users, passwords, and roles would be managed centrally by the IT department.

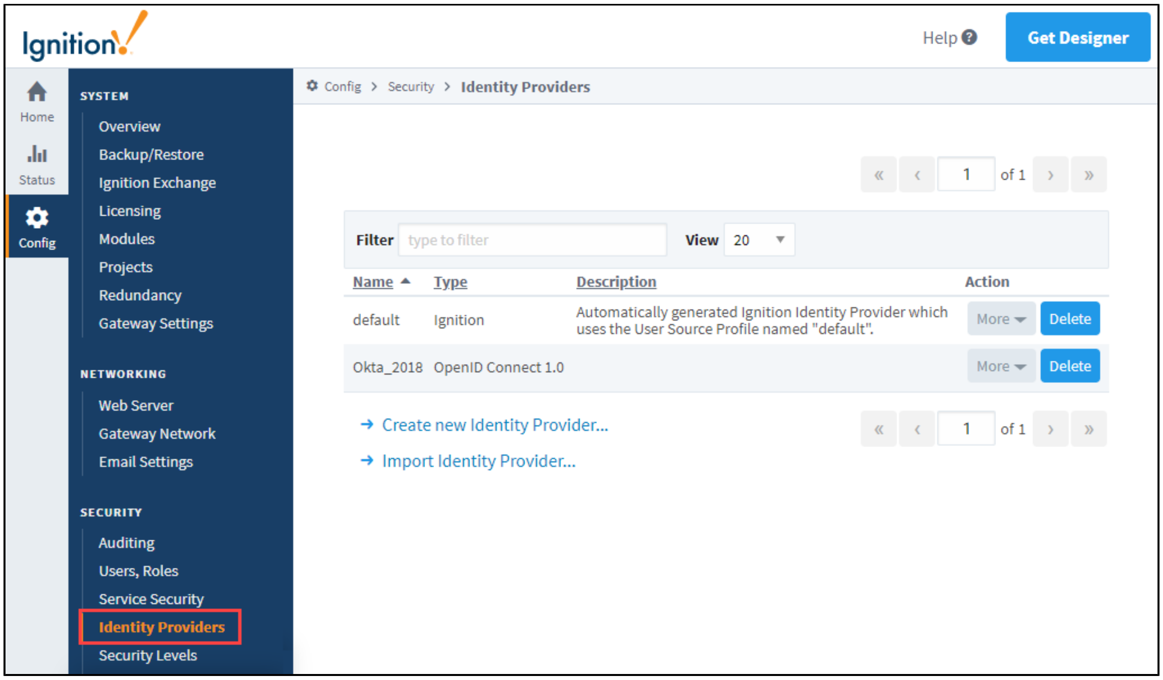
Security with Identity Providers

⚠ Identity Providers and Security Levels are currently only available for use with the Perspective module.

Identity Providers (IdPs) offer user authentication as a service. An IdP creates, maintains, and manages identity information for principals while providing authentication services to relying party applications within a federation or distributed network. Authentication of the user is handled by the IdP. Ignition can connect to these three different types of IdPs:

- Ignition's internal IdP
- OpenID Connect 1.0
- Security Assertion Markup Language (SAML)

IdPs are set up at the Gateway level. Security Levels are also set through the Gateway. The Security Levels enable you to define a hierarchy of access inside a Perspective Session.



Name	Type	Description	Action
default	Ignition	Automatically generated Ignition Identity Provider which uses the User Source Profile named "default".	More Delete
Okta_2018	OpenID Connect 1.0		More Delete

In This Section ...

User Sources

User Sources

User sources (previously known as Authentication Profiles), are a collection of users, roles, and other user data, such as contact information or schedule. When a new user or role is created, it is applied and stored in the user source. Projects and the Gateway are assigned a User Source to authenticate against. This determines which users have access to which project(s).

There are several types of user sources: single-storage types with varying storage mediums, "hybrids" that combine features of the previous three types, and a cache type used in Local Client Fallback systems.

⚠ If you have Ignition 8.0 with the Perspective module, authentication is handled instead by Identity Providers.

On this page

...

Single-Storage

Users and roles are stored in a single location. The single-storage users sources are:

- Internal Authentication - Users and roles are stored internally to Ignition.
- Database Authentication - Users and roles are stored in a SQL database. Managing users is done via direct interaction with the database.
- Active Directory Authentication - Users and roles are managed by Active Directory. Users are authenticated through the LDAP protocol.

Hybrid

Users in hybrid user sources authenticate against Active Directory, meaning that user names and passwords are checked against those stored in Active Directory. However, roles are stored either internally in Ignition or in a SQL database, so it is possible to make a role change without have to contact your Active Directory administrator. This way, Active Directory can be consulted to see if a user is valid, but the management of roles does not require coordination with the IT department, who typically control the Active Directory system. This "best of both worlds" approach is popular for many users of Active Directory.

- Active Directory-Internal Hybrid - Users managed by Active Directory and roles stored to Ignition internally.
- Active Directory-Database Hybrid - Users managed by Active Directory and roles stored in an SQL database.

Fallback Cache

This User Source was developed specifically for a system that is using Local Client Fallback, and allows you to cache the login credentials from a remote user source. This means your users can still log in with their normal username/password on a Local Client Fallback project, even when the network connection is unavailable.

More information can be found on the Fallback Cache Authentication page.

Shared Functionality

Regardless of type, all User Sources have the following functionality:

- **Failover Source**: If the User Source is unavailable for authentication, then a backup User Source can be specified. The type of the fail-over User Source can differ from the primary, so configurations where an internal-type fails over to a database-type are possible.
- **Schedule Restrictions**: The User Source can prevent users from logging in when they are off schedule, meaning that the schedule assigned to the user determines when the user may login.

Main Properties

All User Sources have a section of properties that are categorized as "Main". Below is a description of these properties.

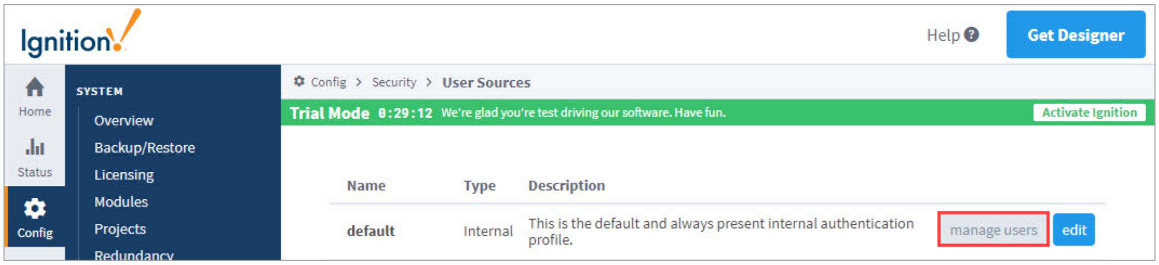
Name	Description
Name	The name of the User Source. This is how other systems in Ignition reference the user source. Note that every User Source **must** have a unique name.
Description	An optional description of the user source. Useful for noting which database connection or AD server the User Source may be referencing.
Schedule Restricted	Forces schedule restrictions on users. Specifically, if a user attempts to log into a client while they are off schedule, the login will fail. Utilizes User Schedules.
Failover Source	Allows authentication attempts against this User Source to failover to another User Source in the event of a network outage, or some other connection issue. Useful with database or Active Directory user sources, as connection failures to the database /AD server will prevent users from logging in. This property is initially set to **None**, meaning a failover User Source is not configured.
Failover Mode	When a **Failover Source** is configured, this property determines when the failover User Source should be consulted. The following options are available: **Hard:** The Failover User Source is only consulted when this User Source is unreachable. **Soft:** The Failover User Source will be consulted if the user's credentials fail authentication, meaning that the user typed in credentials that are unrecognized or incorrect.
Cache Validation Timeout	The following feature is new in Ignition version **8.0.10** Click here to check out the other new features The amount of time between cache updates of the User Source. As of version 8.0.10, if you set this value to -1, the cache validation timeout is turned off.
Lockout Enabled	The following feature is new in Ignition version **8.0.5** Click here to check out the other new features Lock out a user's account after more than the maximum allowed number of failed authentication attempts occur within the lockout window. Default is true. Note that access can be restored to all locked out users by editing the user source, and clicking the **Save Changes** button.
Lockout Attempts	The following feature is new in Ignition version **8.0.5** Click here to check out the other new features Maximum number of failed authentication attempts allowed within the lockout window before locking the user out. Default is 5. If this value is set to something less than zero (for example, -1), then the lockout functionality will be entirely disabled, regardless of what the **Lockout Enabled** property is set to.
Lockout Window	The following feature is new in Ignition version **8.0.5** Click here to check out the other new features The duration of the lockout window in minutes. Default is 15. Setting this property to a value of less than zero (for example, -1) will disable the lockout functionality entirely, regardless of what the **Lockout Enabled** property is set to.

Details on the Password Policy Properties can be found on the Internal Authentication page.

The Default User Source

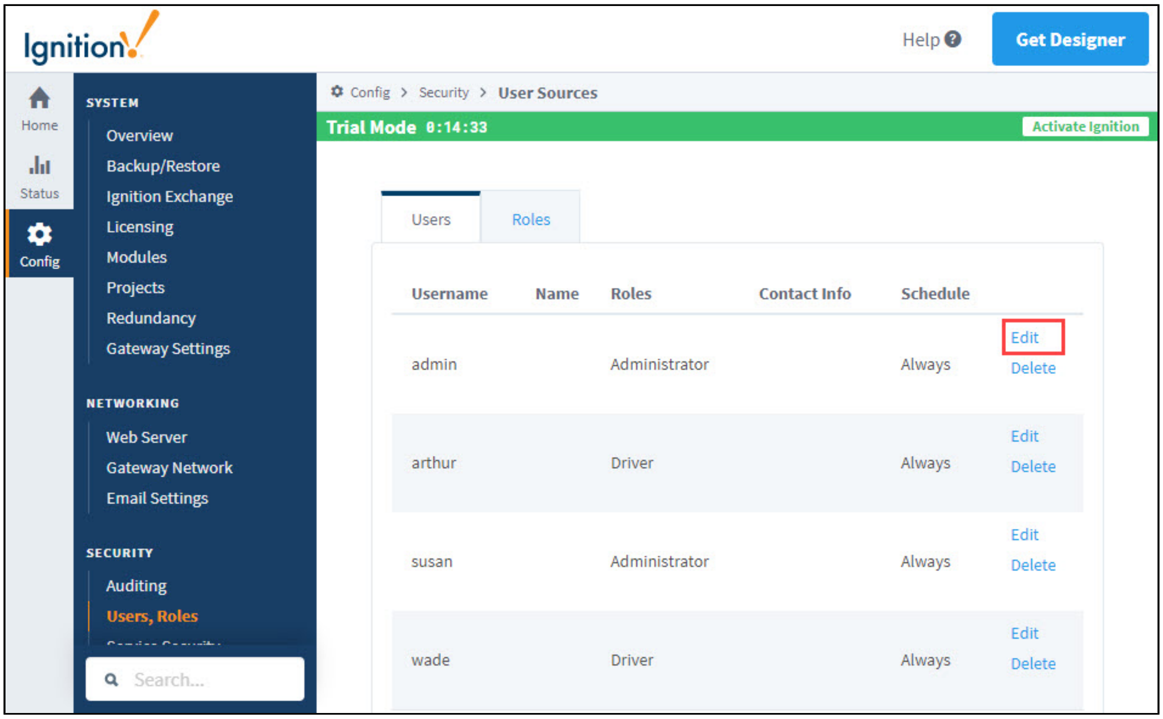
When Ignition is installed for the first time, an internal User Source named 'default' is created. You can manage the default User Source by navigating to the **Config > Security > Users**, **Roles** section of the Gateway.

The manage users link next to the 'default' user source allows you to add new users, modify roles and passwords for existing users, remove users, and add/remove roles from the user source.



Name	Type	Description
default	Internal	This is the default and always present internal authentication profile.

When you open the 'default' user source for the first time, you will see the first user that was created at installation. This is the administrator account that has full privileges. If this user source has been modified before, a list of existing users is displayed.



Username	Name	Roles	Contact Info	Schedule	
admin		Administrator		Always	Edit Delete
arthur		Driver		Always	Edit Delete
susan		Administrator		Always	Edit Delete
wade		Driver		Always	Edit Delete

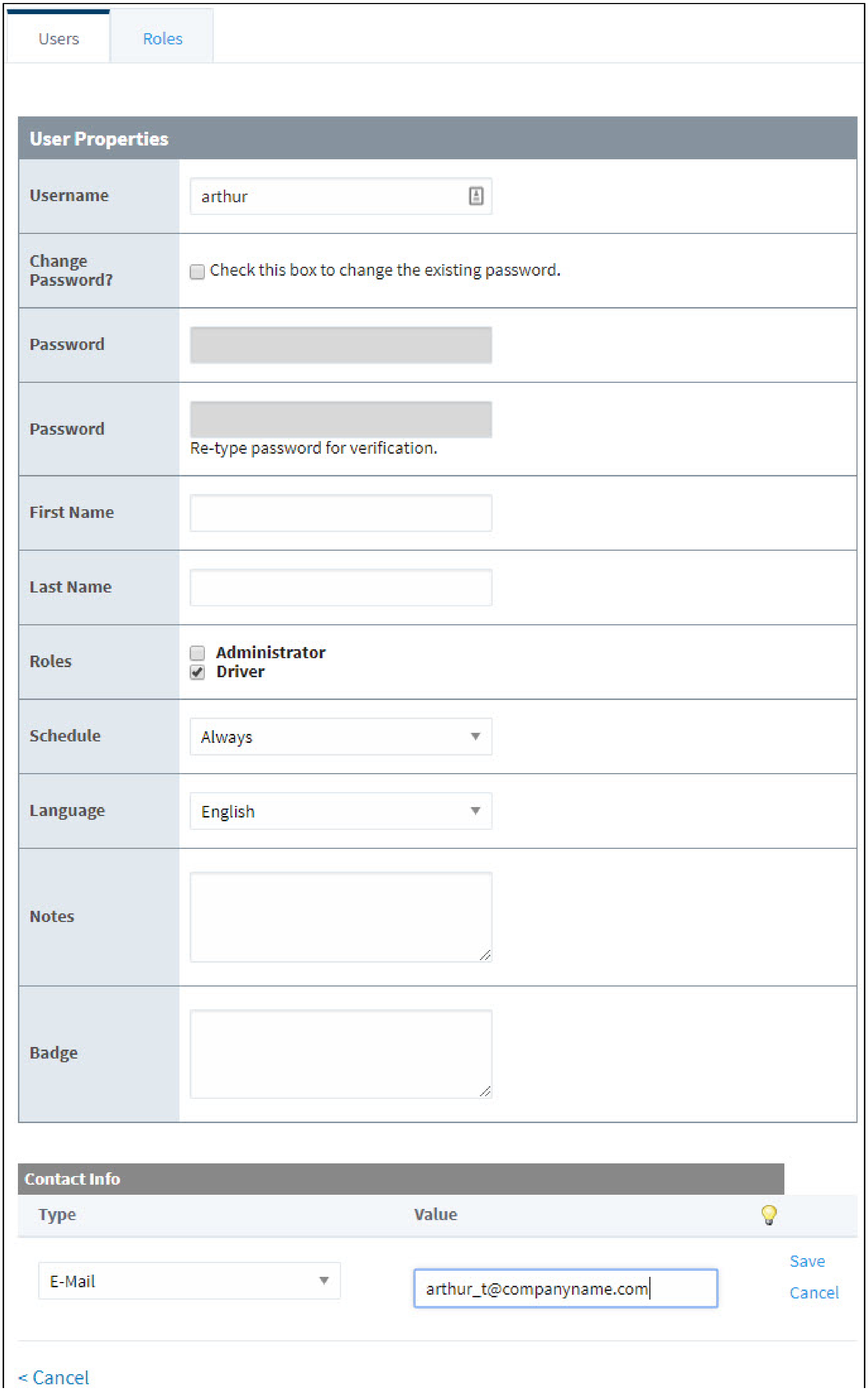
Editing a User

Choosing to edit a user will bring you to the following page allowing you to make any necessary changes to that user. Fill out the fields for that user then click **Save Changes**.

User Properties

Name	Description
Username	The name of the user.
Change Password?	Check this box to change the existing password.
Password	New password.
Password	Re-type password for verification.
First Name	First name of the user.
Last Name	Last name of the user.
Roles	Role(s) assigned to this user. Check the box next to each role you want this user to have.
Schedule	Schedule for the user. Choose from a dropdown list of schedules that are already defined.
Language	Language to be used for the user. Choose from a dropdown list of languages that are already defined.
Notes	Any notes for this user.

Badge	The following feature is new in Ignition version **8.0.5** Click here to check out the other new features A string that represents the value set for the user's badge.
Contact Info	
Type	Choose email or SMS.
Value	The email value or SMS number.



User Properties

Username	arthur
Change Password?	□ Check this box to change the existing password.
Password	
Password	Re-type password for verification.
First Name	
Last Name	
Roles	□ **Administrator** ☑ **Driver**
Schedule	Always ▼
Language	English ▼
Notes	
Badge	

Contact Info

Type	Value	
E-Mail ▼	arthur_t@companyname.com	Save Cancel

< Cancel

Which User Source Controls What?

With potentially multiple User Sources defined, you need to understand which User Sources are controlling which aspects of Ignition. To determine what kind of User Source is governing what, do the following:

1. To manage users and passwords for logging into the Gateway Configuration section, you'll need to see what User Source is currently set as the Gateway's User Source. You can check this under **Config > System > Gateway Settings** by looking at the System User Source field and the Gateway Config Role(s) field.
2. To manage users and passwords for logging into the Designer, you follow the same steps as in #1, except that you need to look at the Designer Role(s) field to see what roles are allowed to log into the Designer.
3. To manage users and passwords for logging into a Vision Client or Perspective Session, go to the **Config > System > Projects** section. Look at the project in question and you can find its User Source listed under **Authentication**, or click **edit** and look at **Authentication Profile**.
4. Now that you know what User Source you need to manage, you can find out what kind it is under in the **Security > Users, Roles** section.

In This Section ...

Managing Users and Roles

Users and Roles

Security is based on the roles that are assigned to specific users. Roles do not have any structure or hierarchy by default, but can be created. You can create a hierarchy based on users with a greater role being assigned all matching lesser roles.

There isn't a built-in restriction to the number of roles a user can have, so each user can have access to many roles, or none at all.

> **Roles and Security**
>
> It's important to think about the different roles in your project and how they affect the security of your project. For instance, what level of access a particular area of a project needs may determine the functional type roles that you create, and the different users assigned to each role.

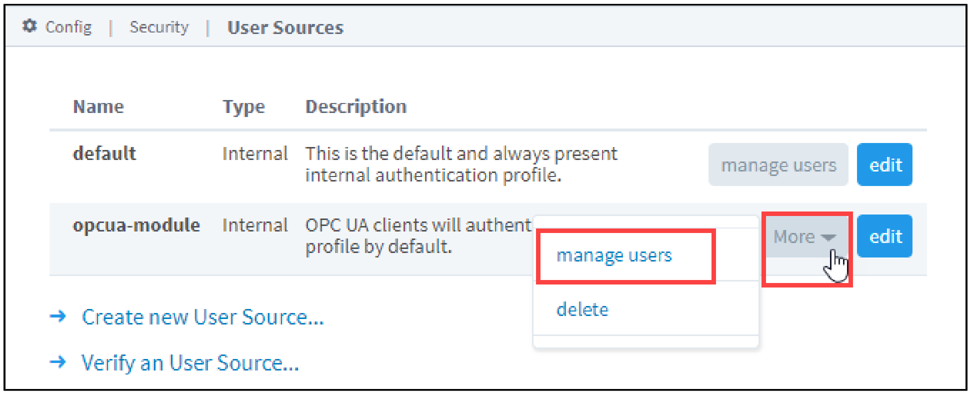
You can manage users and roles using either the Gateway interface, or using the User Management component inside the Designer or Client. This section shows how to manage users and roles using the Gateway interface.

> When using role-based security in a project, the project stores the name of the role as a string. This means that if you were to modify the name of the role in the Gateway, the role-based security in your project will not update to reflect the new name, and instead will try searching for a role with the original name. Be very careful when modifying the names of roles.

On this page ...

Creating a Role

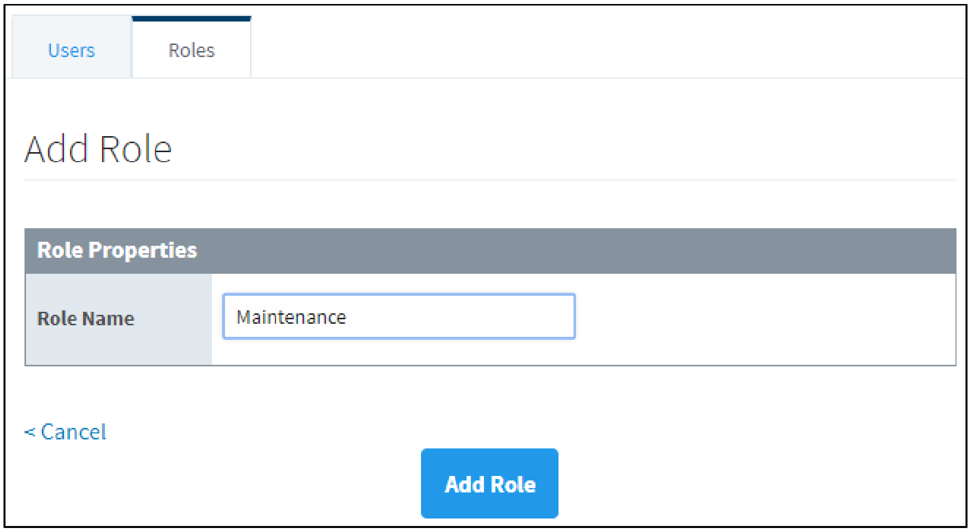
1. On the Gateway Webpage, go to the **Config** section, and choose **Security > Users, Roles** from the menu on the left. The User Sources page is displayed.
2. Click on the **manage users** link for the **User Source** you want to manage.



Config | Security | User Sources
manage users
edit
More
manage users
delete
edit
→ Create new User Source...
→ Verify an User Source...

Name	Type	Description
default	Internal	This is the default and always present internal authentication profile.
opcua-module	Internal	OPC UA clients will authent profile by default.

3. Click the **Roles** tab. Look for the blue arrow at the bottom, and click the **Add Role** link.
4. Name the role by entering it in the **Role Name** field, and click on the **Add Role** button. The role is now available to be associated with specific users.



Add Role

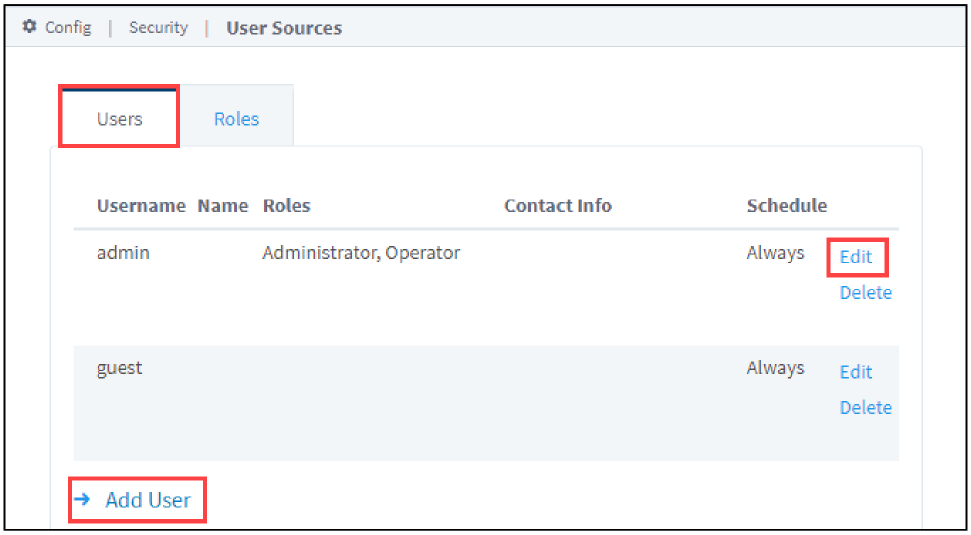
Role Properties	
Role Name	Maintenance

Assigning Roles to Users

1. On the Gateway Webpage, go to the **Config** tab, and choose **Security > Users, Roles** from the menu on the left. The User Sources page is displayed.

2. Click on the **manage users** link for the **User Source** you want to manage.

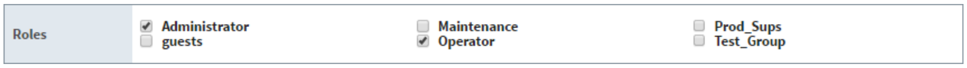
3. Click the **Edit** link for the User you want to edit, or click the blue **Add User** link to add a new user. (When adding a new user, you can also add their roles at the same time).



Username	Name	Roles	Contact Info	Schedule	
admin		Administrator, Operator		Always	Edit Delete
guest				Always	Edit Delete

4. If you're creating a new user, the Add User window will open. Enter the user's properties including the roles you want this user to have. If no roles have been created, then follow the instructions in the Creating a Role section from above. If your user already exists and you simply want to modify their roles, the Edit User window will open. (The Edit User window and the Add User window look identical).

   To assign a role, there is a **Roles** property with a list of roles that have already been created. Select the role(s) that you want this user to have. (It's not required for a user to have a role, but be aware that they might not have access to an area of the project that requires them to have a role).



Roles	☑ Administrator □ guests	□ Maintenance ☑ Operator	□ Prod_Sups □ Test_Group

Administrator Role

> When a project is first created, the **Administrator** role is the only role available, and no other roles will appear until they are created. When more roles are created, they appear as check boxes just like the Administrator option.

5. Click either **Add User** if you adding a new user, or **Save Changes** if you are modifying a user's role(s).
   The user now has the privileges associated with the selected role(s).

Role Hierarchy

Often you might want to have one role that includes all the permissions for another role, i.e., Supervisor can do everything that Administration and Maintenance roles can do. In the Designer, access to Components can be restricted to specific security roles. You can give any Supervisor both of the Administration and Maintenance.

Managing Users

User Sources support managing the users and roles from within Ignition to varying degrees. Some User Sources are fully manageable, meaning that you can administer the users, roles, contact info, and so on from within the Ignition Gateway, as well as inside a Vision Client. Other User Sources do not support this at all, while yet others only partially support it. Make sure you understand how and where the administration takes place before you choose a User Source type.

For User Sources that support it, you can manage the users and roles from within the Ignition Gateway's web configure interface under **Config > Security > Users, Roles**. Click on the **manage users** link for the **User Source** you want to administer.

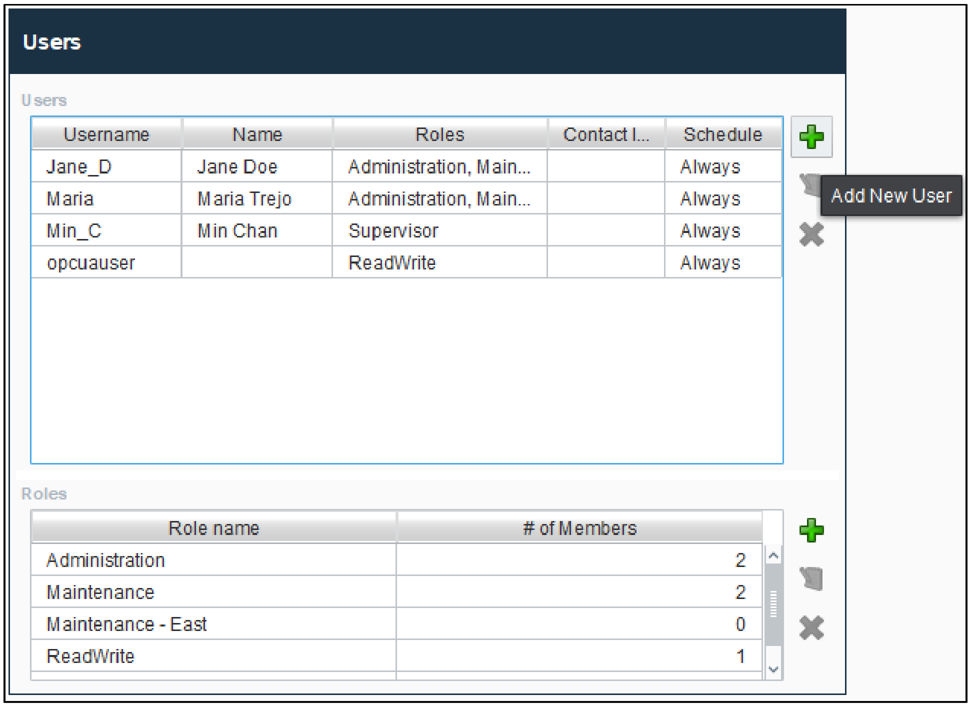
Often, it is desirable to let some management or administrative users of a Vision project manage other users without having to log into the Gateway's Configure section. To do this for a User Source that supports being managed, you can simply use the built-in User Management Panel that comes with the Vision Module.

User Management Component

Ignition has a special User Management component in the Vision Module that allows you to add, modify, and delete users and roles (and more) inside the Designer and the Client. This is simple to set up and use.

Using the User Management Component in the Designer and Vision Client

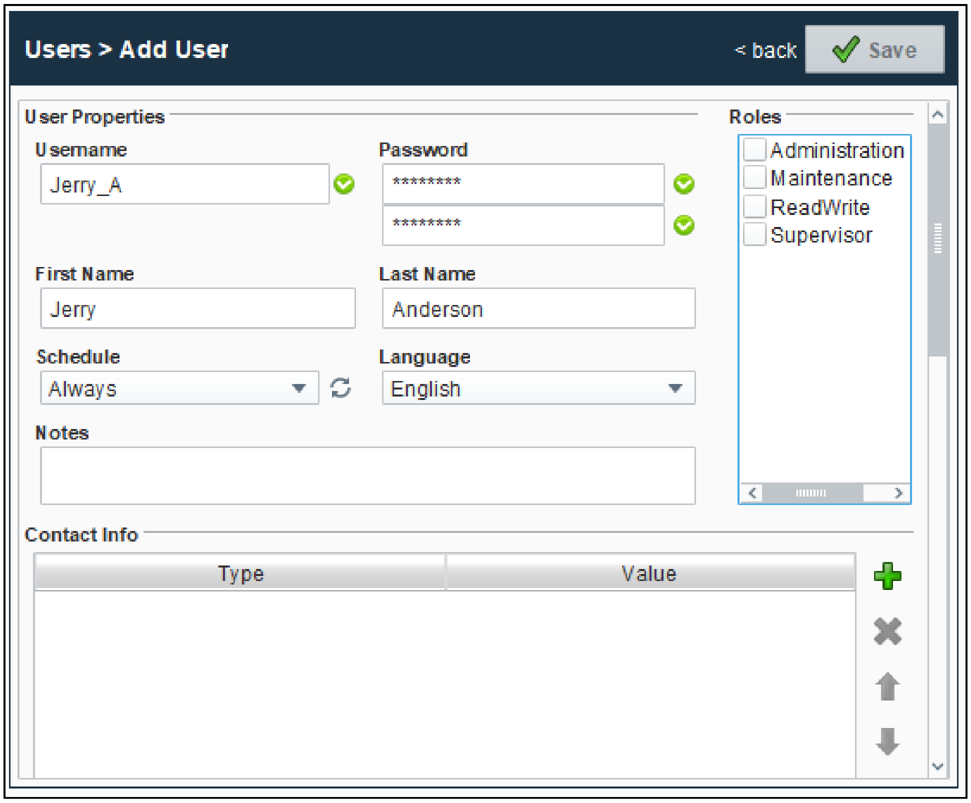
1. In Designer, go to the Project Browser and then to Vision.
2. Create a new Window or open an existing one.
3. Drag a User Management component to your window. This component will automatically point to the default user source being used by your project. You can change the User Source property if needed.
4. If you already have some users and roles setup using the Gateway Webpage, you will see them in the User Management component. If you don't have any users or roles setup, you can create them here. Use the icons on the right side to add, edit, or delete a user or role.
5. To add a new user, put the Designer in **Preview Mode**. Click the the plus ✚icon next next to the user section.



Username	Name	Roles	Contact I...	Schedule
Jane_D	Jane Doe	Administration, Main...		Always
Maria	Maria Trejo	Administration, Main...		Always
Min_C	Min Chan	Supervisor		Always
opcuauser		ReadWrite		Always

Role name	# of Members
Administration	2
Maintenance	2
Maintenance - East	0
ReadWrite	1

6. The Add User window will open. At a minimum, enter the **Username** and **Password**. All other properties are optional. When finished, click **Save**.



Username: Jerry_A

Password: ********

****

First Name: Jerry

Last Name: Anderson

Schedule: Always

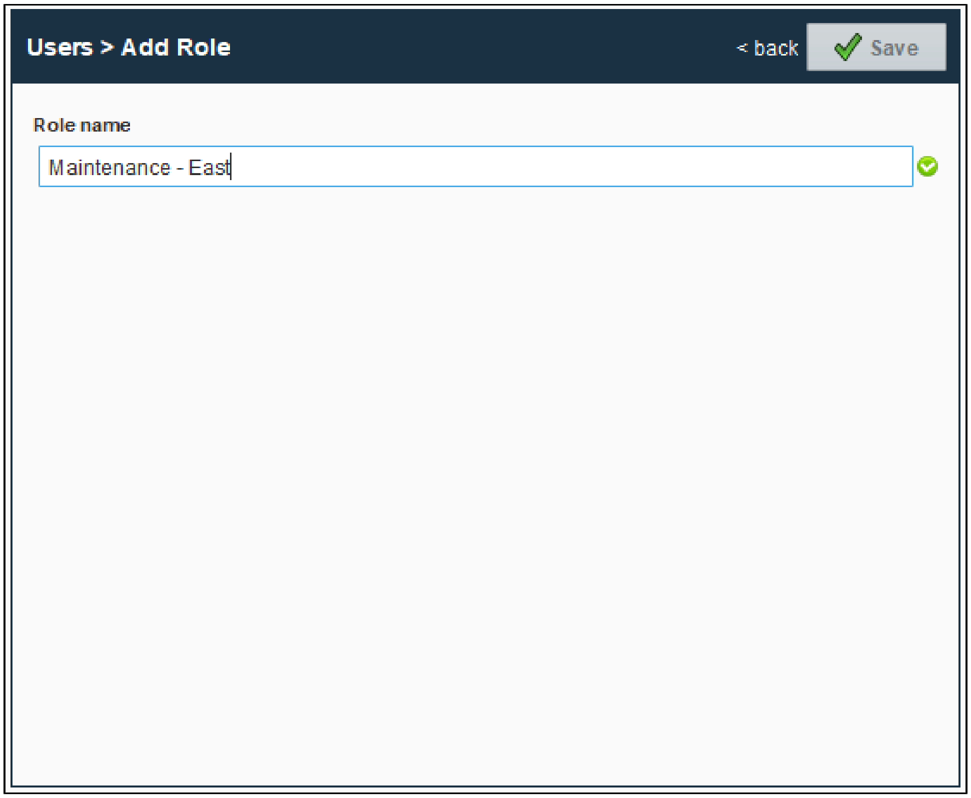
Language: English

Notes:

Roles:
- □ Administration
- □ Maintenance
- □ ReadWrite
- □ Supervisor

Type	Value

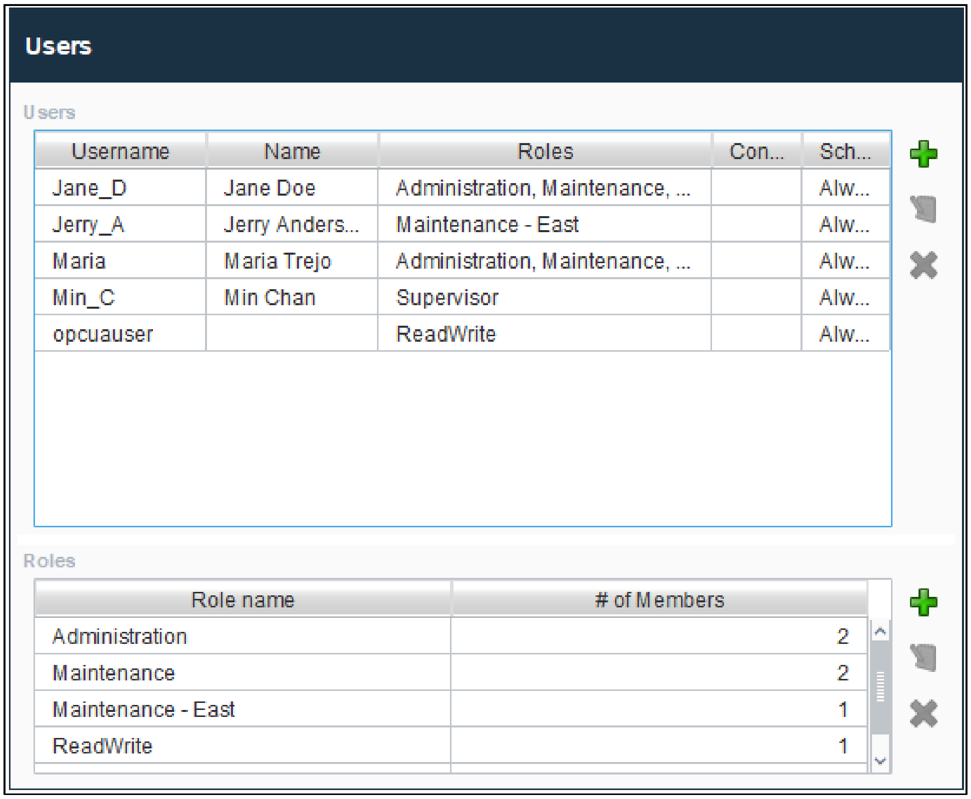
7. To add a new role, make sure the Designer is in **Preview Mode**. Click the the plus ✚icon next to the role. The Add Role window will open.
8. Enter the name of the new role. Click **Save**.



Role name

Maintenance - East

9. Now you can see the user and role that were just added in the User Management window.



Username	Name	Roles	Con...	Sch...
Jane_D	Jane Doe	Administration, Maintenance, ...		Alw...
Jerry_A	Jerry Anders...	Maintenance - East		Alw...
Maria	Maria Trejo	Administration, Maintenance, ...		Alw...
Min_C	Min Chan	Supervisor		Alw...
opcuauser		ReadWrite		Alw...

Role name	# of Members
Administration	2
Maintenance	2
Maintenance - East	1
ReadWrite	1

Save Failed. You are not authorized...

By default, changes to the system's user source may not be made from this component. This prevents users from locking themselves out of the Gateway, or give themselves access to the Gateway.

However, this behavior can be overridden from the Gateway: **Config > System > Gateway Settings** page, and setting the checkbox for the **Allow User Admin** property. This allows for the administration of the Gateway's system user source from the Designer and the Client. Unless this is enabled, the Vision Module's User Management component is prevented from modifying the Gateway system's selected user source and you will see an error at the bottom of the component if it is attempted.

Alternatively, you can simply have a separate User Source for the Gateway. This allows you to have a User Source containing all users that should have client access, and a Gateway-specific User Source that allows access to the Gateway and Designer. This would potentially entail changing the System's User Source: **Config > System > Gateway Settings > System User Source** in the Gateway Webpage.

Related Topics ...

- User Sources
- Internal Authentication
- User Management
- User Schedules

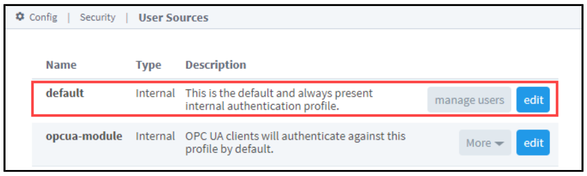
Internal Authentication

Internal User Sources

An Internal type User Source stores user information internally in the Gateway's database. This means that Internal User Sources are included in Gateway Backup files, and don't require an external SQL database, or other external user management system.

When Ignition is first installed, the default User Source that initially grants access to the Gateway and Design is an Internal type User Source. You can, of course, continue to use this default internal User Source for your project(s), or you may choose to use other User Sources instead.

The Internal User Source is fully manageable from within Ignition. You can access User Sources from the Gateway Webpage under the **Config** section, **Security > User, Roles**, and click the **edit** button.



Config | Security | User Sources

Name	Type	Description
default	Internal	This is the default and always present internal authentication profile.
opcua-module	Internal	OPC UA clients will authenticate against this profile by default.

Internal Authentication

Watch the Video

Property Reference

This section details Internal User Source properties, organized by category.

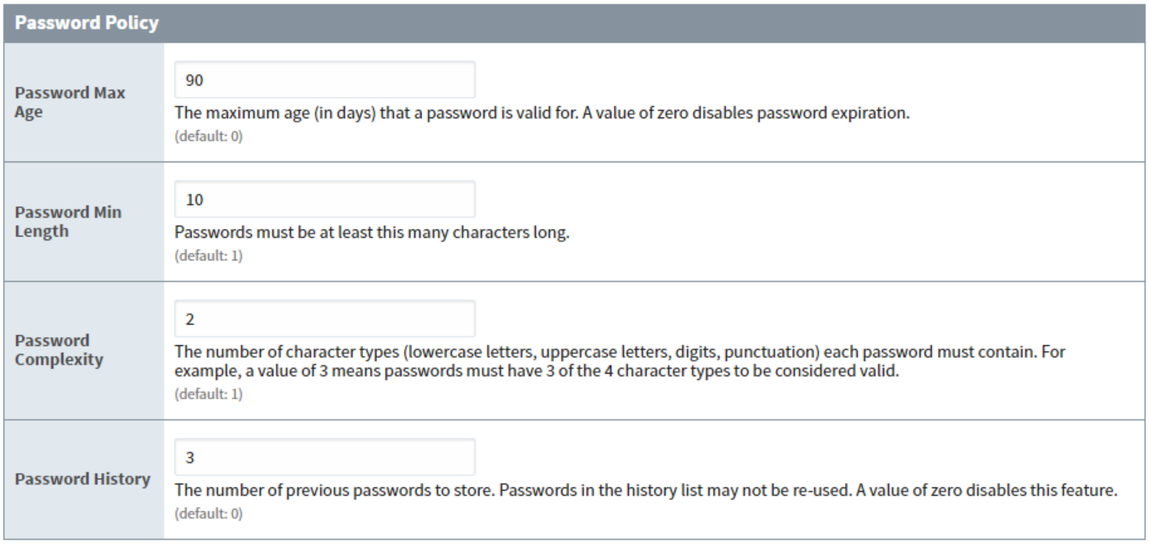
Main Properties

Details on the Main Properties can be found on the User Sources page.

Password Policies Properties

The Internal User Source has password policies that are configurable from within the Gateway to provide an extra layer of security by ensuring that good password practices are used.

1. From the **Config** tab in the **Gateway Webpage**, select **Users, Roles**.
2. Click the Edit button for the User Source you want to update.
3. Scroll down to the Password Policy section. You can change the default password policies by entering the appropriate password values to support your password policies.



Password Policy

Password Max Age	90 The maximum age (in days) that a password is valid for. A value of zero disables password expiration. (default: 0)
Password Min Length	10 Passwords must be at least this many characters long. (default: 1)
Password Complexity	2 The number of character types (lowercase letters, uppercase letters, digits, punctuation) each password must contain. For example, a value of 3 means passwords must have 3 of the 4 character types to be considered valid. (default: 1)
Password History	3 The number of previous passwords to store. Passwords in the history list may not be re-used. A value of zero disables this feature. (default: 0)

Below is a description of the Password Policy properties.

Name	Description
Password Maximum Age	The maximum age in days that the password will still be valid. After the number of days has past, when the user tries to login, it will prompt them to change their password. A value of 0 will disable this feature.
Password Minimum Length	The minimum amount of characters that a password must contain to be considered valid. If the user tries to make a shorter password, it will not allow it, and let them know that it does not meet the minimum length requirements.
Password Complexity	This determines how complex a password must be. There are four character types: lowercase letter, uppercase letter, digits, and special characters. The value here determines how many of those character types must be present at least once in the password for it to be considered valid.
Password History	Determines the number of previously used passwords to store. When users make a new password, old passwords can not be re-used. A value of zero disables this functionality. History is only stored while this setting is turned on, so any passwords used while this is off can be re-used when history is turned back on.

Related Topics ...

- Database Authentication
- Active Directory Authentication
- AD Internal Hybrid Authentication
- AD Database Hybrid Authentication
- Identity Providers
- User Management
- Managing Users and Roles
- Security

Database Authentication

Database User Source

The Database Authentication type uses an external database instead of storing data inside Ignition. Managing users is done via direct interaction with the database. This section addresses how to setup a database user source. The Database Authentication type requires you have a connection to an existing database, like SQL Server, Oracle, or MySQL. It stores all users, roles, schedules, and and more in the database, and uses queries to check login credentials. When you create a database user source, you have the option of setting it up in Automatic or Manual mode.

Automatic Mode

In Automatic mode, Ignition will create and manage the database tables for you. You can specify a prefix for the tables that are created automatically for you, but their names after the prefix are chosen by the user source. In this mode, the user source will be fully manageable in Ignition.

Manual Mode

In Manual mode, you must provide SQL queries for various functions of the user source. In this mode, the user source will not be manageable from the Gateway or the Clients. You'll have to manage the users directly through the database. Examples for each of the queries are given on the user source setup page. Read each query description carefully to make sure you design your queries to return all the columns that are defined in the query's description as shown below.

On this page ...

INDUCTIVE UNIVERSITY

Database Authentication

Watch the Video

Property Reference

Database User Sources have the following properties, organized by category

Main Properties

Details on the Main Properties can be found on the User Sources page. The Database User Source also has the following properties:

Name	Description
Database	The database connection this User Source will retrieve user information from.
Mode	How the Gateway should manage the database tables. Has two settings: **Automatic:** The gateway will automatically create the database tables necessary, and all interactions with the table will use the built-in queries. When this option is set, the **Tablename Prefix** property is utilized. **Manual:** The Gateway will not automatically create any database tables, nor will it automatically modify users or roles. When set to manual, it is assumed that you want to manually write the queries that update the tables, or are utilizing another system. When Mode is set to this option, the **Manual Mode** properties are used to determine how the Gateway should query user data.

Automatic Mode Properties

Name	Description
Tablename Prefix	When set to **Automatic** mode, this property determines the prefix that will be used on all automatically created tables. Useful when multiple database User Sources are connected to the same database scheme.

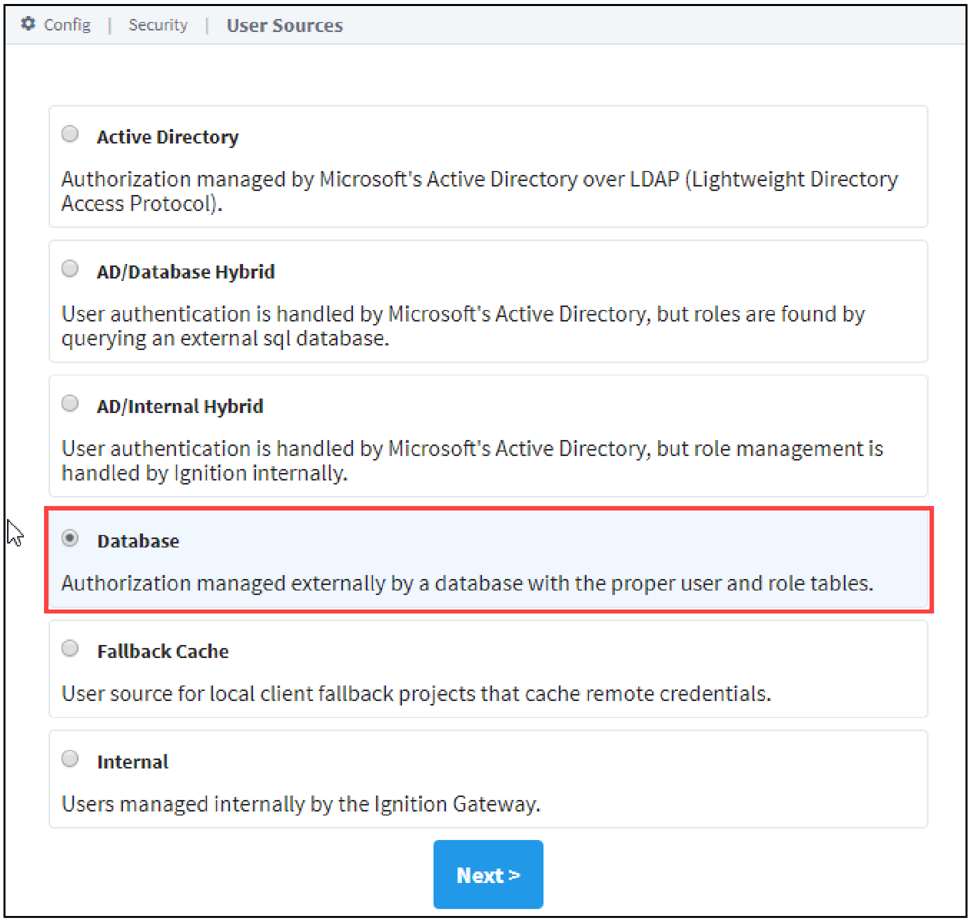
Manual Mode

Name	Description
Authentication Query	A query that **must** return a row if the given username and password combination provided is valid. The query will run as a prepared statement, so use the question mark character (?) to represent username first and then password. The returned row may contain the user's basic properties under the column names: [firstname, lastname, schedule, language, notes] Note that the Gateway will pass both the username and password the user typed in, so this query **MUST** utilize exactly two question marks, otherwise an exception will occur.
List Roles Query	A query that returns all possible roles that any user could be a member of. The role names must be returned in the first column of the query's results.
User's Roles Query	A query that returns all of the roles that the provided user belongs to. The roles must be strings and must be in the first column of the query's results. The query will be run as a prepared statement with one parameter: the username.
List Users Query	A query that returns a row containing each username. There must be at least one column: the username. Other columns are optional, supported columns are: [username, firstname, lastname, schedule, language, notes].
Contact Info Query	A query that returns all of the contact info for the user. The first column must be the contact type, the second column the contact value. Optional, may be blank.
Schedule Adjustment Query	A query that returns the upcoming schedule adjustments for the user. This property is optional, and may be left blank. The results set expects the following columns: • **Start**(date) • **End**(date) • **Available**(boolean) • **Note**(string)
Extra Properties Query	A query that returns name, value pairs of extra properties for the user. Will be run with one parameter: the username. Optional, may be blank.

To Create a Database User Source

1. On the Gateway Webpage under the **Config** tab, go **Security > Users, Roles**.
   The User Sources page will be displayed. Click the blue arrow, **Create new User Source.**

2. Choose the **Database** authentication type, and click **Next**.



Config | Security | User Sources

○ **Active Directory**

Authorization managed by Microsoft's Active Directory over LDAP (Lightweight Directory Access Protocol).

○ **AD/Database Hybrid**

User authentication is handled by Microsoft's Active Directory, but roles are found by querying an external sql database.

○ **AD/Internal Hybrid**

User authentication is handled by Microsoft's Active Directory, but role management is handled by Ignition internally.

● **Database**

Authorization managed externally by a database with the proper user and role tables.

○ **Fallback Cache**

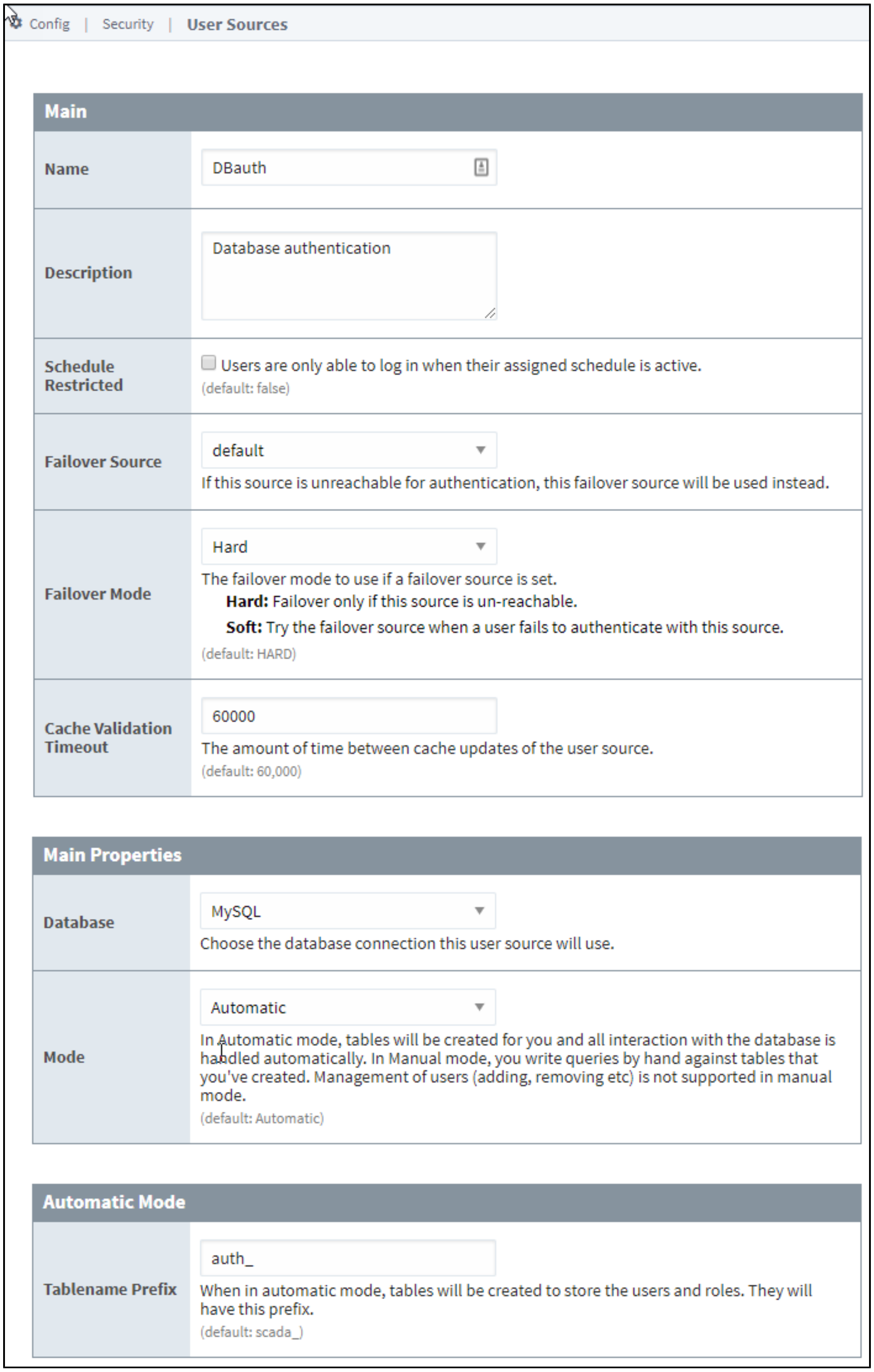
User source for local client fallback projects that cache remote credentials.

○ **Internal**

Users managed internally by the Ignition Gateway.

3. The New User Source window will open. Some properties are optional, but if you're using Automatic mode, enter the following properties as appropriate.

- **Name**: DBAuth - name of the user source.
- **Failover Source**: default - failover user source ('default' is the internal user source).
- **Failover Mode**: Hard - if the source is unreachable, then use the failover source. (Can choose the Hard or Soft option).
- **Database**: MySQL - external database.
- **Mode:** Automatic - tables in the external database will be automatically created when needed.
- **Tablename Prefix**: 'auth_' is the prefix for all the tables that get created. (You can leave this field blank, but if you use a prefix when the tables get created, they will contain the specified prefix in their name).
- When finished, click **Create New User Source**.



Config | Security | User Sources

Main

Name	DBauth
Description	Database authentication
Schedule Restricted	□ Users are only able to log in when their assigned schedule is active. (default: false)
Failover Source	default ▾ If this source is unreachable for authentication, this failover source will be used instead.
Failover Mode	Hard ▾ The failover mode to use if a failover source is set. **Hard:** Failover only if this source is un-reachable. **Soft:** Try the failover source when a user fails to authenticate with this source. (default: HARD)
Cache Validation Timeout	60000 The amount of time between cache updates of the user source. (default: 60,000)

Main Properties

Database	MySQL ▾ Choose the database connection this user source will use.
Mode	Automatic ▾ In Automatic mode, tables will be created for you and all interaction with the database is handled automatically. In Manual mode, you write queries by hand against tables that you've created. Management of users (adding, removing etc) is not supported in manual mode. (default: Automatic)

Automatic Mode

Tablename Prefix	auth_ When in automatic mode, tables will be created to store the users and roles. They will have this prefix. (default: scada_)

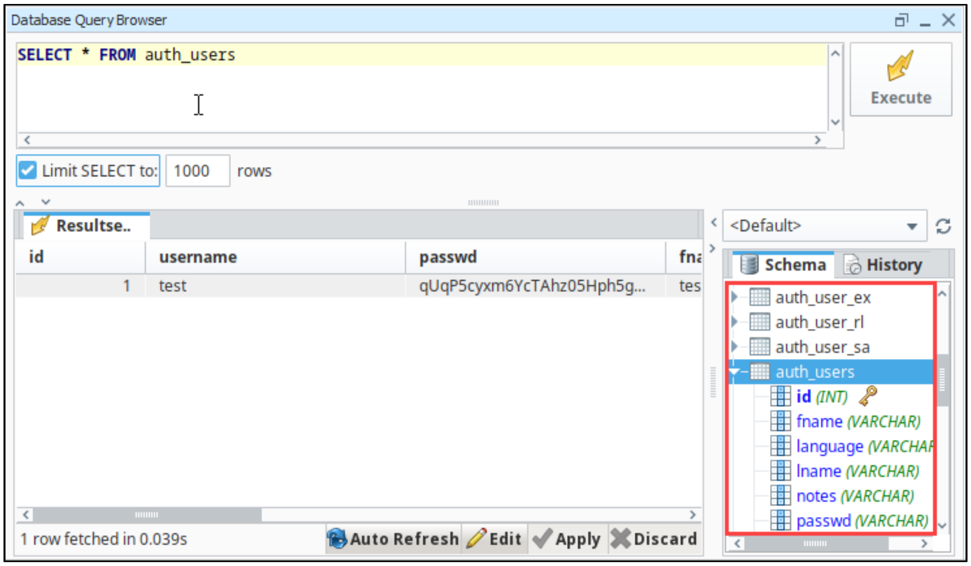
Table Creation

The tables in the database will not be created in the database until they are needed. For example, as soon as a user or role is added, the associated tables will automatically get created.

4. Now that your Authentication profile is created, add a user. On the right, click on the **More > Manage Users** link. Click on the **Add User** link and fill in the required fields.

5. Now that your tables are created we can verify them. To view the tables, go into **Designer** and from the menu bar, select **Tools >** **Database Query Browser**.
   You will see all the tables that were created beginning with 'auth_' when the user and role get created.

6. Double click on any of tables beginning with **'auth_'**, and click **Execute**. In this example, you will see the tables associated with 'roles' and 'users' displayed in the Schema area.



```
SELECT * FROM auth_users
```

☑ Limit SELECT to: 1000 rows

id	username	passwd	fna
1	test	qUqP5cyxm6YcTAhz05Hph5g...	tes

1 row fetched in 0.039s

- auth_user_ex
- auth_user_rl
- auth_user_sa
- auth_users
  - id (INT)
  - fname (VARCHAR)
  - language (VARCHAR
  - lname (VARCHAR)
  - notes (VARCHAR)
  - passwd (VARCHAR)

Related Topics ...

- Internal Authentication
- Active Directory Authentication
- AD Internal Hybrid Authentication
- AD Database Hybrid Authentication
- Identity Providers
- Database Query Browser

Active Directory Authentication

Active Directory User Source

The Active Directory Authentication profile uses Microsoft's Active Directory over **LDAP** (Lightweight Directory Access Protocol) to store all the users, roles, and more that make up an Authentication profile. Active Directory **Groups** are used for Ignition's **roles** and user-role mappings.

While using an Active Directory User Source, administration of users and roles is through Active Directory itself, and not manageable within Ignition. Thus adding new users to an Active Directory User Source, or modifying pre existing users, requires the modifications be made from Active Directory, usually through an AD Administrator.

On this page

...

Property Reference

Active Directory User Sources have the following properties, organized by category

Certain properties in the Active Directory User Source allow you to filter users, such as the **User List Filter**. These filters only determine which users will be displayed on screen. They are not authentication filters, so even if a user does not show in the list they can still authenticate and may have access to unintended areas. Be sure to configure project security appropriately to prevent this from happening!

Main Properties

Details on the Main Properties can be found on the User Sources page.

Active Directory Properties

Name	Description
Domain	The Windows Domain your active Active Directory server is running on. If you aren't sure of your domain, ask your network administrator. Leave blank to set advanced properties manually.
Gateway Username	The login name for the Gateway to use when querying Active Directory. Used for retrieving the list of users and roles via LDAP.
Password	The password for the above username.
Password	Re-type password for verification.
Primary Domain Controller Host	The IP address or hostname of your primary domain controller. Example: "192.168.1.4" or "MainServer"
Primary Domain Controller Port	The port number for the primary domain controller's LDAP interface.
Secondary Domain Controller Host	The IP address or hostname of your secondary domain controller (optional). Example: "192.168.1.4" or "MainServer"
Secondary Domain Controller Port	The port number for the secondary domain controller's LDAP interface.
SSO Enabled	Whether or not to use Single-Sign-On (SSO) to authenticate AD users. This gives you the ability to log into the Client or Designer automatically with the password you logged into windows with. For Client SSO login, each project must also have the **SSO Login** Project Property enabled. For Designer SSO login, **Allow Designer SSO** must be set in the Gateway Settings.

SSO Domain	The domain that Windows users must match in order to use SSO. If blank, the main "Domain" property will be used.

LDAP Search Properties

Name	Description
Username Prefix	This prefix will be prepended to the username before an Active Directory bind is attempted for authentication.
Username Suffix	This suffix will be appended to the username before an Active Directory bind is attempted for authentication.
Automatic Suffix	If this option is checked, and the suffix is left blank, then the suffix will automatically be assigned a value of "@<domain>".
User Search Base	The base folder to search for users under, such as: **DC=MyCompany,DC=com** The entire subtree under this folder will be searched using the User Search Filter. Multiple subtrees can be specified by putting them in parenthesis, like so: **(OU=Administrators,DC=MyCompany,DC=com)(OU=Operators,DC=MyCompany,DC=com)**
User Search Filter	The LDAP search filter that will be used to find a specific user. Use the placeholder {0} as a standin for the login name.
User List Filter	The LDAP search filter used when querying for the list of all users. Should restrict the type to user.
User Name Attribute	The attribute on the User object to define the username.
User Role Attribute	Attributes of this name on the User object will define the user's roles.
Role Name Attribute	The attribute of this name on the Role object will define the role's name. Leave blank to use the raw value of the attribute defined by the **User Role Attribute** property.
Full Name Attribute	The attribute on the User object to define the full name of the user.
Phone Attribute	The attribute name on the user object that represents the user's phone number.
Email Attribute	The attribute name on the user object that represents the user's email address.
SMS Attribute	The attribute name on the user object that represents the phone number that this user receives text messages on.
Read Timeout	The read timeout in milliseconds for LDAP operations.
Results Page Size	The number of entries returned per page of results in a query.
Role Search Base	The base folder to search for roles under, such as: **OU=Roles,DC=MyCompany,DC=com**

	The entire subtree under this folder will be searched using the Role Search Filter. If you specify the root of your tree structure, the search may take a very long time. Multiple subtrees can be specified by putting them in parenthesis, like so: **(OU=Builtin,DC=MyCompany,DC=com)(OU=Users,DC=MyCompany,DC=com)** If you leave this blank the whole subtree of the domain controller will be searched.
Role Search Filter	The LDAP search filter that will be used to locate roles.
Allow Anonymous	Determines whether the Gateway will accept blank usernames and passwords for authentication. Note that this check takes place on the Gateway, prior to handing off any credentials to the AD server. If **Security Authentication** is set to None, then this property should be enabled, otherwise, blank passwords will be rejected by the Gateway. If true, authentication attempts with blank passwords will be passed through to LDAP, which may choose to accept them.
Use SSL	Works in conjunctions with the **Domain Controller Host** and **Domain Controller Port** properties in the Active Directory Properties section. Disable to use "ldap://" protocol, enable to use "ldaps://"
Security Protocol	Specifies the security protocol between the Gateway and AD server. The following options are available: **AUTO:** No security protocol is explicitly used or requested by the Gateway. **SSL:** SSL should be used for the connection.
Security Authentication	This property specifies how usernames and passwords are used to bind to LDAP. The following options are available: **AUTO:** Unspecified from the Gateway side, meaning the LDAP implementation will choose. **NONE:** Anonymous access. **SIMPLE:** Plaintext username and passwords will be used. **STRONG:** Usernames and passwords will be encrypted. The following feature is new in Ignition version **8.0.10** Click here to check out the other new features **SASL**: Simple Authentication and Security Layer. See the SASL Properties table below for additional SASL authentication configuration settings.

SASL Properties

The following feature is new in Ignition version **8.0.10**
Click here to check out the other new features

These settings are utilized when Security Authentication is set to **SASL**.

Name	Description
Mechanism	An ordered list of space-separated mechanism names. The LDAP provider will use the first mechanism for which it finds an implementation. A blank value will leave this setting unspecified. (Default is `DIGEST-MD5 CRAM-MD5`.)
Realm	A realm defines the namespace from which the user is selected. A blank value will leave this setting unspecified. This setting will only be used by mechanisms which support it. (Default is blank.)
Quality of Protection	A comma-separated list of Quality-of-Protection (QoP) values, the order of which specifies the preference order. There are three well-known values: "auth" (authentication only), "auth-int" (authentication with integrity protection), and "auth-conf" (authentication with integrity and privacy protection). A blank value will leave this setting unspecified. This setting will only be used by mechanisms which support it. (Default is `auth-conf,auth-int,auth.`)
Protection Strength	A comma-separated list of privacy protection strength values, the order of which specifies the preference order. The three possible strength values are "low", "medium", and "high". A blank value will leave this setting unspecified. This setting will only be used by mechanisms which support it. (Default is `high,medium,low`.)
Mutu	Enable or disable mutual authentication. This setting will only be used by mechanisms which support it. (Default is disabled.)

al Authentication	

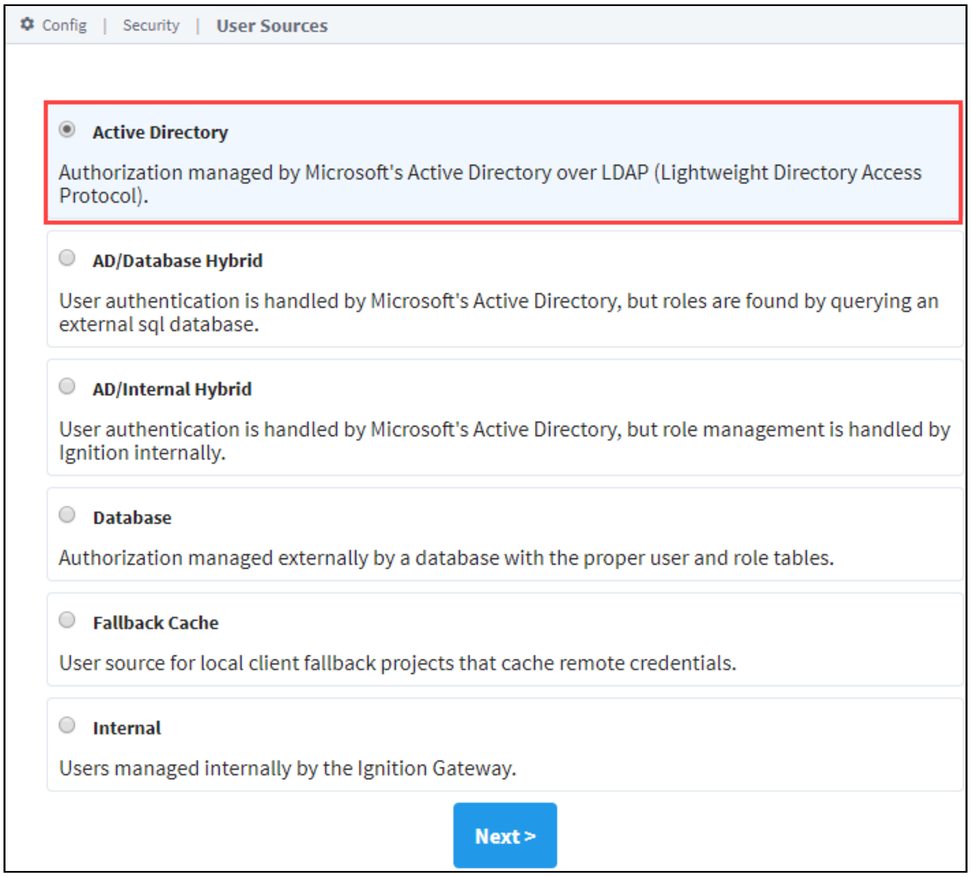
To Create an Active Directory User Source

To configure an Active Directory User Source, you must specify the host that is acting as your primary domain controller. You can also use a secondary domain controller in case the primary is unavailable. You'll also need to specify the name of the domain and credentials for the Gateway itself to use: the Gateway needs a user account to interact with the AD server, even when it's simply querying for a list of roles.

> **May need to contact your internal IT Department**
>
> When using Active Directory User Source, you may need to consult with your internal IT Department to get the required information to complete your user source setup. These settings are common to AD (not specific to Ignition), and your IT department will know what values to supply to each property.

1. On the Gateway Webpage, under the **Config** tab, go to **Security > Users, Roles**.
   The User Sources page will be displayed. Click the blue arrow, **Create new User Source**.

2. Choose the **Active Directory** authentication type, and click **Next**.



● **Active Directory**
Authorization managed by Microsoft's Active Directory over LDAP (Lightweight Directory Access Protocol).

○ **AD/Database Hybrid**
User authentication is handled by Microsoft's Active Directory, but roles are found by querying an external sql database.

○ **AD/Internal Hybrid**
User authentication is handled by Microsoft's Active Directory, but role management is handled by Ignition internally.

○ **Database**
Authorization managed externally by a database with the proper user and role tables.

○ **Fallback Cache**
User source for local client fallback projects that cache remote credentials.

○ **Internal**
Users managed internally by the Ignition Gateway.

3. The New User Source window will open. Note that some properties are optional. In the very least, you must specify the following: **Domain**, **Gateway Username**, **Password**, **Primary Domain Controller Host**.
4. Click the **Create New User Source** button to create the User Source.

Related Topics ...

- Internal Authentication
- Database Authentication

- AD Internal Hybrid Authentication
- AD Database Hybrid Authentication
- Identity Providers
- User Sources
- Project Security in Designer

AD Internal Hybrid

AD/Internal User Source

On this page ...

The Active Directory/Internal Hybrid authentication profile type combines the Internal User Source type with the Active Directory User Source type. Active Directory is used to find all of the users, and to check their credentials when they attempt to log in. However, it allows assigning of roles, contact info, and other meta-information about a user through Ignition, then stores all this information as if it were an Internal User Source. This way, Active Directory can be consulted to see if a username/password is valid, but the management of roles does not require coordination with your IT Department, who typically controls the Active Directory system. This "best of both worlds" approach is popular for many users of Active Directory.

The AD/Internal Hybrid User Source is partially manageable in Ignition.   Users cannot be added or removed, and their usernames and passwords cannot be changed. This is because this information resides in Active Directory, not within Ignition. Other information, such as user roles, contact info, schedules, are manageable in Ignition.

Gateway Settings

Before you can use the User Management component to manage roles, contact info, etc., you first have to go into Gateway Settings, and mark the checkbox to **'Allow User Admin.'** This allows for the administration of the Gateway's system user source from the Designer and the Client. Unless this is enabled, the Vision Module's User Management component is prevented from modifying the Gateway system's user source.

INDUCTIVE UNIVERSITY

AD Internal Hybrid

Watch the Video

Property Reference

This User Source shares many properties with the AD User Source. Please see the Active Directory Authentication page for a list of properties.

Creating an AD/Internal Hybrid User Source

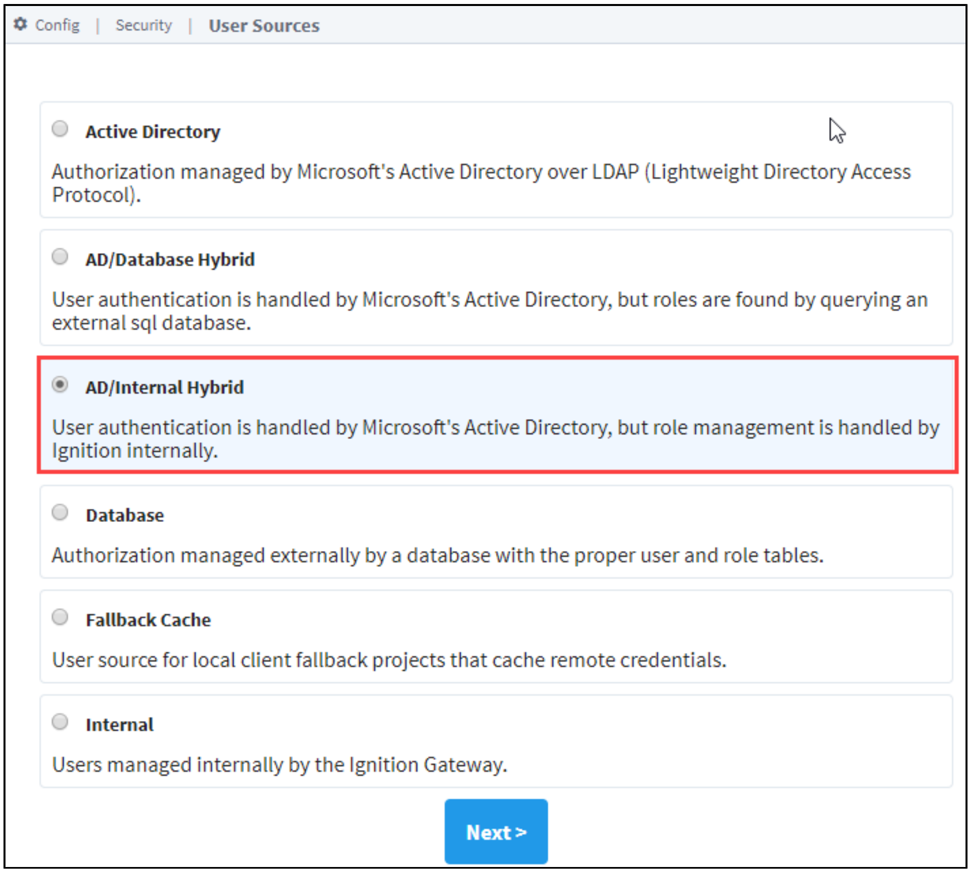
To set up an AD/Internal Hybrid User Source, you must specify the host that is acting as your primary domain controller. You can also use a secondary domain controller in case the primary is unavailable. You'll also need to specify the name of the domain and credentials for the Gateway itself to use for authentication for when it queries the list of roles.

May need to contact your internal IT Department for...

When using AD/Internal Hybrid User Source, you may need to consult with your internal IT Department to get the required information to complete your user source setup.

1. On the Gateway Webpage, under the **Config** tab, go to **Security > Users, Roles**.
   The User Sources page will be displayed. Click the blue arrow, **Create new User Source**.

2. Choose the **AD/Internal Hybrid** authentication type, and click **Next**.



Config | Security | User Sources

○ **Active Directory**

Authorization managed by Microsoft's Active Directory over LDAP (Lightweight Directory Access Protocol).

○ **AD/Database Hybrid**

User authentication is handled by Microsoft's Active Directory, but roles are found by querying an external sql database.

● **AD/Internal Hybrid**

User authentication is handled by Microsoft's Active Directory, but role management is handled by Ignition internally.

○ **Database**

Authorization managed externally by a database with the proper user and role tables.

○ **Fallback Cache**

User source for local client fallback projects that cache remote credentials.

○ **Internal**

Users managed internally by the Ignition Gateway.

3. The New User Source window will open. Some properties are optional depending on how you set up your profile. Details on the Main Properties can be found on the User Sources page. Active Directory properties are listed in the table below.
4. Click **Create New User Source** to save the new user source.

Active Directory Properties

Name	**Description**
Domain	The Windows Domain your active Active Directory server is running on. If you aren't sure of your domain, ask your network administrator. Leave blank to set advanced properties manually.
Primary Domain Controller Host	The IP address or hostname of your primary domain controller. Example: "192.168.1.4" or "MainServer"
Primary Domain Controller Port	The port number for the primary domain controller's LDAP interface.
List Users from Active Directory	If true, Active Directory will be queried for the list of all users. If false, users must be added manually. (Default is true.)
Gateway Username	The login name for the Gateway to use when querying Active Directory. Used for retrieving the list of users and roles via LDAP.
Password	The password for the above username.
Password	Re-type password for verification.
SSO Enabled	Whether or not to use Single-Sign-On (SSO) to authenticate AD users. Note that projects must also have this option enabled for SSO to work. (Default is false.)
SSO Domain	The domain that Windows users must match in order to use SSO. If blank, the main "Domain" property will be used.

Advanced Properties

Name	Description
Secondary Domain Controller Host	The IP address or hostname of your secondary domain controller (optional). Example: "192.168.1.5" or "BackupServer"
Secondary Domain Controller Port	The port number for the secondary domain controller's LDAP interface. (Default: 389)
Read Timeout	The read timeout in milliseconds for LDAP operations. (Default is 60,000)
Results Page Size	The number of entries returned per page of results in a query. (Default is 1,000)
User Listing Base	The base folder to search for users under, such as "DC=MyCompany,DC=com". The entire subtree under this folder will be searched using the User List Filter. Multiple subtrees can be specified by putting them in parenthesis, like so: "(OU=Administrators,DC=MyCompany,DC=com)(OU=Operators,DC=MyCompany,DC=com)" If you leave this blank the whole subtree of the domain controller will be searched.
User List Filter	The LDAP search filter used when querying for the list of all users. Should restrict the type to user. (Default is `(&(objectClass=user)(!(objectClass=computer))).)`
User Search Filter	The LDAP search filter to use to find a specific user. Use the placeholder {0} as a standin for the login name. (Default is `(&(objectClass=user)(sAMAccountName={0})))`
Username Attribute	The attribute on the User object to define the username. (Default is sAMAccountName.)
Username Prefix	This prefix will be prepended to the username before an Active Directory bind is attempted for authentication.
Username Suffix	This suffix will be appended to the username before an Active Directory bind is attempted for authentication.
Automatic Suffix	If this option is checked, and the suffix is left blank, then the suffix will automatically be assigned a value of "@<domain>" (Default is true.)
Allow Anonymous	If true, authentication attempts with blank passwords will be passed through to LDAP, which may choose to accept them. (Default is false.)
Use SSL	Disable to use "ldap://" protocol, enabled to use "ldaps://" (Default is false.)
Security Protocol	Auto or SSL. (Default is Auto.)
Security Authentication	This property specifies how usernames and passwords are used to bind to LDAP. The following options are available: **AUTO:** Unspecified from the Gateway side, meaning the LDAP implementation will choose. **NONE:** Anonymous access. **SIMPLE:** Plaintext username and passwords will be used. **STRONG:** Usernames and passwords will be encrypted. The following feature is new in Ignition version **8.0.10** Click here to check out the other new features **SASL** : Simple Authentication and Security Layer. See the SASL Properties table below for additional SASL authentication configuration settings.

SASL Properties

The following feature is new in Ignition version **8.0.10**
Click here to check out the other new features

These settings are utilized when Security Authentication is set to **SASL**.

Name	Description
Mechanism	An ordered list of space-separated mechanism names. The LDAP provider will use the first mechanism for which it finds an implementation. A blank value will leave this setting unspecified. (Default is `DIGEST-MD5 CRAM-MD5`.)
Realm	A realm defines the namespace from which the user is selected. A blank value will leave this setting unspecified. This setting will only be used by mechanisms which support it. (Default is blank.)
Quality of Protection	A comma-separated list of Quality-of-Protection (QoP) values, the order of which specifies the preference order. There are three well-known values: "auth" (authentication only), "auth-int" (authentication with integrity protection), and "auth-conf" (authentication with integrity and privacy protection). A blank value will leave this setting unspecified. This setting will only be used by mechanisms which support it. (Default is `auth-conf,auth-int,auth.`)
Protection Strength	A comma-separated list of privacy protection strength values, the order of which specifies the preference order. The three possible strength values are "low", "medium", and "high". A blank value will leave this setting unspecified. This setting will only be used by mechanisms which support it. (Default is `high,medium,low`.)
Mutual Authentication	Enable or disable mutual authentication. This setting will only be used by mechanisms which support it. (Default is disabled.)

Related Topics ...

- Internal Authentication
- Database Authentication
- Active Directory Authentication
- AD Database Hybrid Authentication
- Identity Providers
- User Management Component

AD Database Hybrid

AD/Database User Source

This AD/Database Hybrid User Source is not manageable from within Ignition. Users/passwords must be administered through Active Directory, and roles, contact info, and so on, must be administered directly through the database. The way AD/Database Hybrid works, is it has all the same information requirements as the other authentication profiles, but it also has a number of Database properties. You need to specify a database (i.e., MySQL) to store information, and set up queries that you want to use. You must also specify the host that is acting as your primary domain controller, and a secondary domain controller in case the primary is unavailable. You'll also need to specify the name of the domain and credentials for the Gateway itself to use for authentication when it queries the list of roles.

On this page ...

Property Reference

This User Source shares many properties with both the AD User Source and Database User Source.

- See the Active Directory Authentication page for a list of Active Directory User Source related properties.
- See the Database Authentication page for a list of Database User Source related properties.

INDUCTIVE UNIVERSITY

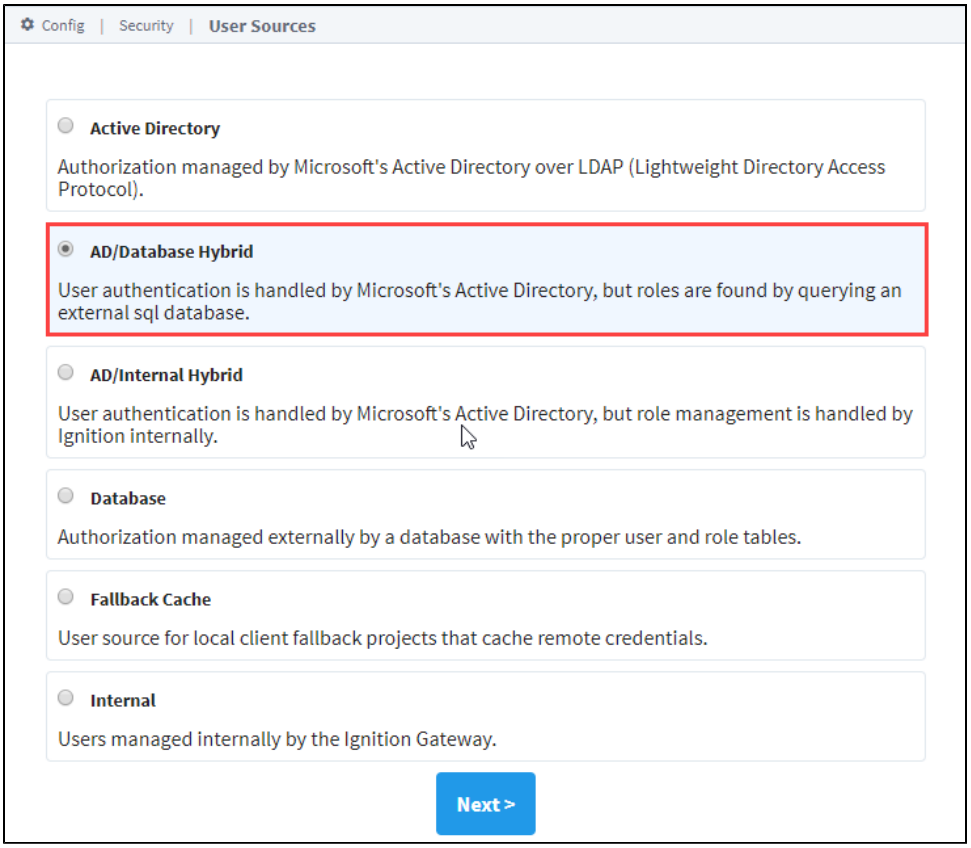
AD Database Hybrid

Watch the Video

Creating an AD/Database Hybrid User Source

1. On the Gateway Webpage, under the **Config** tab, go to **Security > Users, Roles**.
   The User Sources page will be displayed. Click the blue arrow, **Create new User Source**.

2. Choose the **AD/Database Hybrid** authentication type, and click **Next**.



○ **Active Directory**

Authorization managed by Microsoft's Active Directory over LDAP (Lightweight Directory Access Protocol).

● **AD/Database Hybrid**

User authentication is handled by Microsoft's Active Directory, but roles are found by querying an external sql database.

○ **AD/Internal Hybrid**

User authentication is handled by Microsoft's Active Directory, but role management is handled by Ignition internally.

○ **Database**

Authorization managed externally by a database with the proper user and role tables.

○ **Fallback Cache**

User source for local client fallback projects that cache remote credentials.

○ **Internal**

Users managed internally by the Ignition Gateway.

> **May need to contact your internal IT Department for..**
>
> When using AD/Database Hybrid User Source, you may need to consult with your internal IT Department to get the required information to complete your user source setup.

3. The New User Source window will open. Some properties are optional depending on how you set up your profile. Details on the Main Properties can be found on the User Sources page. The Active Directory Properties and Database properties are listed in the tables below.
4. Click **Create New User Source** to save the new user source.

Active Directory Properties

Name	Description
Domain	The Windows Domain your active Active Directory server is running on. If you aren't sure of your domain, ask your network administrator. Leave blank to set advanced properties manually.
Primary Domain Controller Host	The IP address or hostname of your primary domain controller. Example: "192.168.1.4" or "MainServer"
Primary Domain Controller Port	The port number for the primary domain controller's LDAP interface.
List Users from Active Directory	If true, Active Directory will be queried for the list of all users. If false, users must be added manually. (Default is true.)
Gateway Username	The login name for the Gateway to use when querying Active Directory. Used for retrieving the list of users and roles via LDAP.
Password	The password for the above username.
Password	Re-type password for verification.
SSO Enabled	

	Whether or not to use Single-Sign-On (SSO) to authenticate AD users. Note that projects must also have this option enabled for SSO to work. (Default is false.)
SSO Domain	The domain that Windows users must match in order to use SSO. If blank, the main "Domain" property will be used.

Database Properties

Name	Description
Database	Dropdown list. Choose the database connection this authentication profile will use.
User Properties Query	A query that returns the basic properties for a single user. Supported return columns are [username, firstname, lastname, schedule, language, notes].
Role List Query	A query that returns all possible roles that any user could have. The role names must be returned in the first column of the query's results
User's Roles Query	A query that returns all of the roles that the provided user belongs to. The roles must be strings (i.e., the role names), and must be in the first column of the query's results. The username will be inserted into this query as a parameter.
Contact Info Query	A query that returns all of the contact info for the user. The first column must be the contact type, the second column the contact value, and the third column the name of a schedule. Optional, may be blank.
Schedule Adjustment Query	A query that returns the upcoming schedule adjustments for the user. Columns must be Start(date), End(date), Available (boolean), Note(string). Optional, may be blank.
Extra Properties Query	A query that returns name, value pairs of extra properties for the user. Will be run with one parameter: the username. Optional, may be blank.
List Users Query	A query that returns a row containing each username. Only used if "List Users from Active Directory" is false. There must be at least one column: the username. Other columns are optional, supported columns are: [username, firstname, lastname, schedule, language, notes].

Advanced Properties

Name	Description
Secondary Domain Controller Host	The IP address or hostname of your secondary domain controller (optional). Example: "192.168.1.5" or "BackupServer"
Secondary Domain Controller Port	The port number for the secondary domain controller's LDAP interface. Default: 389.
Read Timeout	The read timeout in milliseconds for LDAP operations. Default is 60,000.
Results Page Size	The number of entries returned per page of results in a query. Default is 1,000.
User Listing Base	The base folder to search for users under, such as "DC=MyCompany,DC=com". The entire subtree under this folder will be searched using the User List Filter. Multiple subtrees can be specified by putting them in parenthesis, like so: "(OU=Administrators,DC=MyCompany,DC=com)(OU=Operators,DC=MyCompany,DC=com)" If you leave this blank the whole subtree of the domain controller will be searched.
User List Filter	The LDAP search filter used when querying for the list of all users. Should restrict the type to user. Default is `(&(objectClass=user)(!(objectClass=computer)))`.
User	

Search Filter	The LDAP search filter to use to find a specific user. Use the placeholder {0} as a standin for the login name. Default is `(&(objectClass=user)(sAMAccountName={0})).`
Username Attribute	The attribute on the User object to define the username. Default is sAMAccountName.
Username Prefix	This prefix will be prepended to the username before an Active Directory bind is attempted for authentication.
Username Suffix	This suffix will be appended to the username before an Active Directory bind is attempted for authentication.
Automatic Suffix	If this option is checked, and the suffix is left blank, then the suffix will automatically be assigned a value of "@<domain>". Default is true.
Allow Anonymous	If true, authentication attempts with blank passwords will be passed through to LDAP, which may choose to accept them. Default is false.
Security Protocol	Auto or SSL. Default is Auto.
Security Authentication	This property specifies how usernames and passwords are used to bind to LDAP. The following options are available: **AUTO:** Unspecified from the Gateway side, meaning the LDAP implementation will choose. **NONE:** Anonymous access. **SIMPLE:** Plaintext username and passwords will be used. **STRONG:** Usernames and passwords will be encrypted. The following feature is new in Ignition version **8.0.10** Click here to check out the other new features **SASL**: Simple Authentication and Security Layer. See the SASL Properties table below for additional SASL authentication configuration settings.

SASL Properties

The following feature is new in Ignition version **8.0.10**
Click here to check out the other new features

These settings are utilized when Security Authentication is set to **SASL**.

Name	Description
Mechanism	An ordered list of space-separated mechanism names. The LDAP provider will use the first mechanism for which it finds an implementation. A blank value will leave this setting unspecified. (Default is `DIGEST-MD5 CRAM-MD5`.)
Realm	A realm defines the namespace from which the user is selected. A blank value will leave this setting unspecified. This setting will only be used by mechanisms which support it. (Default is blank.)
Quality of Protection	A comma-separated list of Quality-of-Protection (QoP) values, the order of which specifies the preference order. There are three well-known values: "auth" (authentication only), "auth-int" (authentication with integrity protection), and "auth-conf" (authentication with integrity and privacy protection). A blank value will leave this setting unspecified. This setting will only be used by mechanisms which support it. (Default is `auth-conf,auth-int,auth.`)
Protection Strength	A comma-separated list of privacy protection strength values, the order of which specifies the preference order. The three possible strength values are "low", "medium", and "high". A blank value will leave this setting unspecified. This setting will only be used by mechanisms which support it. (Default is `high,medium, low`.)
Mutual Authentication	Enable or disable mutual authentication. This setting will only be used by mechanisms which support it. (Default is disabled.)

Related Topics ...

- Internal Authentication
- Database Authentication
- Active Directory Authentication
- AD Internal Hybrid Authentication
- Identity Providers
- Verifying a User Source
- User Sources
- Database Query Browser

Fallback Cache Authentication

Overview

This User Source was developed specifically for a system that is using Local Vision Client Fallback, and allows you to cache the login credentials from a remote user source. This means your users can still log in with their normal username/password on a Local Vision Client Fallback project, even when the network connection is unavailable.

> Fallback Cache Authentication does not work with Perspective sessions, or with Identity Providers in general. This User Source will only function with Vision clients and user sources.

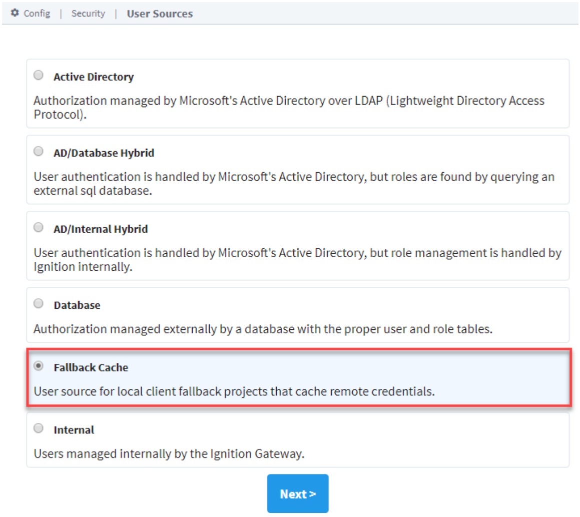
On this page

...

Creating the Fallback Cache User Source

The Fallback Cache User Source is created in a similar fashion to any other User Source:

1. On the **Gateway Webpage,** go to the **Config** tab. Select **Security > User, Roles.**
2. Click on the **Create new User Source...** link**.**
3. Select the Fallback Cache option and click the **Next** button.



Config | Security | User Sources

○ **Active Directory**
Authorization managed by Microsoft's Active Directory over LDAP (Lightweight Directory Access Protocol).

○ **AD/Database Hybrid**
User authentication is handled by Microsoft's Active Directory, but roles are found by querying an external sql database.

○ **AD/Internal Hybrid**
User authentication is handled by Microsoft's Active Directory, but role management is handled by Ignition internally.

○ **Database**
Authorization managed externally by a database with the proper user and role tables.

● **Fallback Cache**
User source for local client fallback projects that cache remote credentials.

○ **Internal**
Users managed internally by the Ignition Gateway.

4. Type in a name for the new User Source and click **Create New User Source**.
5. Details on the Main Properties can be found on the User Sources page. You can also set Cache Retention as follows.

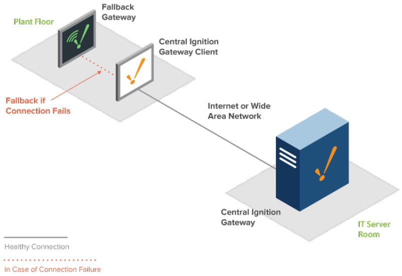
Fallback Cache Properties	
Cache Retention	Number of days that the cache will retain recently used credentials. This property determines the number of days credentials will be stored in the cache. Once exceeded, the credentials will be removed from the cache. (Default: 15)

6. Click the **Create New User Source** button.

Populating the Cache

Users and Roles can not be manually added to the Fallback Cache. Instead, they are automatically copied from remote Gateways. This type of User Source is normally configured on an Edge Gateway, but can be utilized on an Ignition Gateway.

- Before the Fallback Cache will populate, both the **Central Ignition Gateway** and **Fallback Gateway** must be connected over the Gateway Network. The credentials are passed from a User Source on the **Central Ignition Gateway** to the Fallback Cache over the Gateway Network.
- A Fallback Cache User Source must exist on the **Fallback Gateway**.
- A client must be launched on the **Fallback Gateway** from the **Central Ignition Gateway**. If the user successfully authenticates against the **Central Ignition Gateway's** User Source, then the credentials are cached into the Fallback Cache.


Plant Floor
Fallback Gateway
Central Ignition Gateway Client
Fallback if Connection Fails
Internet or Wide Area Network
Central Ignition Gateway
IT Server Room
Healthy Connection
In Case of Connection Failure

Related Topics ...

- User Source

Verifying a User Source

You can verify that a user exists in a given User Source, if the password is correct, what roles a user has, and any other information added about the user.

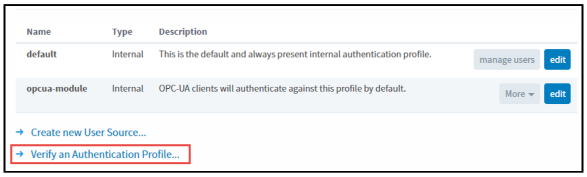
On this page ...

Verifying an Authentication Profile

Watch the Video

Verify a User

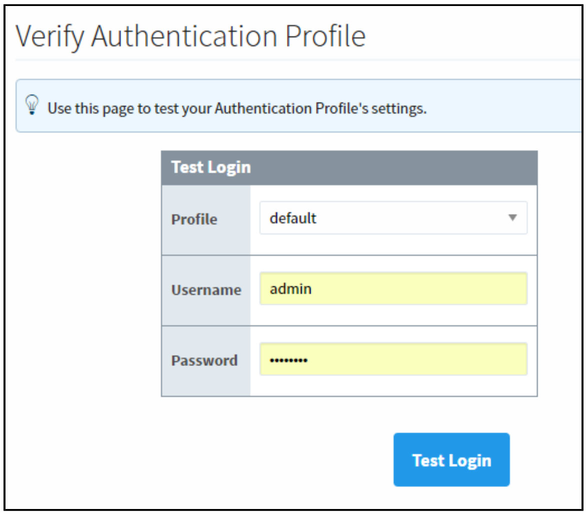
1. On the Gateway Webpage, go to the **Config** tab.
2. Choose **Security > Users, Roles** from the menu on the left.
   The **User Sources** page is displayed.
3. Select the **Verify an Authentication Profile** link.



Name	Type	Description	
default	Internal	This is the default and always present internal authentication profile.	manage users edit
opcua-module	Internal	OPC-UA clients will authenticate against this profile by default.	More edit

→ Create new User Source...
→ Verify an Authentication Profile...

4. The Verify Authentication Profile window will appear. Choose a **Profile** from the dropdown list.
   Enter a **Username** and **Password.** Click **Test Login**.
   Ignition will test the credentials then display the results of the validity test.



Verify Authentication Profile

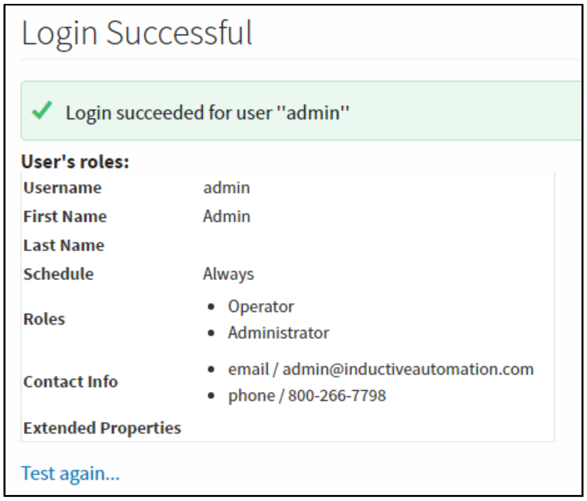
Use this page to test your Authentication Profile's settings.

Test Login

Profile	default
Username	admin
Password	••••••••

If the Login is incorrect, an error message will appear stating that the Login failed for a specified user.
If the Login is correct, a successful message will appear for a specified user along with

their information.



Login Successful

✓ Login succeeded for user "admin"

User's roles:

Username	admin
First Name	Admin
Last Name	
Schedule	Always
Roles	Operator Administrator
Contact Info	email / admin@inductiveautomation.com phone / 800-266-7798
Extended Properties	

Identity Providers

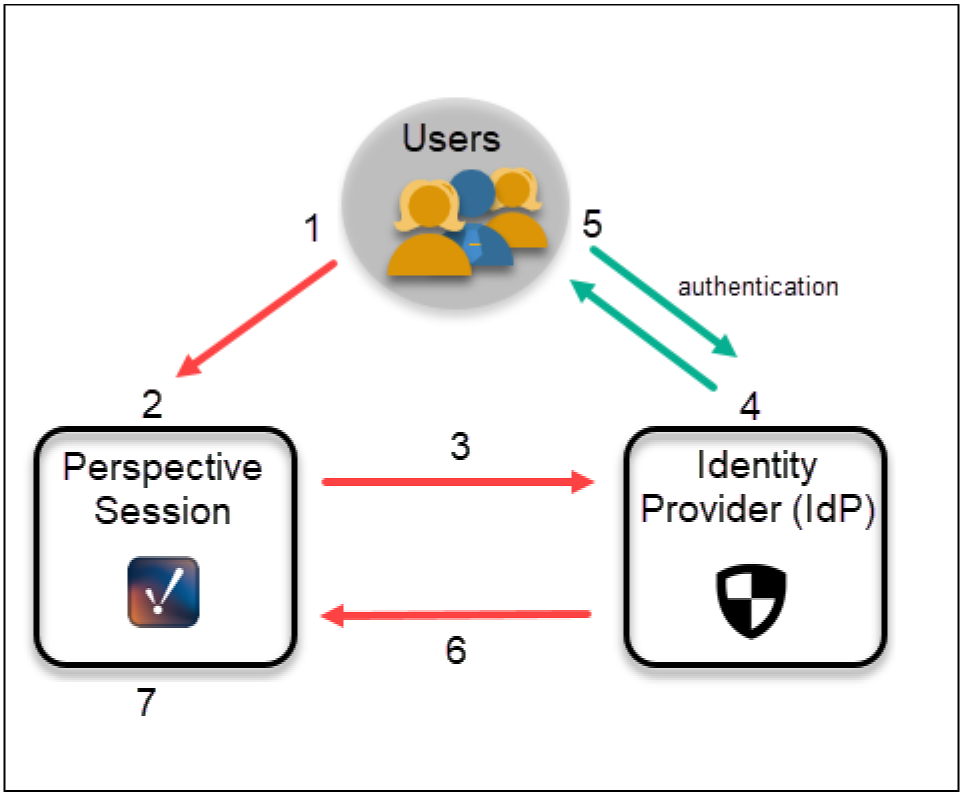
An Identity Provider (IdP) offers a way for users to log in to Ignition using credentials stored outside of Ignition. An IdP creates, maintains, and manages identity (login) information while providing authentication services to Ignition. This provides a secure login that allows Ignition to use SSL and two-factor authentication (2FA).

> ⚠ In 8.0, Identity Providers are only utilized by Perspective. Authentication in other areas of Ignition, such as the Vision module, is handled by User Sources.

On this page ...

Identity Provider Authentication Workflow

The following diagram illustrates how IdP authentication works.



1. **User starts a Perspective Session**.
2. **User attempts some action that requires authentication**.
3. **User is Redirected to Identity Provider**: The Session sees that authentication is required and redirects the user to a webpage hosted by the IdP.
4. **IdP Authenticates the User**: The IdP prompts the user with a security challenge, such as requesting a username and password. The extent of the challenge depends entirely on the provider, but many providers may offer support for multi-factor authentication (MFA).
5. **User Responds**: The user correctly responds to the security challenge.

6. **Redirect back to the Session:** If the IdP successfully validates the user, it will redirect the user back to the Perspective Session. Some IdPs may have an additional workflow they will guide the user through, such as re-verifying an email address or replacing an expired password. The IdP will also return information about that user to the Session. This provides some context about the user that the Session can use to assign Security Levels.
7. **Update the User's Security Level**: Once back at the session, the user will be mapped to the specified Security Level, giving the user access to the restricted action.

Types of Identity Providers

The following types of providers are available. More information on the types can be found on the Identity Provider configuration reference page.

- **Ignition** - The Gateway will act as an Identity Provider, accepting authentication requests from other Perspective Sessions. Users and roles are stored internally to Ignition. Useful when an external identity provider is unavailable.
- **OpenID Connect 1.0** - Used to configure an external IdP via OpenId Connect.
- **Security Assertion Markup Language (SAML)** - Used to configure an external IdP via SAML.

Suggested External Identity Providers

Your organization's IT may have some sort of existing integration with an Identity Provider. Some popular Identity Providers are listed below.

- Ping Identity
- Okta
- Active Directory Federation Services
- Duo

Using Identity Providers

Once an Identity Provider has been configured, there are a few things that can be done to test and adjust how it works. You can map the attributes that are returned in the IdP response document to more familiar user properties that are available to use within the project. You can add rules to custom security levels that determine when a user falls into the level. Overrides can be given to users in the form of User Grants, so that they are granted certain security levels regardless of the rules. Finally, you can test out the IdP by logging in with a user to confirm what is returned in the response document.

In This Section ...

Configuring Identity Providers

Registering the Ignition Gateway

Before configuring an Identity Provider on the Ignition Gateway, it must first be registered as an Identity Provider Client. Your Identity Provider will have a workflow to register, and it will most likely request something called a **return URL** or **redirect URI**. The paths provided utilize your Gateway's address/hostname, and they change depending on the type of provider.

> ⚠ The same redirect URI is used for login and logout.

OpenID Connect Providers (OP):

OpenID Connect Providers

```
http://yourGatewayAddress:Port/data/federate/callback/oidc
```

SAML Providers:

SAML Providers

```
http://yourGatewayAddress:Port/data/federate/callback/saml
```

Secure Integration with IdPs

You should always use the secure SSL versions of those redirect URIs (https) in production environment. To do this you must enable SSL in Ignition and install a valid certificate. This is the best practice for maintaining a secure integration with third party Identity Providers.

On this page ...

INDUCTIVE UNIVERSITY

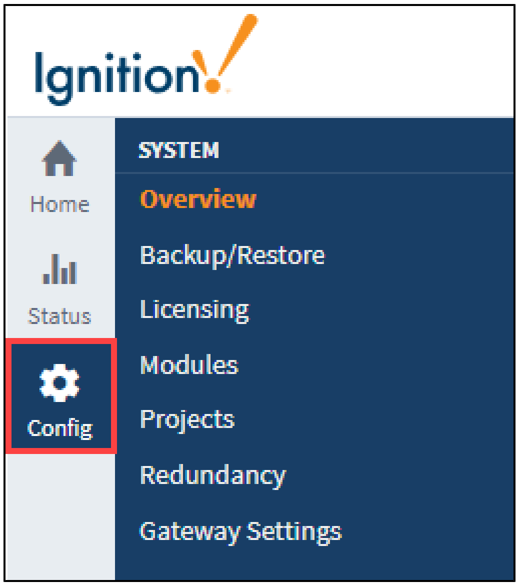
Configuring Identity Providers

Watch the Video

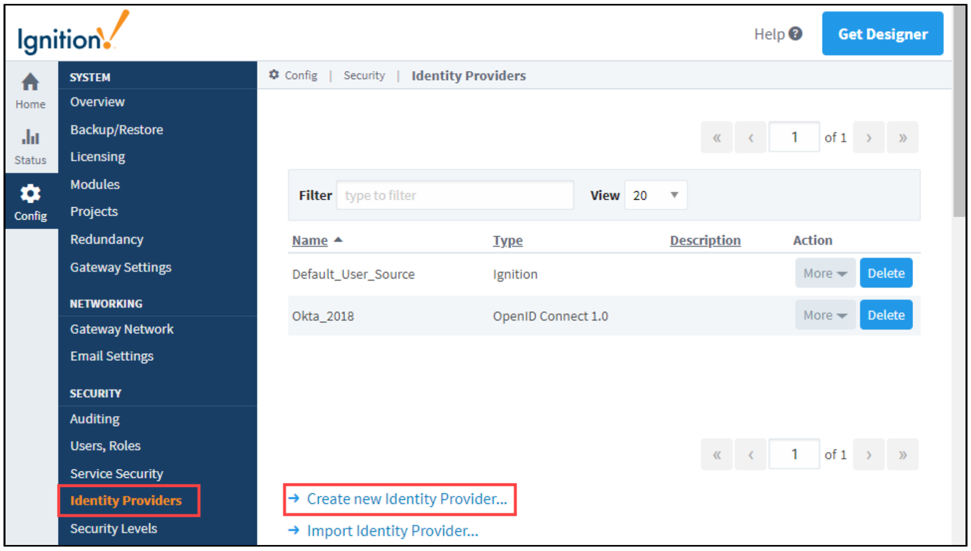
Configuring an Identity Provider

Although there are several types, the general workflow for creating an Identity Provider is the same.

1. On the Gateway Webpage, click on the **Config** tab.

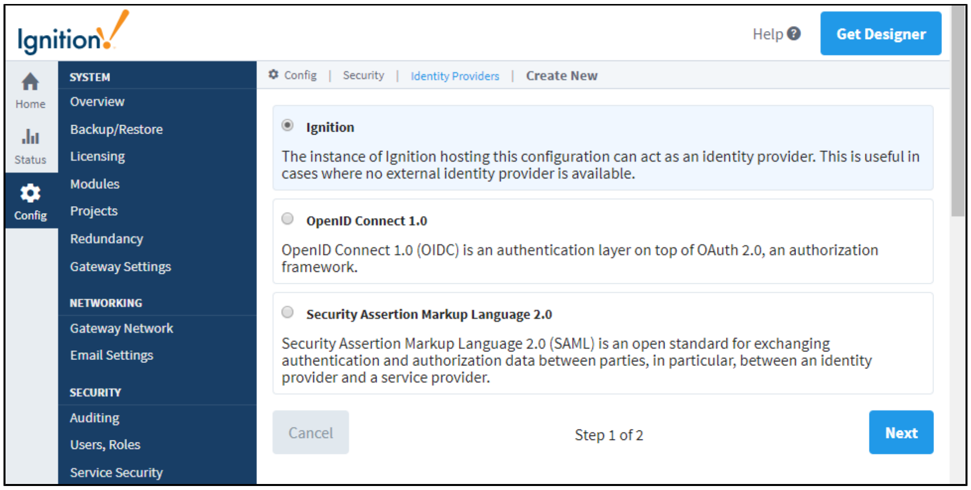


2. Under the Security section, click on **Identity Providers**. The Identity Providers screen is displayed. This screen will list all IdPs that have been configured. You can change filter by name or adjust the number of IdPs displayed in the view.



Name	Type	Description	Action
Default_User_Source	Ignition		More Delete
Okta_2018	OpenID Connect 1.0		More Delete

3. Click on **Create New Identity Provider...**
4. Choose the type of provider. The current options are Ignition, OpenID Connect 1.0, or Security Assertion Markup Language 2.0 (SAML).



● Ignition

The instance of Ignition hosting this configuration can act as an identity provider. This is useful in cases where no external identity provider is available.

○ OpenID Connect 1.0

OpenID Connect 1.0 (OIDC) is an authentication layer on top of OAuth 2.0, an authorization framework.

○ Security Assertion Markup Language 2.0

Security Assertion Markup Language 2.0 (SAML) is an open standard for exchanging authentication and authorization data between parties, in particular, between an identity provider and a service provider.

Step 1 of 2

5. Click the **Next** button.
6. Configure the adapter. This step varies based on the type of provider. Please see the reference tables below for a description of properties.
7. Once you've filled in the properties, click **Save**.

Common Properties

All Identity Provider types share the following properties:

Property Name	Description	Required?
Provider Name	The name of the adapter. Adapter names must be unique, so no two adapters on the same Gateway may have the same name. The following feature is new in Ignition version **8.0.6** Click here to check out the other new features As of release 8.0.6, the naming conventions for IdPs are as follows: • IdP names must begin with an underscore or alpha character. • The remaining characters in the name must be either underscores or alphanumeric. • IdP names are **not** case sensitive.	Yes
Provider Description	A description of the provider.	No
Provider Type	The type of Identity Provider. The value for this field comes from the previous screen. It cannot be changed here.	Yes

Ignition Identity Provider

The Ignition Identity Provider has the following properties:

Property Name	Description	Required?
User Source	The User Source for this IdP. In order to properly authenticate users, the Ignition Identity Provider must be able to query the list of users from the underlying user source profile. Ensure that your Active Directory, Database, and Hybrid user source profiles are configured accordingly to allow for this.	Yes
Authentication Methods	The following feature is new in Ignition version **8.0.5** Click here to check out the other new features	Yes

	You can opt into Badge based authentication for the IdP by enabling the “Badge” Authentication Method. The “Default” radio button determines which option users first see when attempting to authenticate against the IdP.	
Badge Secret	The following feature is new in Ignition version **8.0.5** Click here to check out the other new features Choose whether or not the user is required to enter a secret (password) along with their badge scan.	Yes

Built-In Attributes

The following attributes are available in the Ignition IdP.

Attribute	Type	Description	Example
auth_time	Date	The following feature is new in Ignition version **8.0.14** Click here to check out the other new features Represents the time the user last authenticated.	`//Checkifithasbeenwithin15minutessincethelast` `//authentic`

			`ationattemptdateDiff({idp-attributes:auth_time},now(),"minutes")<=15`
challenged	Boolean	The following feature is new in Ignition version **8.0.14** Click here to check out the other new features	`//Ret`

		Signifies if the user provided credentials at the last login. If true, then the user was asked to re-validate their credentials the last time they attempted to login. If false, then they were not challenged to re-validate their credentials during the last login attempt. This can happen when a login request was made after a user was already authenticated. For example, if a user was already authenticated in a Perspective Session, and a separate call to system.perspective.login function was made with the forceAuth parameter set to false, meaning the user did not provide credentials during the last authentication challenge.	urns True or False, depending on whether or not the user // provided credentials at the la

				`stlogin. {idp-attributes:challenged}`

OpenID Connect Providers

OpenID Connect Providers (OP) properties are listed in the following tables. The values on many of these properties may require you to refer to information from your third-party IdP.

Importing Metadata from the Provider

This method is preferred because of its ease-of-use and accuracy. After importing, you will only need to add your client ID and secret manually. (However you can revise the imported data if needed as well.)

Property Name	Description
Import from URL	URL to the OpenID Provider Configuration document. Typically, if the issuer is "https://example.org/foo" then the metadata URL would be "https://example.org/foo/.well-known/openid-configuration"
Import From File	File must be a JSON document with the properties described in section 3 (OpenID Provider Metadata) of the OpenID Connect Discovery 1.0 specification.

Configuring the Provider

Most OpenID Providers will require registering Ignition as a client. After the registration process is complete, the provider will generate a client ID and secret for Ignition, which is required below. This gives Ignition the ability to communicate securely with the provider. Most providers will also require a set of redirect URIs. An example redirect URI would look like: `http://hostname:port/data/federate/callback/oidc`

Property Name	Description	Required?

Client ID	The client identifier registered within the identity provider. This value is provided the Identity Provider.	Yes
Client Secret	The client secret registered within the identity provider. This value is provided by the Identity Provider.	Yes
Authorization URL	URL of the OP's OAuth 2.0 Authorization Endpoint.	Yes
Token URL	URL of the OP's OAuth 2.0 Token Endpoint.	Yes
Logout URL	Optional URL at the OP to which an RP can perform a redirect to request that the end user be logged out at the OP.	No
JSON Web Keys URL	URL of the OP's JSON Web Key Set document.	No
Use Json Web Keys URI	If checked, then identity provider public keys will be automatically downloaded from given JSON Web Keys URL. New keys will be automatically fetched when the identity provider generates new keys. If unchecked, then the static set of JSON Web Keys (configured below) are used, so when the identity provider rotates keys, they must be manually added to this configuration.	No
User Info URL	The following feature is new in Ignition version **8.0.16** Click here to check out the other new features Optional URL to retrieve UserInfo claims from the provider. Resulting claims are typically determined by the scopes listed under the Scope setting.	No
Issuer	Entity that issues a set of claims.	Yes
Supported ID Token Signing Algorithm Values	A list of the JSON Web Signature (JWS) signing algorithms supported by the OP for the ID Token to encode the claims in a JWT.	Yes
Scope	A list of scopes which will be sent for each auth request to the OP. Commonly used scopes would be `email` and `profile` but check your Identity Provider's documentation for more information.	No
JSON Web Key Config	A list of signing key(s) the RP uses to validate signatures from the OP.	No

JSON Web Key Configuration

Property Name	Description	Required?
Key Type	The cryptographic algorithm family used with the key. Options are EC, RSA or oct.	Yes
Public Key Use	The intended use of the public key. Options are sig or eng.	No
Key Operations	The operation(s) for which the key is intended to be used.	No
Algorithm	The algorithm intended for use with the key.	Yes
Key ID	Used to match a specific key.	No
X.509 URL	A URI that refers to a resource for an X.509 public key certificate or certificate chain. The identified resource MUST provide a representation of the certificate or certificate chain that conforms to RFC 5280 in PEM-encoded form, with each certificate delimited as specified in Section 6.1 of RFC 4945.	No
X.509 Certificate Chain	The "x5c" (X.509 certificate chain) parameter contains a chain of one or more PKIX certificates. Each entry must be a base64-encoded (Section 4 of RFC4648 -- not base64url-encoded) DER PKIX certificate value.	No
X.509	A base64url-encoded SHA-1 thumbprint (a.k.a. digest) of the DER encoding of an X.509 certificate.	No

Certificate SHA-1 Thumbprint		
X.509 Certificate SHA-256 Thumbprint	A base64url-encoded SHA-256 thumbprint (a.k.a. digest) of the DER encoding of an X.509 certificate.	No

There are some additional properties, that depend on which Key Type is selected.

Key Type: EC

Property Name	Description	Required?
crv (Curve)	The cryptographic curve used with the key.	Yes
x (X Coordinate)	The x coordinate for the Elliptic Curve point represented as the base64url encoding of the octet string representation of the coordinate.	Yes
y (Y Coordinate)	The y coordinate for the Elliptic Curve point represented as the base64url encoding of the octet string representation of the coordinate.	No
d (ECC Private Key)	The Elliptic Curve private key value represented as the base64url encoding of the octet string representation of the private key value.	No

Key Type: RSA

Property Name	Description	Required?
n (Modulus)	The modulus value for the RSA public key represented as a Base64urlUInt-encoded value.	Yes
e (Exponent)	The exponent value for the RSA public key represented as a Base64urlUInt-encoded value.	Yes
d (Private Exponent)	The private exponent value for the RSA public key represented as a Base64urlUInt-encoded value.	No
p (First Prime Factor)	The first prime factor represented as a Base64urlUInt-encoded value.	No
q (Second Prime Factor)	The second prime factor represented as a Base64urlUInt-encoded value.	No
dp (First Factor CRT Exponent)	The Chinese Remainder Theorem (CRT) exponent of the first factor represented as a Base64urlUInt-encoded value.	No
dq (Second Factor CRT Exponent)	The CRT exponent of the second factor represented as a Base64urlUInt-encoded value.	No
qi (First CRT Coefficient)	The CRT coefficient of the second factor represented as a Base64urlUInt-encoded value.	No
oth (Other Primes Info)	Information about any third and subsequent primes, should the exist. Each new Prime added will provide users with new Prime Factor, Factor CRT Exponent, and Factor CRT Coefficient properties, all of which are required.	No

Key Type: oct

Property Name	Description	Required?
k (Key Value)	The value of the symmetric (or other single-values) key represented as the base64url encoding of the octet sequence containing the key value.	Yes

Security Assertion Markup Language (SAML) Providers

The properties for Security Assertion Markup Language (SAML) are listed in the following tables. The values on many of these properties may require you to refer to information from your third-party IdP.

Importing Metadata from the Provider

This method is preferred because of its ease-of-use and accuracy. After importing, you will only need to add your client ID and secret manually. (However you can revise the imported data if needed as well.)

Property Name	Description
Import from URL	URL to the SAML Identity Provider Metadata document.
Import From File	File must be an XML document which conforms to the SAML 2.0 metadata schema described in saml-metadata-2.0-os.

The SAML Service Provider (SP) metadata for an Ignition Gateway may be accessed at the following URL: **http://<ipaddress>:<port>/data/saml/metadata/sp**.

The Assertion Consumer Service (ACS) URL for this Ignition Gateway is: **http://<ipaddress>:<port>/data/federate/callback/saml**

Both of these addresses assume you know the IP Address and port of your Ignition install. For example, if you are on the computer Ignition is installed on, you could use: `http://localhost:8088/data/saml/metadata/sp` for the SP metadata.

Configuring the Provider

Property Name	Description	Required?
Entity ID	The Identity Provider's Entity ID.	Yes
Assertion Consumer Service (ACS) Binding	The expected binding used by the Identity Provider when interacting with Ignition's Assertion Consumer Service.	Yes
Name ID Format	The expected name ID format for subjects of assertions resulting from Authn Requests. Options are UNSPECIFIED, EMAIL_ADDRESS, X509_SUBJECT_NAME, WINDOWS_DOMAIN_QUALIFIED_NAME, KERBEROS_PRINCIPAL_NAME, ENTITY_IDENTIFIER, PERSISTENT_IDENTIFIER, TRANSIENT_IDENTIFIER.	Yes
Single Sign-On (SSO) Service URL	The Identity Provider's Single Sign-On (SSO) Service URL.	Yes
Single Sign-On (SSO) Service Binding *	The binding Ignition will use for sending Authn Requests to the Identity Provider's Single Sign-On (SSO) Service.	Yes
Force Authn	Check this box to force complying Identity Providers to authenticate the user each time instead of relying on a previous security context. See section 3.4.1 of saml-core-2.0-os for more details.	Yes
Validate Response Signatures	Check this box to validate the signature of the response from the Identity Provider.	Yes
Validate Assertion Signature	Check this box if it is expected that assertions will be signed. Ignition will validate the signatures of each assertion.	Yes
Signature Verifying Keys	A list of signing key(s) that Ignition uses to validate signatures from the IdP.	Yes
Signature Verifying Certificates	A base64-encoded DER PKIX certificate value.	No

SAML Signature Verifying Key Configuration

Property Name	Description	Required?
Key Algorithm	The algorithm identifier for this signature verifying key. Options are DSA, RSA, or EC.	Yes
Key Value	A base64-encoded DER key value.	Yes

IdP Examples and Troubleshooting

The OpenID Connect 1.0 Example page will show you how to configure an external IdP that used OpenID Connect 1.0 with your Ignition system.
Go to Troubleshooting Identity Providers for helpful examples to help you diagnose and troubleshoot issues with configuring IdPs.
Refer to SAML Example page for how to configure an Identity Provider that is using the SAML protocol.

User Attribute Mapping

The User Attribute Mapping page allows you to map information in the Identity Providers (IdP) response document to easily understandable properties. These properties are then made available as Session Properties in the Perspective Session. To work, this requires that you are already connected to an IdP and are getting fields back when a user logs in.

User Attributes

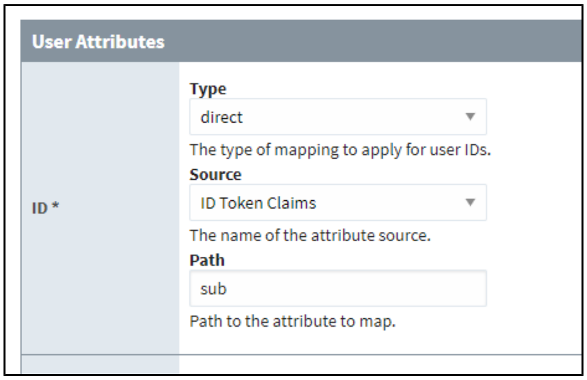
Direct

Direct mappings require that you enter in the path to an attribute in the response document for the given property. Values in the Identity Provider’s response document are dereferenced using the configured paths and are copied to the respective user properties in Ignition.

The following feature is new in Ignition version **8.0.16**
Click here to check out the other new features

Ignition 8.0.16 introduces the Source dropdown, which allows a particular mapping to target a specific object in the response document. The Path property then determines what object inside of source should be.

In the image below, the ID of the user, as represented on the Ignition Gateway, will be determined by the **sub** attribute, located in the **ID Token Claims** object in the response.


User Attributes

ID *

Type: direct
The type of mapping to apply for user IDs.
Source: ID Token Claims
The name of the attribute source.
Path: sub
Path to the attribute to map.

Each Identity Provider type has a different list of possible source items:

SAML

- Authentication Response

Ignition IdP

- ID Token Claims (default)
- Token Endpoint Response

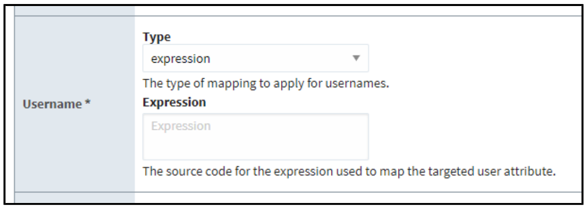
OIDC

- ID Token Claims (default)
- Token Endpoint Response
- User Info Claims

⚠ The user info claims attribute source is present only when the user info URL is configured and Ignition receives a valid user info response from the IdP.

Expression

The expression type allows you to use the expression language to derive the attribute from contextual data such as the IdP response document or Tags.



Username *

Type

expression

The type of mapping to apply for usernames.

Expression

Expression

The source code for the expression used to map the targeted user attribute.

Generally this type of mapping is used in conjunction with the built-in `attribute-source` object, which yields the full response. From there the rest of the expression language can be used to parse the document, looking for specific values.

For example, if the response looked like the following:

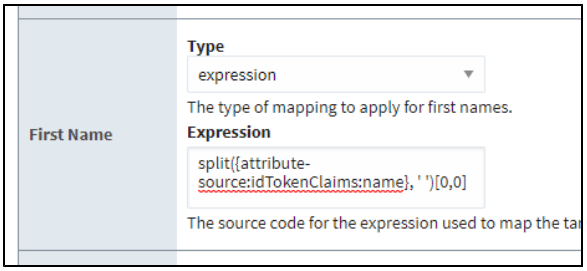
```
 "idTokenClaims": {
    "name": "Paul McCoolGuy"
}
```

Then a mapping could be configured with the following expression:

```
{attribute-source:idTokenClaims:name}
```

Further more, you could use other functions and syntax in the expression language to further manipulate the results, such as returning just the first name:

```
split({attribute-source:idTokenClaims:name}, ' ')[0,0]
```



First Name

Type

expression

The type of mapping to apply for first names.

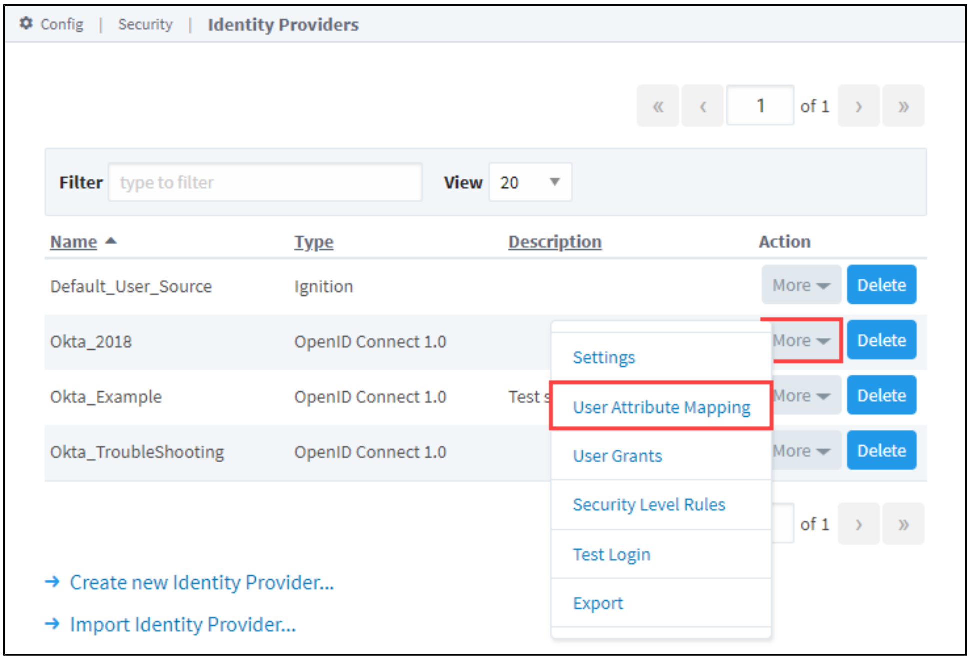
Expression

split({attribute-source:idTokenClaims:name}, ' ')[0,0]

The source code for the expression used to map the tar

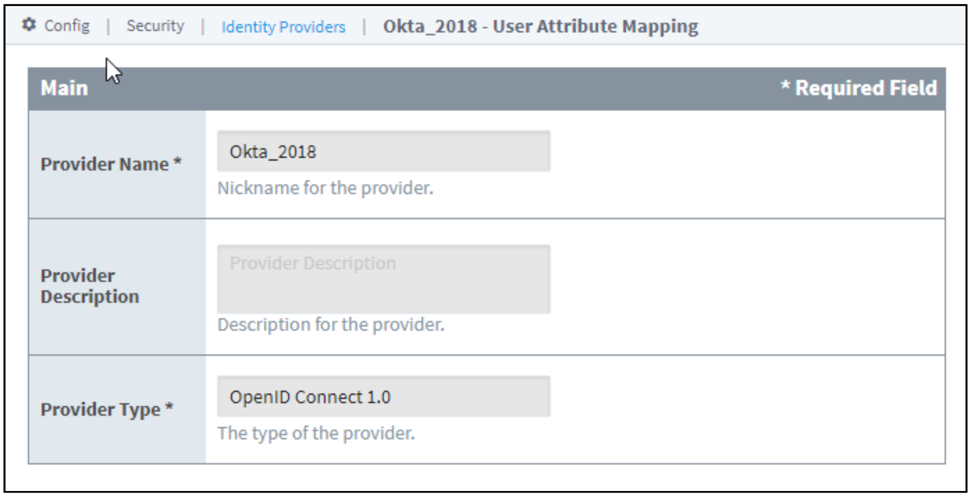
Configuring a User Attribute Mapping

1. From the Gateway Webpage **Config** tab, click on **Security > Identity Providers**.
2. Select the **Identity Provider** and click on **User Attribute Mapping** under the **More** button.



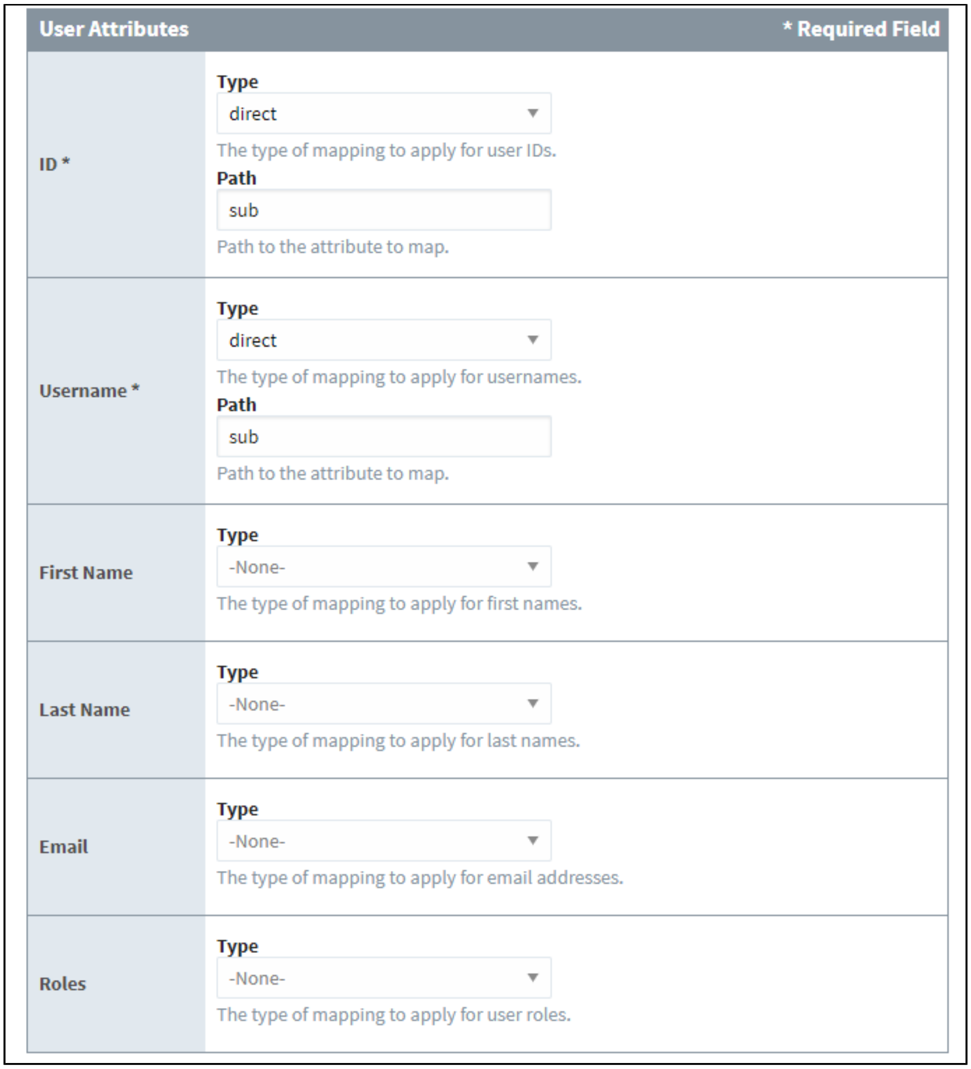
Name ▲	Type	Description	Action
Default_User_Source	Ignition		More Delete
Okta_2018	OpenID Connect 1.0		More Delete
Okta_Example	OpenID Connect 1.0	Test s	More Delete
Okta_TroubleShooting	OpenID Connect 1.0		More Delete

3. The Name, Description, and Provider Type are not editable here, but are listed on the page to make clear which IdP the User Attribute Mapping is being configured on.


Config | Security | Identity Providers | Okta_2018 - User Attribute Mapping

Main	* Required Field
Provider Name *	Okta_2018 Nickname for the provider.
Provider Description	Provider Description Description for the provider.
Provider Type *	OpenID Connect 1.0 The type of the provider.

4. Change the Type for each attribute to edit additional fields.



User Attributes	* Required Field
ID *	Type direct The type of mapping to apply for user IDs. Path sub Path to the attribute to map.
Username *	Type direct The type of mapping to apply for usernames. Path sub Path to the attribute to map.
First Name	Type -None- The type of mapping to apply for first names.
Last Name	Type -None- The type of mapping to apply for last names.
Email	Type -None- The type of mapping to apply for email addresses.
Roles	Type -None- The type of mapping to apply for user roles.

5. When you're finished mapping user attributes, click **Save**.

User Attribute Mapping Property Reference Table

Property Name	Description
ID	The unique ID of the user.
Username	The username of the user.
First Name	The user's first name.
Last Name	The user's last name.
Email	The user's email.
Roles	The user's roles.

Security Level Rules

When a user accesses a Perspective project, they can fall into one or more of many Security Levels that you can set up. The Security Level Rules determine if the user has that Security Level or not. The rules come in the form of expressions that can access any of the Expression Language's functions, Tag values, Security Zone information, or Identity Provider attributes. Security Level Rules are accessible from the Gateway Webpage **Config** tab in **Security > Identity Providers**.

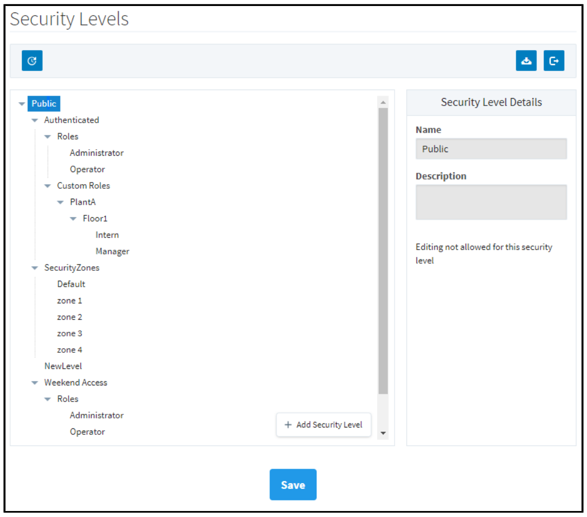
On this page ...

Security Level Rules

Watch the Video

Predefined Rulesets

One thing you may notice right away is that on the Security Level Rules page, many of the built in Security Levels are missing. Some Security Levels don't allow you to create an expression that defines their rules. These particular Security Levels already have a set of rules that govern how a user gets them. The Security Levels corresponding to the different Security Zones are automatically given to users depending on which zones they fall into. Similarly, the Security Levels that correspond to the Roles a user gets while authenticated are automatically given to users depending on what roles we receive from the Identity Provider. These Security Levels are removed from the tree here because rules can't be defined on them. The "Public" and "Authenticated" Security Levels also can't have Security Level Rules defined on them, but are present in the tree because they can potentially be a parent to custom nodes which can have Security Level Rules. The Public level is granted to every user when they open the project, and the Authenticated level is granted when the user authenticates against an Identity Provider, regardless of what roles they may have.



Security Levels

- Public
  - Authenticated
    - Roles
      - Administrator
      - Operator
    - Custom Roles
      - PlantA
        - Floor1
          - Intern
          - Manager
  - SecurityZones
    - Default
    - zone 1
    - zone 2
    - zone 3
    - zone 4
  - NewLevel
  - Weekend Access
    - Roles
      - Administrator
      - Operator

Security Level Details

Name: Public

Description:

Editing not allowed for this security level

Defining Security Level Rules

For the Security Levels that can have rules defined, the rules are defined in the form of an expression which should return either True or False, the results of which determine whether a user falls into that level or not. The rules can take advantage of everything the expression language has to offer, including the built in expression functions and any Tag values. The expressions here also have the unique ability to access attributes from the Identity Providers response document for the authenticated user, as well as what Security Zone the user falls into.

Special Object Reference

These special objects can be used to reference information gathered from the IdP response document or the Security Zone that the user falls under.

Object Name	Description
	This object gives the collection of security zones that the user currently has. The collection can be handled using one of the

`{security-zones}`

unique functions: `containsAll` or `containsAny`.

`{idp-attributes:X}`

> This feature has been deprecated.
>
> This object was deprecated in 8.0.16, but maintained for backwards compatibility. The `{attribute-source:X:Y}` object should be used instead.

This object gives you access to specific information within the Identity Provider's response document, where "X" is the path to the attribute in the document. This path can vary depending on what IdP the user has authenticated through. Can be used in the unique functions `containsAll` or `containsAny` if the information returned is a collection object, such as a role list.

Attribute	Path	Example
ID	sub	`{idp-attributes:sub}`
Username	preferred_username	`{idp-attributes:preferred_username}`
First Name	given_name	`{idp-attributes:given_name}`
Last Name	family_name	`{idp-attributes:family_name}`
Email	email	`{idp-attributes:email}`
Roles	roles	`{idp-attributes:roles}`

`{attribute-source:X:Y}`

> The following feature is new in Ignition version **8.0.16**
> Click here to check out the other new features

As of 8.0.16, you can use this object to access values in deeper structures by using colons to delimit each object.

Assuming the following JSON response from an identity provider:

```
{
    "idTokenClaims":
    {
        "foo" : "bar"
    },
    "userInfo":
    {
        "email" : "person@place.com"
    }
}
```

`{attribute-source:userInfo:email}` would retrieve the value of the user's email.

Attribute	Path	Example
ID	sub	`{attribute-source:idTokenClaims:sub}`

Username	preferred_username	`{attribute-source:idTokenClaims:preferred_username}`
First Name	given_name	`{attribute-source:idTokenClaims:given_name}`
Last Name	family_name	`{attribute-source:idTokenClaims:family_name}`
Email	email	`{attribute-source:idTokenClaims:email}`
Roles	roles	`{attribute-source:idTokenClaims:roles}`

The `{attribute-source:X:Y}` object is only available to Security Levels that fall within the Authenticated Security Level. See below under Special Considerations for Rules.

Special Function Reference

When writing an expression to determine Security Level Rules, there are functions available that are not a part of the normal set available to Expression Bindings. These additional functions are:

Function Name	Description	Example

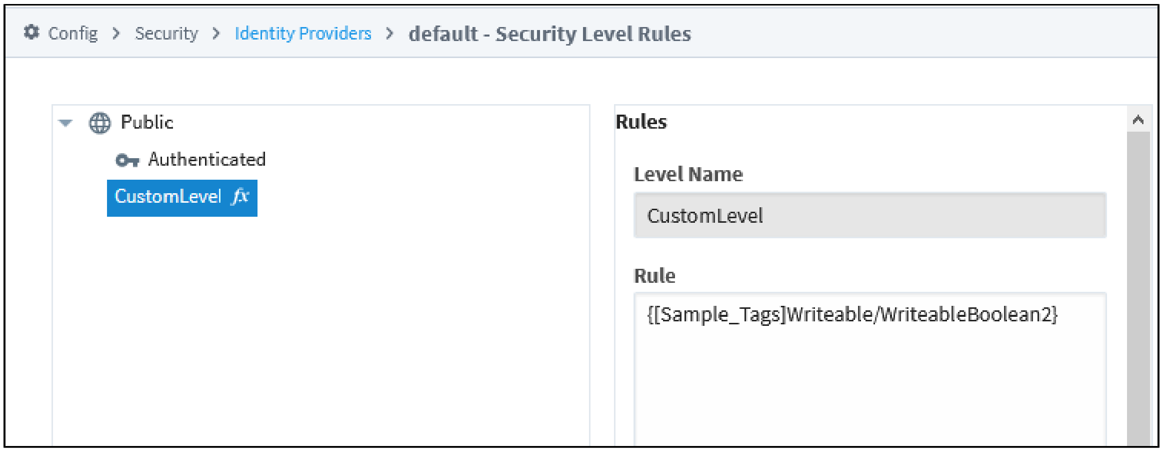
containsAll(collection, element 0, ..., element N)	Checks to see if all of the listed elements are present in the collection object. The function requires at least two arguments, a `collection` and an `element`.	`containsAll(`
		`{security-zones},`
		`'PlantA',`
		`'Floor1',`
		`'Press Room'`
		`)`

containsAny (collection, element 0, ..., element N)	Checks to see if any of the listed elements are present in the collection object. The function requires at least two arguments, a `collection` and an `element`.	`containsAny(`
		`{attribute-source:idTokenClaims:roles},`
		`'Manager',`
		`'Operator'`
		`)`

Evaluating Tag Values

Tag values can be accessed in the Security Level Rules expression area by encasing the Tag Path (including the Tag Provider) in braces ({})

```
{[tagProvider]path/to/tag}
```



Config > Security > Identity Providers > default - Security Level Rules
Public
Authenticated
CustomLevel *fx*
Rules

Level Name: CustomLevel

Rule: {[Sample_Tags]Writeable/WriteableBoolean2}

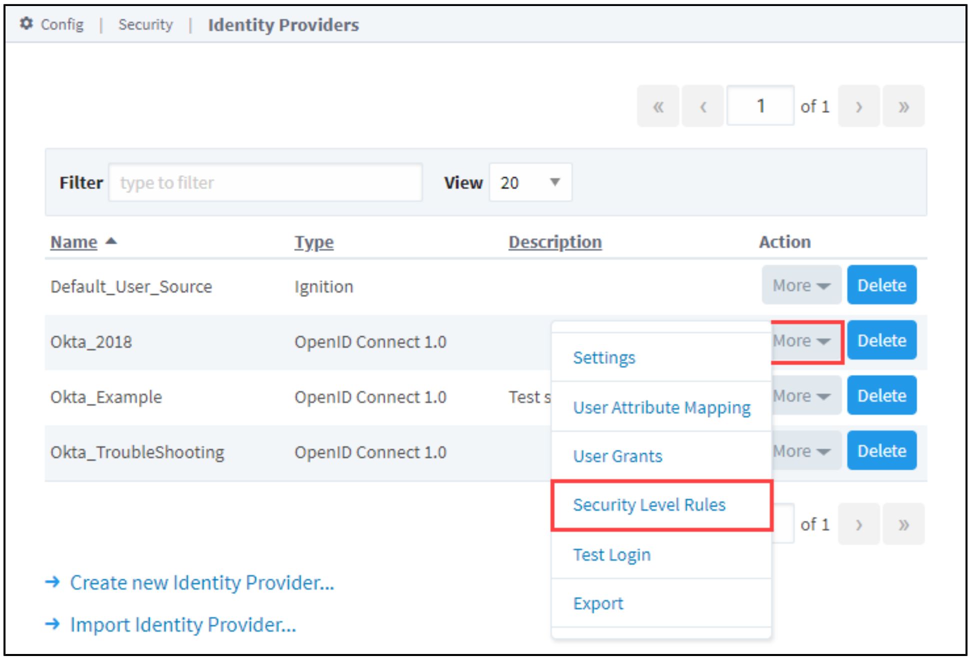
Note: Security Levels are determined on initial login for each session, so if a Security Level is using an expression that references a Tag value, changing the value while the session is running won't change the Security Levels applied to the users already logged in.

Special Considerations for Rules

When defining rules for a Security Level, it is important to note where in the Security Level tree you are. If you want to access information out of the Identity Provider such as the username, you will need to ensure that the Security Level is located in the Authenticated branch. User information is only captured once a user logs in, so that information will only fall under Security Levels that come from the user being Authenticated. If a Security Level lies outside of the Authenticated branch, then the level will only have access to information such as Tag values and Security Zones.

Configuring Security Level Rules

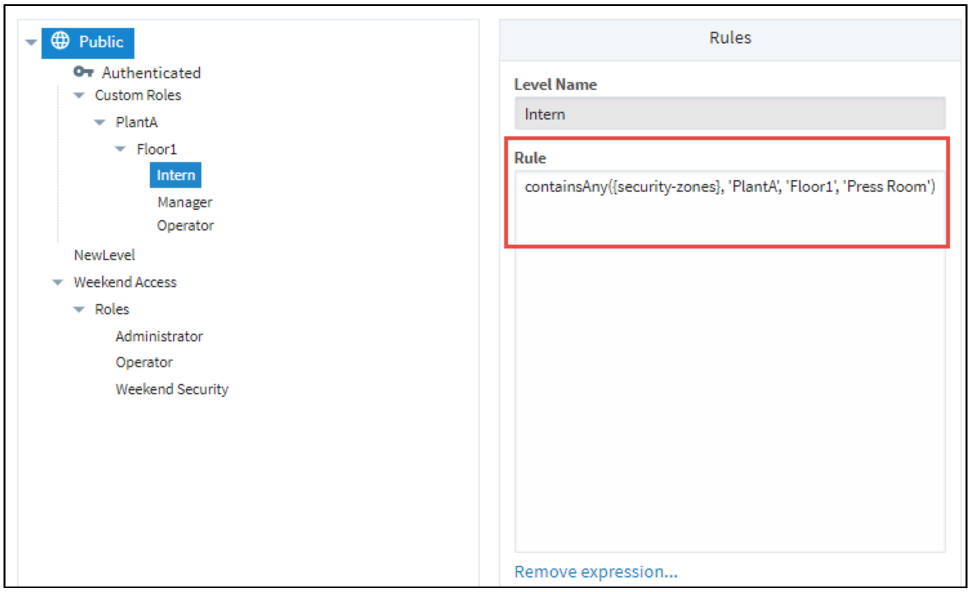
1. From the Gateway Webpage under the **Config** tab, go to **Security > Identity Providers**.
2. A list of the Identity Providers will be displayed. Click the **More** button for the Identity Provider you wand to edit, and select **Security Level Rules**.



Config | Security | Identity Providers
« ‹ 1 of 1 › »
Filter type to filter View 20
Settings
User Attribute Mapping
User Grants
Security Level Rules
Test Login
Export
of 1 › »
Create new Identity Provider...
Import Identity Provider...

Name	Type	Description	Action
Default_User_Source	Ignition		More Delete
Okta_2018	OpenID Connect 1.0		More Delete
Okta_Example	OpenID Connect 1.0	Test s	More Delete
Okta_TroubleShooting	OpenID Connect 1.0		More Delete

3. Select the **Security Level Name** and, if a rule is defined, it will appear in the Rule field. If not, you can create one. We copied the expression **"containsAny ({security-zones}, 'PlantA', 'Floor1', 'Press Room')"** from the example above for the following example.



Level Name

Intern

Rule

```
containsAny({security-zones}, 'PlantA', 'Floor1', 'Press Room')
```

4. After your enter your rule, click **Save**.

User Grants

A User Grant is a way to directly assign a user to a Security Level, even if they do not meet the requirements of the Security Level Rules. User Grants essentially act as an override to the original rules of the Security Level.

User Grants are accessed from the Gateway Webpage **Config** section in **Security > Identity Providers**. Users can be added and edited using the buttons in the Users table so that Security Levels can then be granted to them.

> ⓘ When adding, editing, and deleting users in User Grants, you are only modifying the User Grant (whether the user is granted permission that overrides the Security Rules). The user is not changed in the Identity Provider.

Users are identified by either their username or their ID from the provider response document. Once you have identified a user, you can assign them any number of grants to Security Levels. Selecting a level will automatically select all security levels above it. The User Grants can only be applied to a user after they authenticate with the Identity Provider, though the grants do not have to be for levels within the Authenticated branch.

> ⚠ The system can't validate any user created here against actual users in the Identity Provider (IdP). Instead, the username or ID needs to be entered exactly, and when a user logs in, the system will check to see if they match any of the configured usernames /IDs to give User Grants to.

On this page

...

INDUCTIVE UNIVERSITY

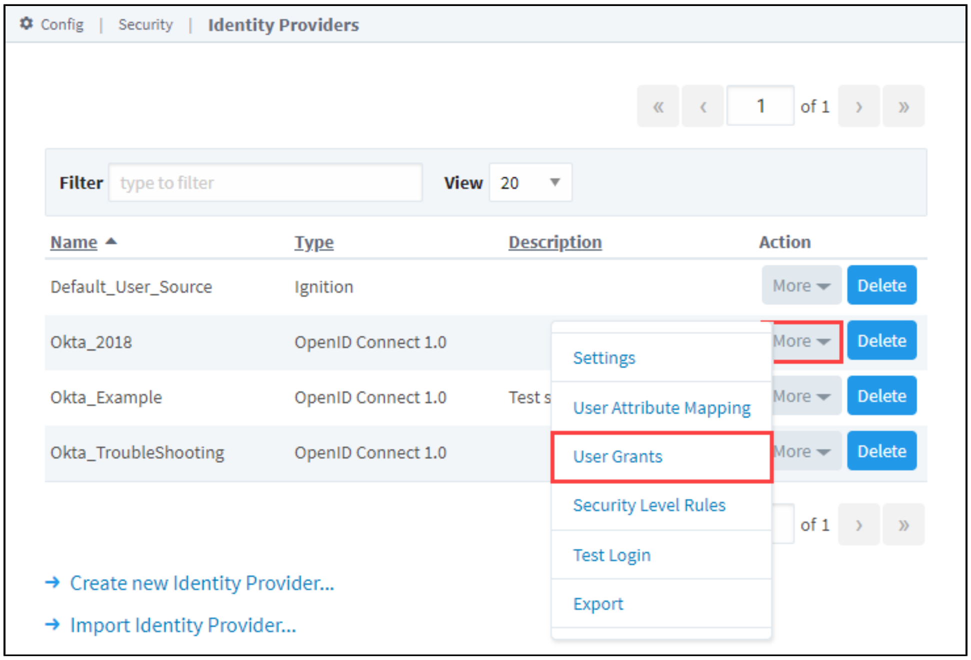
User Grants

Watch the Video

Configuring a User Grant

There are two parts to configuring a User Grant: Adding a user then applying User Grants.

1. From the Gateway Webpage **Config** tab, click on **Security > Identity Providers**. The screen will refresh and you will see a list of all your IdPs.
2. Choose the IdP and click the **More** button to see the actions in the dropdown list.
3. Select **User Grants**.



Config | Security | Identity Providers

« ‹ 1 of 1 › »

Filter type to filter View 20

Name ▲	Type	Description	Action
Default_User_Source	Ignition		More Delete
Okta_2018	OpenID Connect 1.0		More Delete
Okta_Example	OpenID Connect 1.0	Test s	More Delete
Okta_TroubleShooting	OpenID Connect 1.0		More Delete

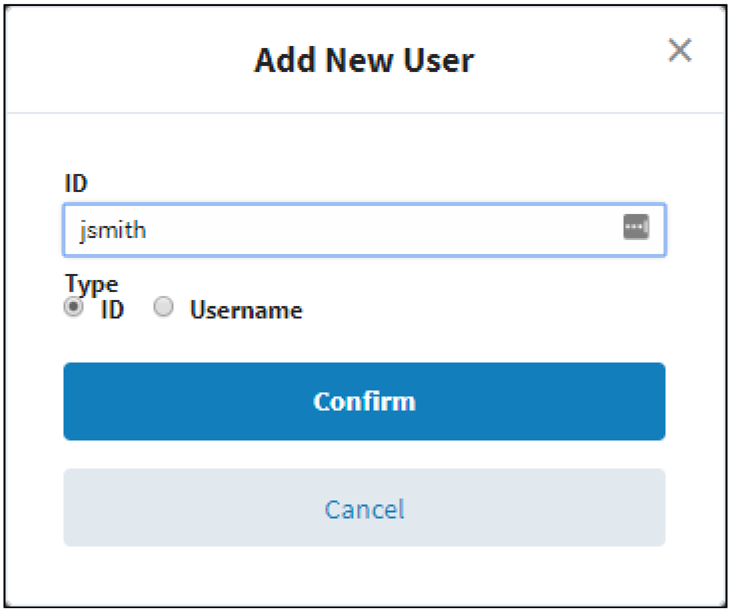
of 1 › »

→ Create new Identity Provider...

→ Import Identity Provider...

4. To add a new user, click the **Add** ✚ icon.

5. Choose how you will identify the user; either with a username or an ID. Click **Confirm** to save the changes.

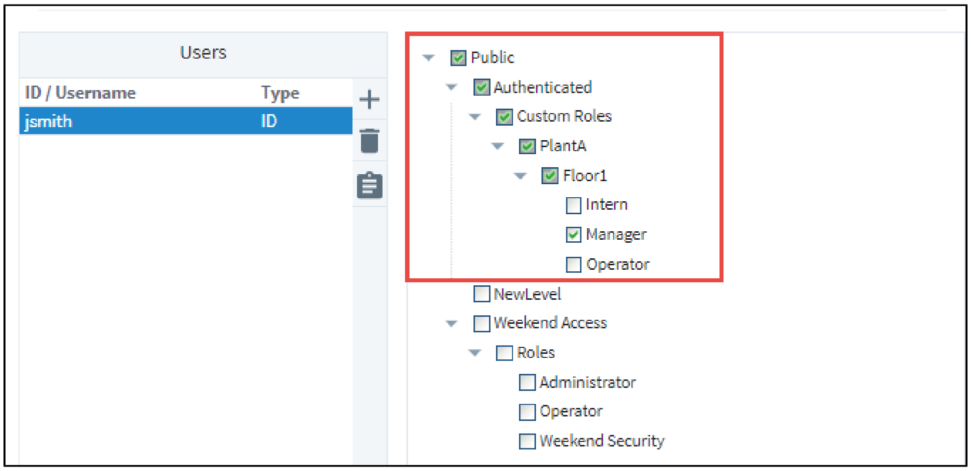


Add New User

ID: jsmith

Type: ● ID ○ Username

6. With the user created and highlighted in the Users table, select Security Levels to grant them when they Authenticate with this Identity Provider.



Users

ID / Username	Type
jsmith	ID

- ☑ Public
  - ☑ Authenticated
    - ☑ Custom Roles
      - ☑ PlantA
        - ☑ Floor1
          - □ Intern
          - ☑ Manager
          - □ Operator
  - □ NewLevel
  - □ Weekend Access
    - □ Roles
      - □ Administrator
      - □ Operator
      - □ Weekend Security

7. Click **Save**.

Now you can test this user through the Test Login and Logout screen to verify the new roles have been assigned.

Test Login and Logout

On this page ...

On the Identity Providers screen you can test a username and password combination against an Identity Provider (IdP).

When you select the Test Login option for your IdP, it will confirm the IdP name and Type that you are testing against. It gives you a way to test your attribute mapping configuration and your security level rules / direct user grants configuration

Clicking the Test Login button will redirect you to the IdP where you can login. Upon successful authentication with the IdP, the page navigates back to Ignition, and Ignition displays the response document as the results. These results can vary between IdPs, so it can be useful to test out a login to see what your IdP returns in its response document.

You can use Test Logout option to log out of the ID you were testing.

The following feature is new in Ignition version **8.0.5**
Click here to check out the other new features

As of 8.0.5, Ignition's IdP returns an 'amr' attribute that indicates how the user was authenticated.

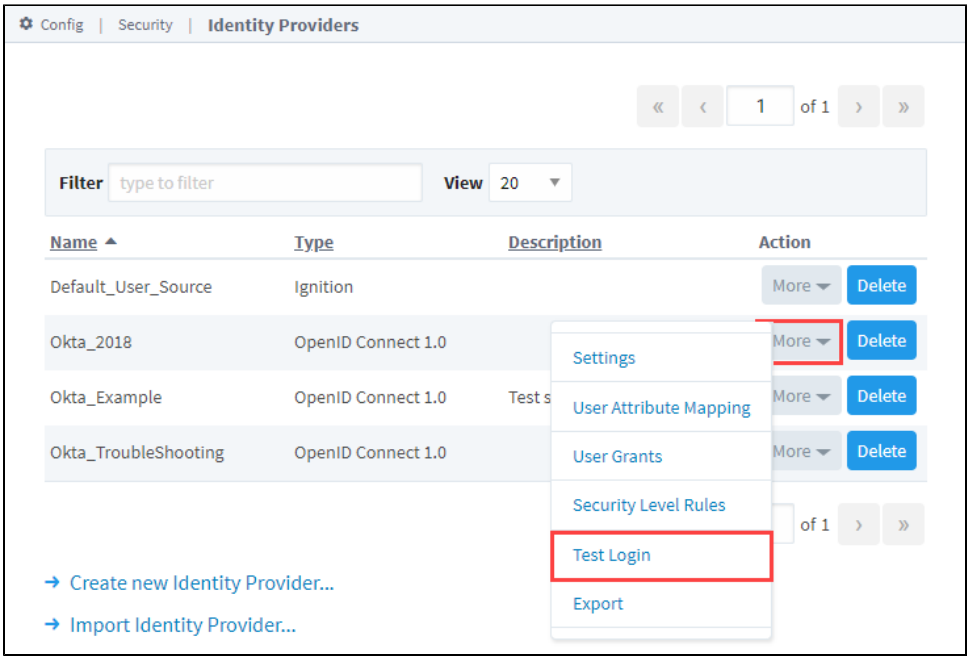
- If the user was authenticated with a username and password challenge, the amr returns ["uname", "pwd"]
- If the user was authenticated with a badge challenge, the amr returns: ["badge"]
- If the user was authenticated with a badge and password challenge, the amr returns:[: badge", "pwd"]

With this feature, you can enable different security levels based on how the user authenticated. In a security level rule, you could enter:

```
containsAll ({idp-attributes:amr}, 'uname', 'pwd')
```

Testing a Login

1. From the Gateway Webpage **Config** tab, go to **Security > Identity Providers**. The window will refresh and your list of Identity Providers will be displayed.
2. Choose the Identity Provider and click the **More** button to see the actions in the dropdown list, and select **Test Login**.

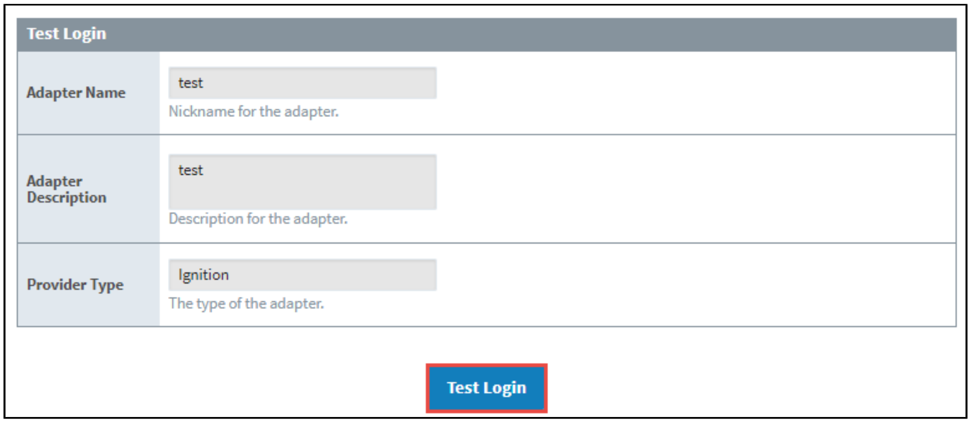


Config | Security | Identity Providers

Filter: type to filter   View: 20

Name ▲	Type	Description	Action
Default_User_Source	Ignition		More Delete
Okta_2018	OpenID Connect 1.0		More Delete
Okta_Example	OpenID Connect 1.0	Test s	More Delete
Okta_TroubleShooting	OpenID Connect 1.0		More Delete

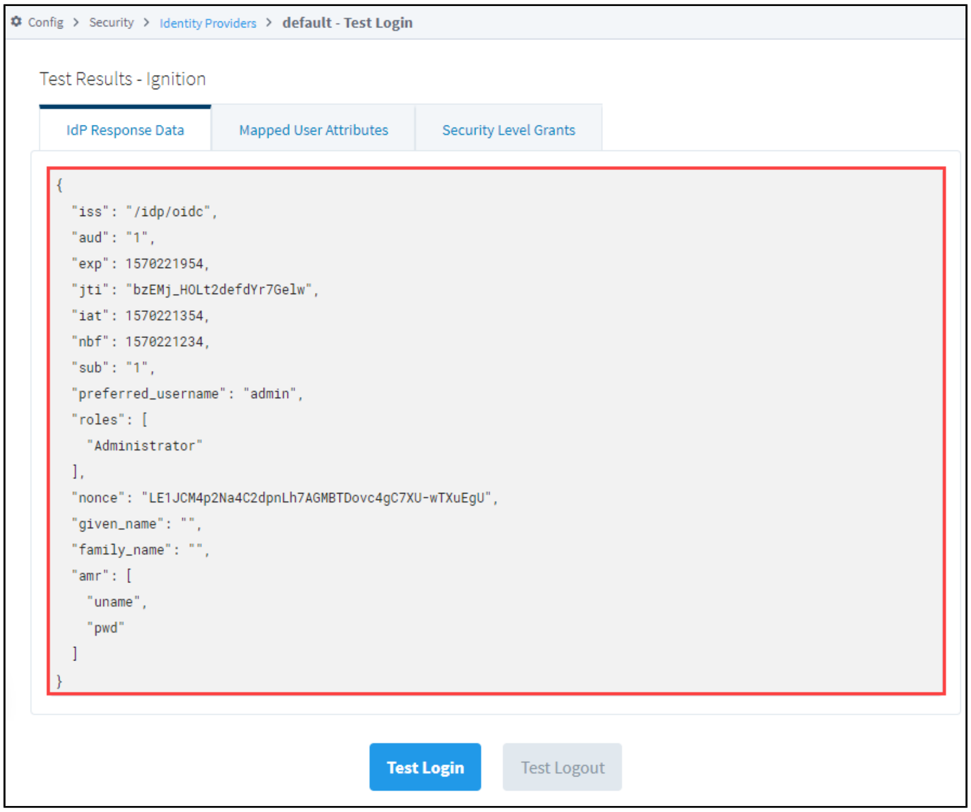
3. Confirm the IdP, and click the Test Login button at the bottom of the screen.



Test Login

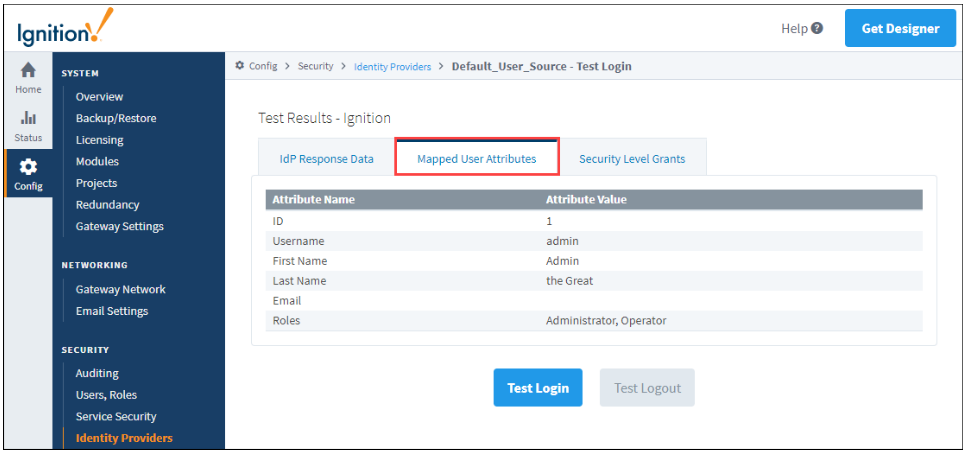
Adapter Name	test Nickname for the adapter.
Adapter Description	test Description for the adapter.
Provider Type	Ignition The type of the adapter.

4. Login at your IdP's login screen.
5. If the login is successful, you will be returned to the Identity Provider Test Login screen. The returned results will be displayed under the IdP Response Data tab.



```
{
  "iss": "/idp/oidc",
  "aud": "1",
  "exp": 1570221954,
  "jti": "bzEMj_HOLt2defdYr7Gelw",
  "iat": 1570221354,
  "nbf": 1570221234,
  "sub": "1",
  "preferred_username": "admin",
  "roles": [
    "Administrator"
  ],
  "nonce": "LE1JCM4p2Na4C2dpnLh7AGMBTDovc4gC7XU-wTXuEgU",
  "given_name": "",
  "family_name": "",
  "amr": [
    "uname",
    "pwd"
  ]
}
```

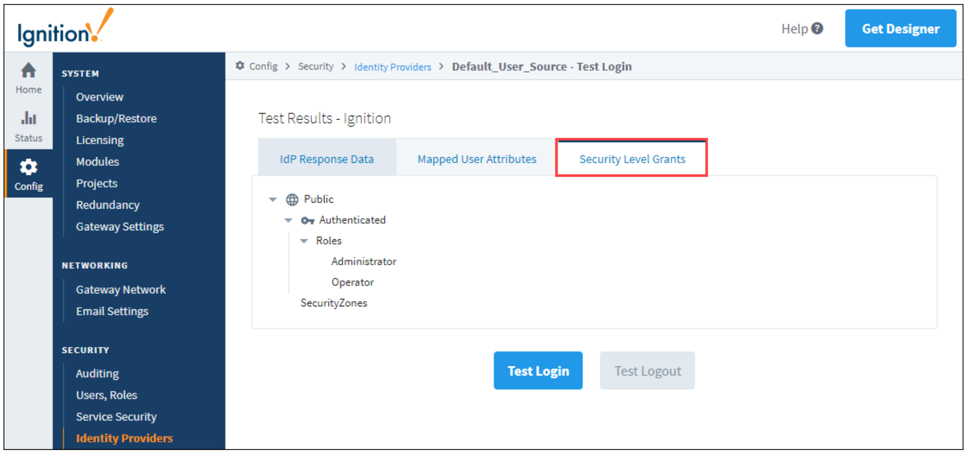
6. Click on the Mapped User Attributes tab to view the user attributes for the currently logged in user.



Test Results - Ignition

Attribute Name	Attribute Value
ID	1
Username	admin
First Name	Admin
Last Name	the Great
Email	
Roles	Administrator, Operator

7. Click on the Security Level Grants tab to view the Security Levels for the roles of the currently logged in user.



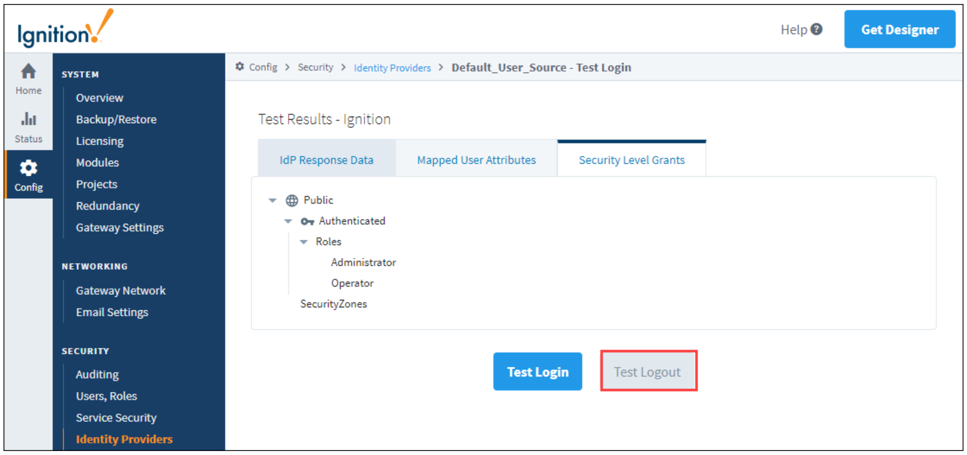
Test Results - Ignition

- Public
  - Authenticated
    - Roles
      - Administrator
      - Operator
  - SecurityZones

Testing Logout

After testing a User ID, you do not want to stay logged in as the user. You can use the Test Logout function to log out. For the Ignition IdP, this function also logs you out of the IdP. For an OpenID Connect IdP, this function will also log you out of the IdP if you have a Logout URL.

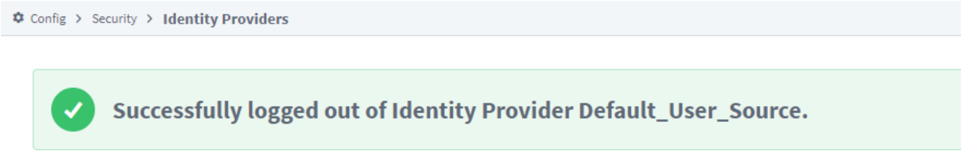
1. To log out of the ID you were testing, click the Test Logout button on the Test Login page.



Test Results - Ignition

- Public
  - Authenticated
    - Roles
      - Administrator
      - Operator
  - SecurityZones

2. You will get a confirmation message of a successful logout.



Successfully logged out of Identity Provider Default_User_Source.

Security Levels

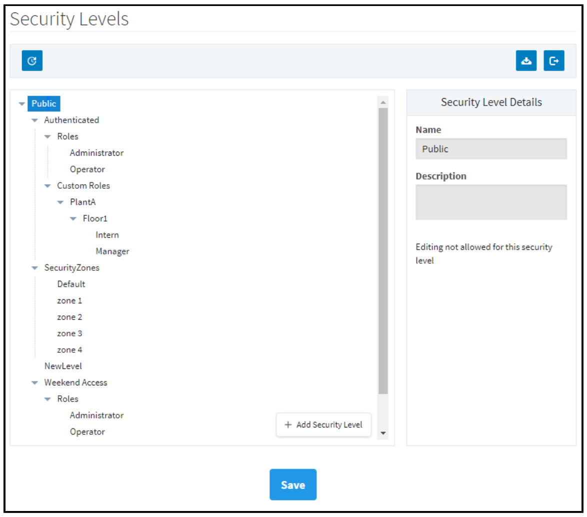
With Security Levels, you define a hierarchy for access inside a Perspective Session. This authorization system provides a way for you to map roles from an Identity Provider (IdP) to Ignition roles. Any IdP can be used to provide roles, and security levels are independent of the type of IdP being used. Any role from the IdP is automatically granted to the user as a role, but only roles in your Security Levels are available to the security screens in the Designer. You can also use the User Grants option to grant additional access for each user.

Security Levels are defined at the Gateway and they are arranged in a tree structure. Each child (nested) level of the tree inherits the security of its parent levels.

There are four reserved Security Levels in the platform:

- Public
- Authenticated
- Authenticated/Roles
- Security Zones.

To access Security Levels, go to the Gateway Webpage under the **Config** tab, and choose **Security > Security Levels**.



Security Levels

- Public
  - Authenticated
    - Roles
      - Administrator
      - Operator
    - Custom Roles
      - PlantA
        - Floor1
          - Intern
          - Manager
  - SecurityZones
    - Default
    - zone 1
    - zone 2
    - zone 3
    - zone 4
  - NewLevel
  - Weekend Access
    - Roles
      - Administrator
      - Operator

Security Level Details

Name: Public

Description:

Editing not allowed for this security level

On this page ...

INDUCTIVE UNIVERSITY

Security Levels

Watch the Video

Reserved Security Levels

The reserved security levels are mostly created for you, and have special rules that determine when a user is granted that level. They can't be renamed or deleted.

Public

All users are always granted the Public security level, even if they are not authenticated (logged in). Public security level indicates open access and the least amount of security. A session that only has the Public security level is not authenticated. This is similar to being a guest or anonymous. Unless another security level is required, guest access will be allowed. The Public security level is the ancestor of all other security levels in the hierarchy.

Authenticated

The Authenticated Security Level is a child of the Public Security Level. If a session has authenticated against the configured IdP successfully, the Authenticated Security Level is granted. Users are required to be logged in in order to have access to this level.

Authenticated/Roles

The Roles level is a special level which itself has no special rules, but it acts as a parent placeholder for potential roles returned from the IdP. This particular level is not configurable; however there can be levels added underneath the Roles level as children. These levels should correspond to the names of roles that would be expected from the IdP. If the IdP provides role information, these roles are automatically mapped to the child security levels underneath Authenticated/Roles. The names of the roles must match exactly for them to be correctly mapped to. For example, if you authenticate against the Ignition IdP configured to delegate to the Internal user source, and your user was granted the roles “A”, and “B”, you would have (at a minimum) the following security levels granted to you:

- Public
    - Authenticated
        - Roles
            - A
            - B

> You can only add one level of children to the Roles Security Level. Custom Roles can be nested as deeply as you want.

SecurityZones

The SecurityZones level is another special placeholder level that itself has no rules but is a parent for all of the Security Zones on the Gateway. Security Zones are automatically pulled in from the Gateway. A Security Zone is a list of Gateways, Computers, or IP addresses that are defined and grouped together. This group is a zone on the Gateway Network, which can have additional policies and restrictions placed on it. Security Zones provide a way to bridge the IdP method of permissions with location-based permission modeling. You cannot add, edit, or remove the SecurityZones node or any node in the SecurityZones sub-tree.

Custom Security Levels

Custom Security Levels can be added to almost anywhere within the tree. When these levels are granted to a user is determined by the Security Level Rules, which can pull information from the IdP, Security Zones, and even Tags. The placement of custom Security Levels can affect when they may be potentially granted to a user. Any custom levels set under the Public level, but not within Authenticated, do not need to have a user authenticate against the IdP to be granted to a user. However, custom levels within Authenticated do need to have the user authenticate to be granted to the user, even if the rule for that level does not use any of the IdP attributes.

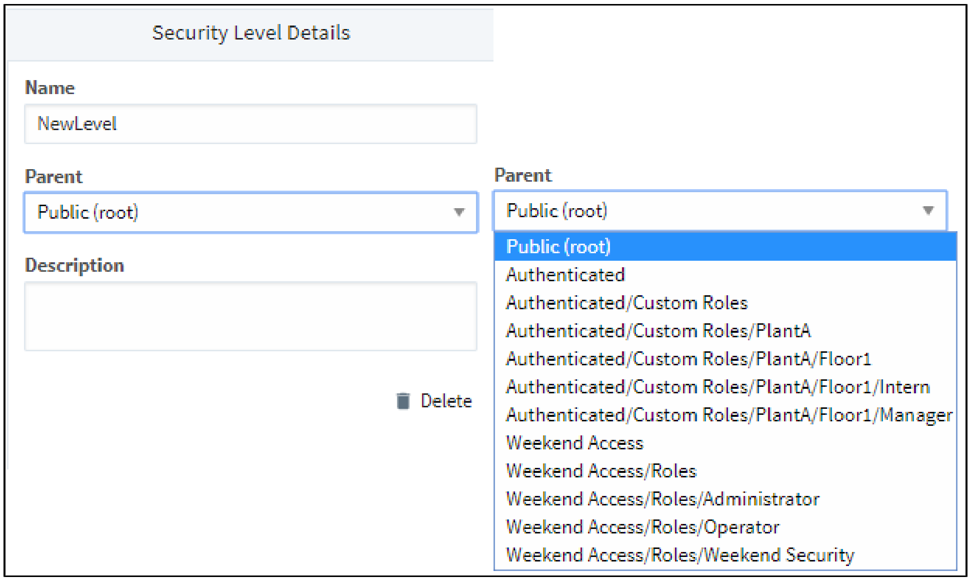
Add a New Security Level

1. From the Gateway Webpage **Config** tab, click on **Security > Security Levels**.
2. In the Security Level tree, select the level that will be a parent for the new level.
3. Click the **Add Security Level** button.

4. In the **Security Level Details** screen area, enter the **Name** for the level.

> Security Level names within the same parent must be unique.

5. The path for the parent is filled in automatically. Use the **Parent** dropdown list if you want to change the parent for this new level.



Security Level Details

Name: NewLevel

Parent: Public (root)

Description

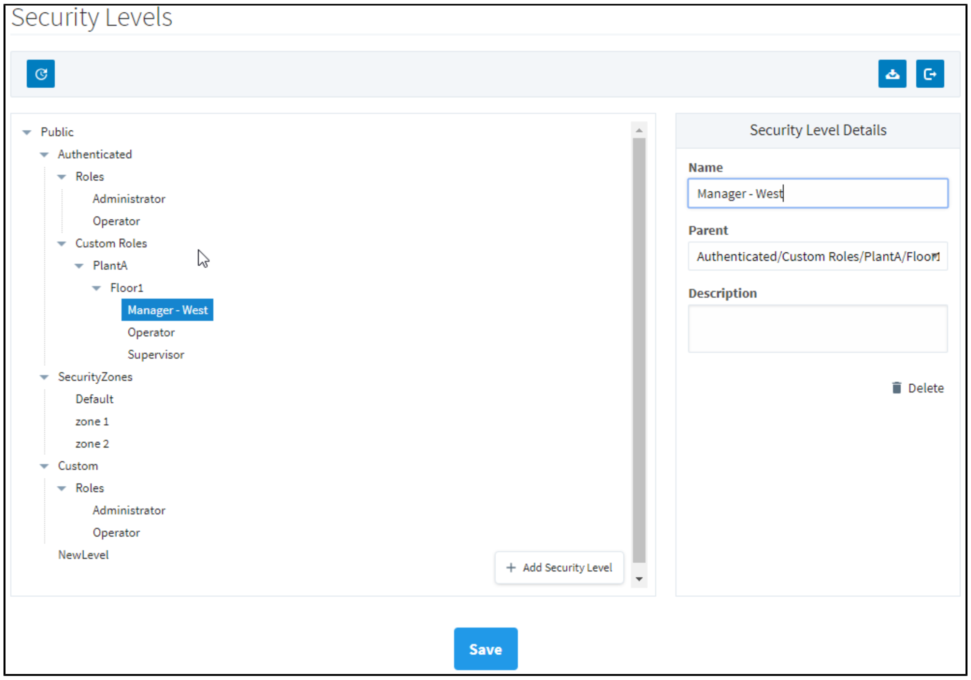
Parent: Public (root)

Delete
Public (root)
Authenticated
Authenticated/Custom Roles
Authenticated/Custom Roles/PlantA
Authenticated/Custom Roles/PlantA/Floor1
Authenticated/Custom Roles/PlantA/Floor1/Intern
Authenticated/Custom Roles/PlantA/Floor1/Manager
Weekend Access
Weekend Access/Roles
Weekend Access/Roles/Administrator
Weekend Access/Roles/Operator
Weekend Access/Roles/Weekend Security

6. Add a **Description** for the new level (optional).
7. If you make changes to Security Levels, but decide not to save them, you can use the Reset button to return the tree view to its currently saved configuration.
8. To save the changes, press **Save**.

Editing Security Levels

1. From the Gateway Webpage **Config** section, click on choose **Security > Security Levels**.
2. In the Security Level Tree, select the level that you want to edit.



Security Levels

Security Level Details

Name: Manager - West

Parent: Authenticated/Custom Roles/PlantA/Floor1

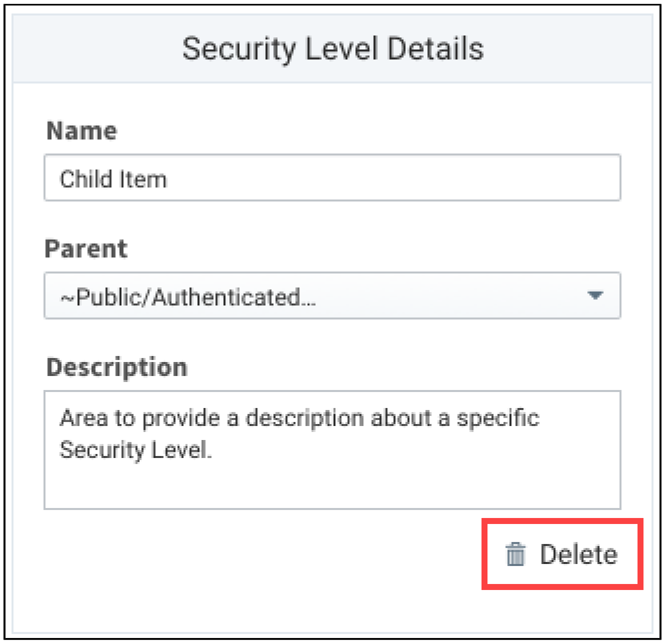
Description

3. Make the desired changes in the **Security Levels Details** screen area.
4. If you make changes to Security Levels, but decide not to save them, you can use the Reset button to return the tree view to its currently saved configuration.
5. To save the changes, click the **Save** button.

Delete a Security Level

> ⚠ When you delete a Security Level, all children under that level will also be deleted.

1. From the Gateway Webpage **Config** tab, click on **Security > Security Levels**.
2. In the Security Level tree, select the level that you want to delete.
3. Click the **Delete** button on the **Security Level Details** screen.



Security Level Details

Name: Child Item

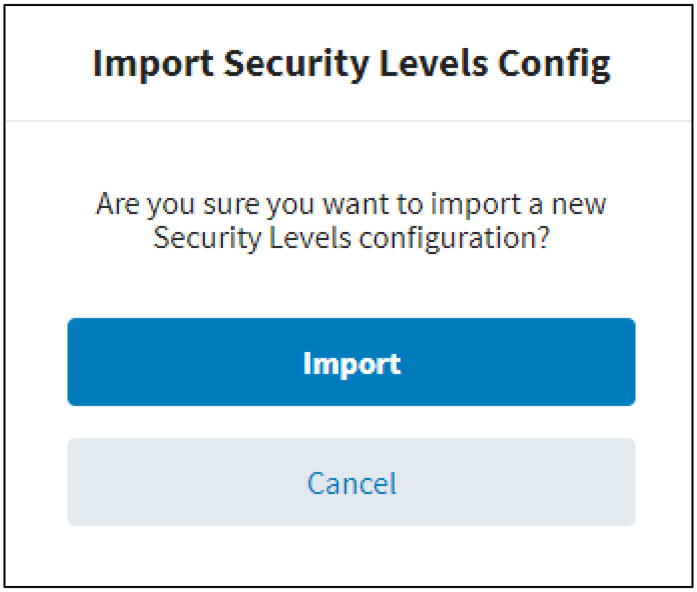
Parent: ~Public/Authenticated...

Description: Area to provide a description about a specific Security Level.

4. In the confirmation box, click **Delete** to confirm the delete.

Import a Security Levels Configuration

1. From the Gateway webpage **Config** tab, click on **Security > Security Levels**.
2. Click the Import icon .
3. Choose **Import** on the confirmation screen.



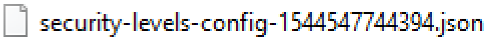
Import Security Levels Config

Are you sure you want to import a new Security Levels configuration?

4. Choose a security levels configuration file to import.
5. Click **Open**.

Export a Security Levels Configuration

1. From the Gateway Webpage **Config** tab, choose **Security > Security Levels**.
2. Click the Export icon .
3. The security levels configuration will be saved as a .json file with a unique number, for example:

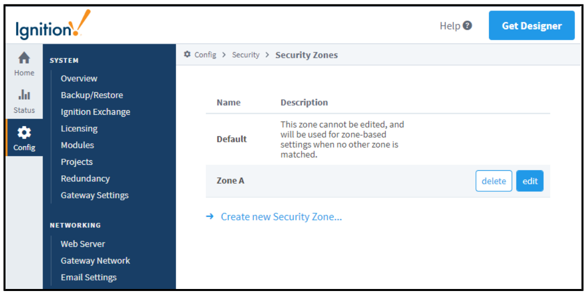


security-levels-config-1544547744394.json

Security Zones

What Are Security Zones?

A Security Zone is a list of Gateways, Computers, or IP addresses that are defined and grouped together. This group now becomes a zone on the Gateway Network, which can have additional policies and restrictions placed on it. While Users and Roles restrict access to specific functions within the Gateway like making certain controls read-only for certain users and read/write for others, Security Zones provide this functionality to the Gateway Network, limiting locations instead of people to be read-only for specific actions. This allows for greater control over the type of information that is passing over the network, improving security and helping to keep different areas of the business separate, while still allowing them to interconnect.



Name	Description
Default	This zone cannot be edited, and will be used for zone-based settings when no other zone is matched.
Zone A	

INDUCTIVE UNIVERSITY

Security Zones and Service Security

Watch the Video

Using Security Zones

Sometimes, in addition to knowing who the user is, it is important to know their location. An operator may have permissions to turn on a machine from an HMI, but if the operator is logged into a project on a different Gateway in the network that has remote access to those Tags, it might not be a good idea to let the operator write to those Tags from a remote location. The operator can't see if the physical machine is clear to run.

This is where Security Zones come in. While Security Zones themselves don't define the security, they instead define an area of the Gateway Network, breaking up Gateways and network locations into manageable zones that can then have a Security Policy set on them. Once there are zones defined, a Security Policy can be assigned to each zone, and a priority of zones can be set in the event that more than one zone applies in a given situation.

> When using zone-based security in a project, the project stores the name of the security zone as a string. This means that if you were to modify the name of the zone in the Gateway, the zone-based security in your project will not update to reflect the new name, and instead will try searching for a zone with the original name. Be very careful when modifying the names of security zones.

Define a Security Zone

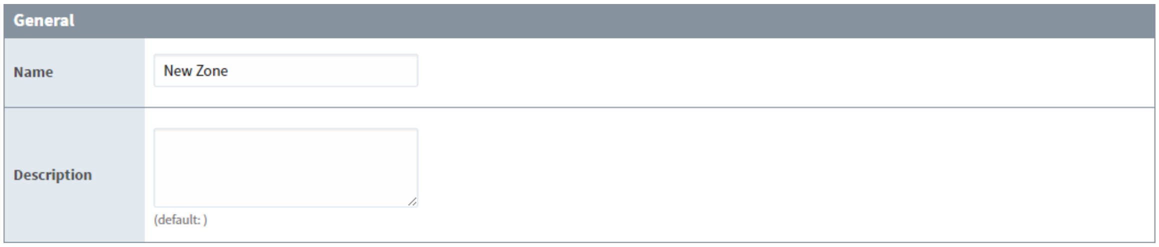
When setting up a new Security Zone, it is a good idea to setup a Gateway Network first if you haven't already. While Security Zones can be defined and used without a connected Gateway, they work best when used in conjunction with other Gateways on a Gateway Network.

Security Zones are defined under the **Config** tab of the Gateway Webpage by going to **Security > Security Zones**.

> There is a special zone called Default. It is always present and can't be modified, and will be used if an incoming connection does not match any of the other defined zones.

Selecting the **Create new Security Zone** link brings up a new page where the Security Zone can be defined.

Besides the name and description, there are two major sections here: Identifiers and Qualifiers. These are two forms of checks that need to happen before an incoming connection gets placed in a Security Zone.



General	
Name	New Zone
Description	(default:)

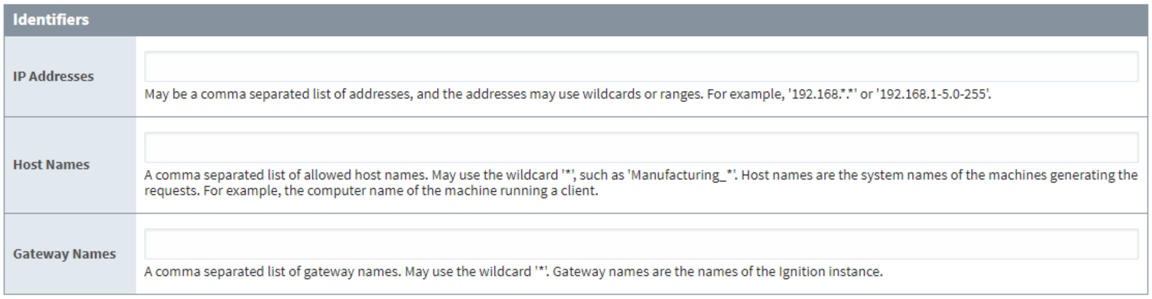
Identifiers

The identifiers are how incoming connections are distinguished between different zones. While there are a few different ways to define the incoming connection, it only needs to match one of them to match this zone.

- **IP Addresses** - This defines an IP address that the connection is coming from. This can be a list of IP addresses by using commas to separate them. It can also make use of the (*) wildcard like '192.168.100.*', or use a range such as '100.100.1-100.0-255'. With IP addresses, virtually all connections can be listed.
- **Host Names** - The host name refers to the system name of the machine generating the request such as Joe_Workstation. This can be a list of names separated by commas, and it can also use the (*) wildcard like '*_Workstation'.
- **Gateway Names** - The Gateway name is the name of the Ignition instance. This is set in the **Config** section by going to **System > Gateway Settings**, and changing the System Name. This can be a comma separated list and can use the wildcard such as '* Ignition Gateway'.

Many incoming connections can be defined using any of the three identifiers, or even multiple at once. As mentioned before, an incoming connection only needs to match one of the identifiers for it to be accepted.

> ⚠ When identifying a Gateway through a proxy Gateway, the IP Address should be using the IP of the proxy, but the Gateway name should use the name of the Gateway we are trying to identify.



Identifiers	
IP Addresses	May be a comma separated list of addresses, and the addresses may use wildcards or ranges. For example, '192.168.*.*' or '192.168.1-5.0-255'.
Host Names	A comma separated list of allowed host names. May use the wildcard '*', such as 'Manufacturing_*'. Host names are the system names of the machines generating the requests. For example, the computer name of the machine running a client.
Gateway Names	A comma separated list of gateway names. May use the wildcard '*'. Gateway names are the names of the Ignition instance.

Qualifiers

After first being identified as part of a particular Security Zone, the connection then must check the Qualifiers. With the Qualifiers, the incoming connection needs to fit in with all of the properties before it is fully placed into the Security Zone.

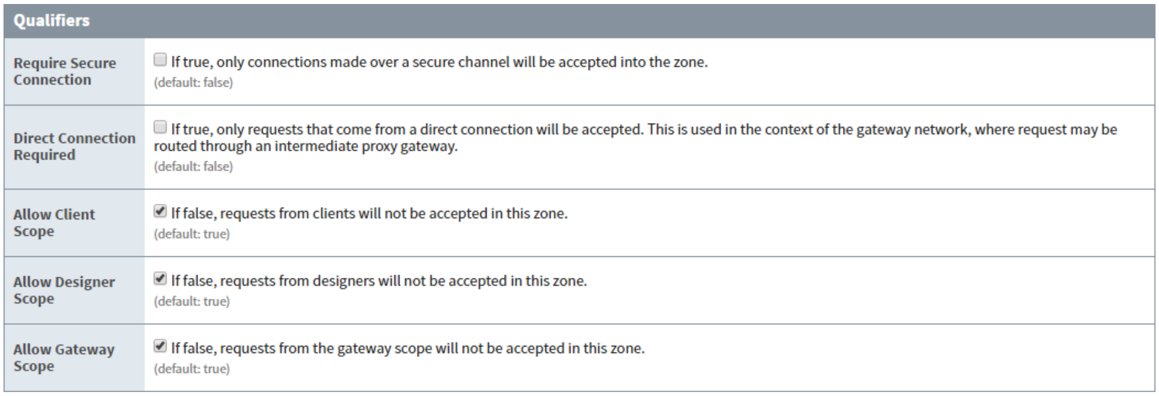
- **Require Secure Connection** - If this is true, only connections that are made over a secure channel will be accepted.

  > ⓘ A Security Zone is only able to check connections made to the Gateway the zone is defined on. This means that connections through a proxy Gateway may not be secure.

- **Direct Connection Required** - If this is true, only connections that come from a direct connection will be accepted. The Gateway Network allows you to connect three Gateways in a 1-2-3 configuration, where Gateway 1 can see Gateway 3 through the proxy Gateway 2.
- **Allow Client Scope** - If this is false, any client scoped requests will not be accepted.
- **Allow Designer Scope** - If this is false, any Designer scoped requests will not be accepted.
- **Allow Gateway Scope** - If this is false, any Gateway scoped requests will not be accepted.

As mentioned above, a connection must pass all of the qualifier checks before being accepted into a Security Zone. So if Require Secure Connection was checked, and Allow Client Scope was not, any requests coming from Clients would be rejected even if they are secure, and the same goes for any non-secure connections coming from sources other than a Client.

Requests can be a part of more than one zone, depending on how the zones are setup. This can be useful for making a whole section of IP addresses read only, but a specific Gateway in that IP address range may be listed specifically in another zone, which can be given read /write access. Any connection which does not fall into one of the zones will be placed in the Default zone.



Qualifiers	
Require Secure Connection	☐ If true, only connections made over a secure channel will be accepted into the zone. (default: false)
Direct Connection Required	☐ If true, only requests that come from a direct connection will be accepted. This is used in the context of the gateway network, where request may be routed through an intermediate proxy gateway. (default: false)
Allow Client Scope	☑ If false, requests from clients will not be accepted in this zone. (default: true)
Allow Designer Scope	☑ If false, requests from designers will not be accepted in this zone. (default: true)
Allow Gateway Scope	☑ If false, requests from the gateway scope will not be accepted in this zone. (default: true)

After entering all the information on the Security Zone page, click **Create New Security Zone**. The page will refresh and you will see a green banner stating that your new Security Zone was successfully created.

Service Security

After creating some Security Zones, a Security Policy can then be defined for each zone. This can be found by going to the **Config** section of the Gateway Webpage and navigating to **Security > Service Security**. At first, none of the zones will have a policy defined, and the Default zone will be at the top. Editing any of them will bring up the Security Policy definition page for that zone. The Security Policy has four sections: Alarm Notification, Alarm Status, History Provider Access, and Tag Access. They work together to define how the local Gateway gives access to incoming Gateway connections. All four sections also have the ability to completely block access to specific services with the Service Access setting in each section. Setting that to deny will deny the zone access to that particular information, regardless of what the rest of the options are set to.

> ⚠ It is important to realize that if you have a single Gateway, limiting access of certain clients to certain Tags is still done in the individual Tags.

- **Alarm Notification** - The Accessible Pipeline Filter setting is a list of Pipelines in the current Gateway that other connections can use for alarm notification. Pipelines must be entered in the format "project_name/pipeline_name". The list is a comma separated list, and it can make use of the (*) wildcard. This setting is an inclusionary list not an exclusionary list, meaning that if there are no pipelines listed here, then all of them will be available.

- **Alarm Status** - The Allow Acknowledge setting will allow the Gateways that fall within the zone to acknowledge alarms on the local Gateway. The Allow Shelving setting will allow the Gateways that fall within the zone to Shelve alarms on the local Gateway. IE: Other Gateways can shelve alarms on this Gateway. For this Gateway to shelve alarms on others, this must be set on the remote Gateway.

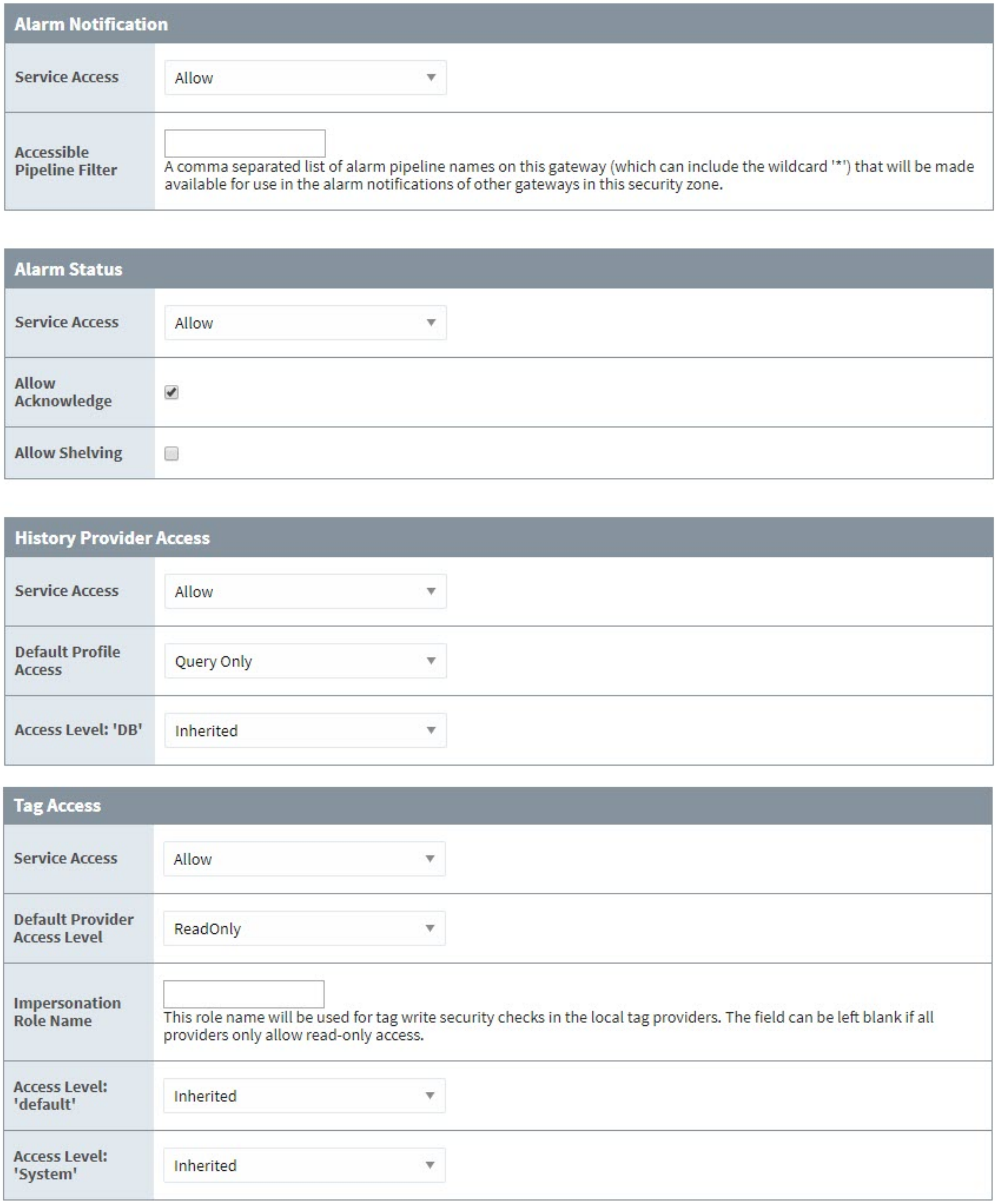
- **History Provider Access** - The History Provider Access has two different settings. First, it has a Default Access Profile. This is the default access rights for Tag History. Second, there will be a setting for each History Provider setup on the local Gateway. In the image below, there is an "Access Level: 'New Connection" that can be set that corresponds to the History Provider that was created when a database was connected. It can be set to Query and Storage, which will allow connections in the current zone to both run queries and store Tag History against the Tag History provider, Query Only, which will only allow the zone to query out history data, but not store it, and No Access, which will completely block access to that History Provider. The final setting is Inherited, which will inherit the Default Profile Access rights. Any new history providers will automatically get added to the Security Policy set at inherited so it may be beneficial to set the Default Profile Access to be either Read Only or No Access so that a recently added history provider does not accidentally get storage rights when it should not.

  > ⚠ The Default Access Profile should not be set to Inherited. This also goes for the Default Provider Access Level in the Tag Access section.

- **Tag Access** - The Tag Access section also has a few different settings. The Default Provider Access Level, which works the same as the History Provider Default Access Profile. It sets the default access rights for realtime providers. It will then have a setting for each provider configured in the local Gateway, as well as an additional one for system Tags. These can be set to **ReadWriteEdit**, which will allow connections in the current zone to read, write to, and edit the Tags in that provider, **ReadWrite**, which allows the zone to read and write to Tags, and **ReadOnly**, which only allows the zone to read the Tags. It also can be set to **None**, which will prevent the zone from interacting with the Tag Provider altogether, and **Inherited**, which will again inherit the access rights set in the Default Provider Access Level. Any new Tag Providers will automatically get added to the Security Policy with Inherited access rights. The Trust Remote Roles setting will allow similarly named roles on other Gateways to access Tags with role specific security on them.

> The following feature is new in Ignition version **8.0.3**
> Click here to check out the other new features

The Impersonation Role Name field allows you to specify a role name to use when writing to a Tag from an incoming Gateway Network connection (from the selected Zone). This replaces the Trust Remote Roles setting, which is only available in 8.0.0 to 8.0.2. Note that you must use this role if you are configuring Tag Security.



Alarm Notification

Service Access: Allow

Accessible Pipeline Filter: A comma separated list of alarm pipeline names on this gateway (which can include the wildcard '*') that will be made available for use in the alarm notifications of other gateways in this security zone.

Alarm Status

Service Access: Allow

Allow Acknowledge: ☑

Allow Shelving: ☐

History Provider Access

Service Access: Allow

Default Profile Access: Query Only

Access Level: 'DB': Inherited

Tag Access

Service Access: Allow

Default Provider Access Level: ReadOnly

Impersonation Role Name: This role name will be used for tag write security checks in the local tag providers. The field can be left blank if all providers only allow read-only access.

Access Level: 'default': Inherited

Access Level: 'System': Inherited

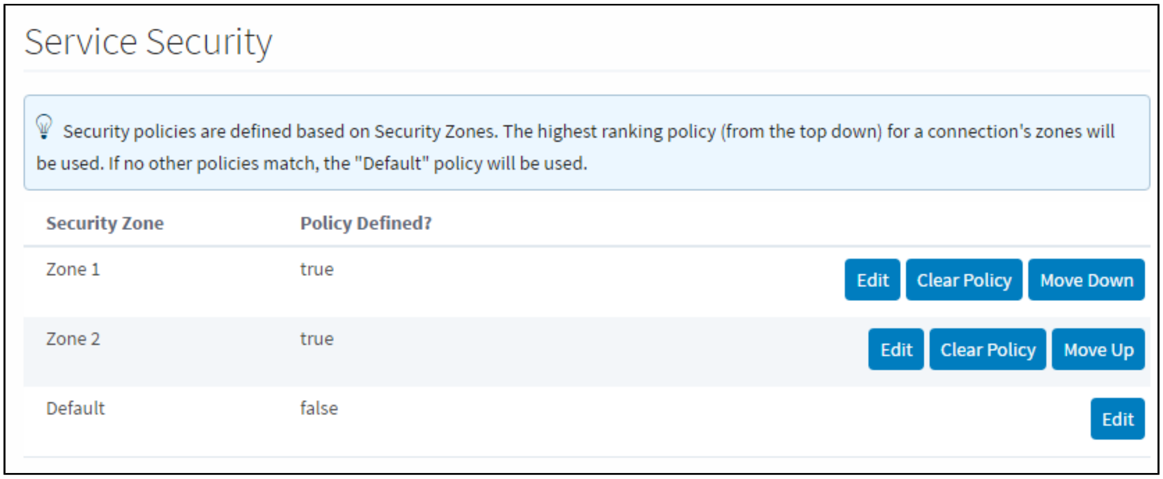
Default Security Zone

While the Default zone may not have a custom Security Policy defined, it does default to not include any notification pipelines, allow alarm acknowledgment, query only history access, and read only Tag access. This means that if a remote Tag Provider is setup on a remote Gateway, and the local Gateway has not changed the default security settings, the remote Gateway will have read only access to the

Tag History Provider. This can be changed by editing the Default zone's Security Policy to fit a different preference, or creating new Security Zones with custom security policies. Once a Security Policy has been defined on a zone, it will automatically jump to the top of the list. A new option will also become available that will clear the policy from the zone.

Setting Zone Priority

Once a Security Policy has been defined for two or more zones, a new option appears on the Service Security page to move the zones up and down the list. This allows a priority to be set on the Security Zones, since a connection can apply to multiple zones. For example, say Zone 2 dictates that all requests coming from a range of IP addresses have query only history access, and read write access to Tags. Zone 1 includes specific Gateways, one of which is also contained in Zone 2, that will have query and storage history access and read write edit access to Tags. When a request comes in from a connection, it first determines which Security Zones it belongs to. The request then starts at the top of the Service Security list and goes down until it finds the first zone that it is in, and uses the access rights of that zone. In our example, we want to make sure Zone 1 is above Zone 2, so that the Gateway that is in both Zone 1 and Zone 2 gets the full access rights afforded to it by the Security Policy of Zone 1 instead of getting the limited access rights from Zone 2.



Service Security

Security policies are defined based on Security Zones. The highest ranking policy (from the top down) for a connection's zones will be used. If no other policies match, the "Default" policy will be used.

Security Zone	Policy Defined?
Zone 1	true
Zone 2	true
Default	false

Related Topics ...

- Security in Vision

User Schedules

Schedules

Schedules define the times of users on-call availability and unavailability. For example, the **Always** schedule is a schedule that is active 24/7. You can set a schedule for each user in the alarm notification system. The notification messages are then sent only to those users with an active schedule. When a message reaches a notification block in a pipeline, that block's on-call roster is used to find the users with active schedules so they can be notified.

There are also a number of system functions that allow you to create, read, edit, and delete schedules or holidays from a user source using scripting. (i.e., system.user.addSchedule, system.user.getHoliday, etc.,). To learn what system functions are available for user scheduling and holiday scripting, refer to the Scripting Functions in the Appendix.

On this page

...

User Schedules

Watch the Video

Define a New Schedule

1. Go to the **Config** section of the Gateway.

2. Choose **Alarming > Schedules** from the menu on the left.
   The Schedule Management page is displayed. Here you can see an Always and an Example schedule.
   The **Always** Schedule is a built-in schedule that is always available: 24x7x365.
   The **Example** Schedule is an example of a M-F 8am-5pm schedule with a lunch break. Click on the **edit** to see the detailed settings.

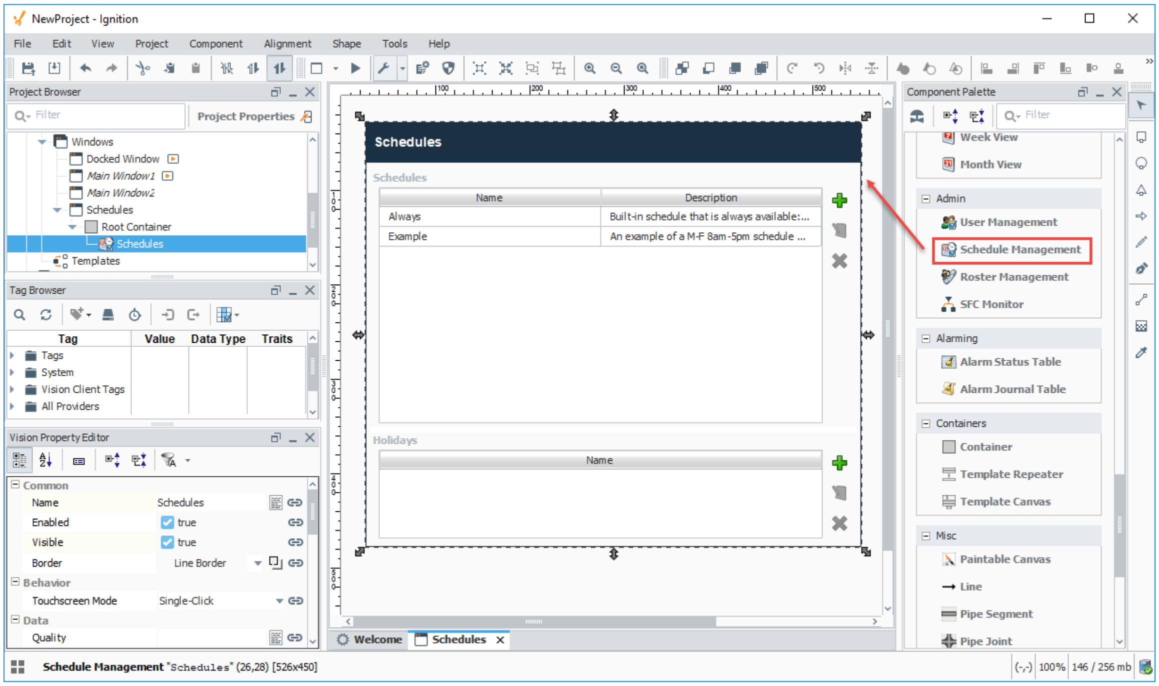
3. Click on **Create new Schedule**.

4. Choose a **Standard** or **Composite** schedule and click **Next**.
   The **Standard** schedule is defined by hours of operation on each day, with repeating options.
   The **Composite** schedule is defined by the combining of two other schedules.

5. On the **New Schedule** page, enter a **Name**, the **Schedule** to be notified, and any **Repeating** options, then click **Add Schedule**.
   Each schedule is defined by which days of the week it is active for, and during what times. Additionally, you can define repeating patterns that repeat either weekly or daily, and even plug in holidays. When defining time ranges, use 24-hour format and commas to break up different spans. For example, to if you want the schedule to be active from 8am-5pm with a 45 minute break starting at noon, you'd use: 8:00-12:00, 12:45-17:00.

Managing User Schedules from the Vision Client

There are a few ways to manage user schedules from the Vision client. The first is to use the Schedule Management component on a window. This component allows you to quickly and easily manage the schedules from the Vision client.



Name	Description
Always	Built-in schedule that is always available:...
Example	An example of a M-F 8am-5pm schedule ...

Name

Name: Schedules
Enabled: ☑ true
Visible: ☑ true
Border: Line Border
Touchscreen Mode: Single-Click
Quality

Schedule Management "Schedules" (26,28) [526x450]
(-,-) 100% 146 / 256 mb

For more granular control, you may instead want to use scripting to manage the schedules. This may offer a more granular control at the click of a button.

```
# This code creates a new schedule by using an old schedule but setting observe holidays to true.
mySchedule = system.user.getSchedule("WeeklySchedule")
if mySchedule != None and mySchedule.getType() == "basic schedule":
        mySchedule.setObserveHolidays(True)
        mySchedule.setName("NewWeeklySchedule")
        system.user.addSchedule(mySchedule)
```

Using Schedules for Alarm Notification

The alarm notification system always uses the Schedules. When an alarm notification pipeline reaches a notification block, it looks at all of the users defined in that block's configured on-call roster. Only those users whose schedules are currently active will be notified. This way, you can group people in call rosters by role, not by shift. For example, suppose you have alarms that should be sent to all supervisors. You can put all of the supervisors in one call roster, and the scheduling system will automatically only notify those supervisors who are on-shift when the alarm goes active.

Using Schedules for Restricting Login

You can use Schedules to restrict users' ability to log in. To enable this, select the **Schedule Restricted** option on the user source in question. That user source will then reject logins for users whose schedule is inactive, even if their credentials were accurate.

Related Topics ...

- On-Call Rosters
- User Notifications

Project Security in Designer

When several users are all working on the same project, managing changes to the project can become cumbersome. By default, all users with Designer access can modify, delete, save, and publish all resources available in the Designer. In some situations, it is desirable to limit what each user can do in the Designer. Ignition has several built-in Designer restriction methods to help in these scenarios.

Designer Project Permissions

Actions such as publishing, viewing, saving, deleting, and editing of project resources are restricted to users who have sufficient roles to do so. Editing of the these required roles is done in the permissions section of the Project Properties dialog in the Designer. If required roles are not set for an action, then all users with Designer access can perform the action.

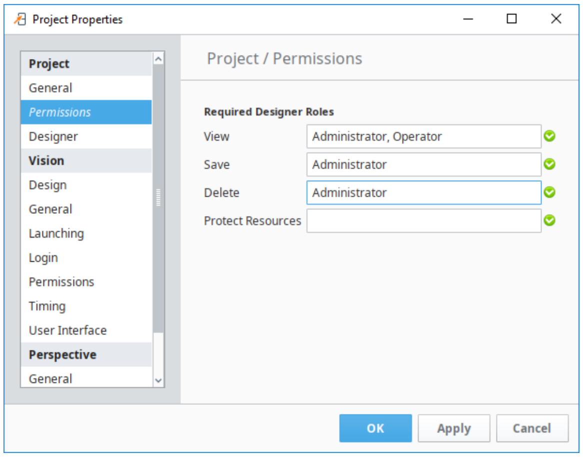
The Designer does not poll for role changes, so if a user who is currently logged into the Designer has their roles changed, they will need to re-launch the Designer for the new role(s) to take effect.

On this page ...

Controlling Project Edits by Role

You can control who gets to login to a project by assigning roles and giving permissions to those roles in the Required Designer Roles property which you set up in the Designer.

1. In the Designer, from the menubar, choose **Project > Properties**.
2. Go to the **Project > Permissions** area, and under the Required Designer Roles, enter the appropriate roles next to each project level restriction, as required.
   In the text boxes on this page, enter a comma-separated list of role names that
   are required to access the project. As you start typing, matching role names will pop up.



Project / Permissions

Required Designer Roles

View: Administrator, Operator

Save: Administrator

Delete: Administrator

Protect Resources:

Project Permissions

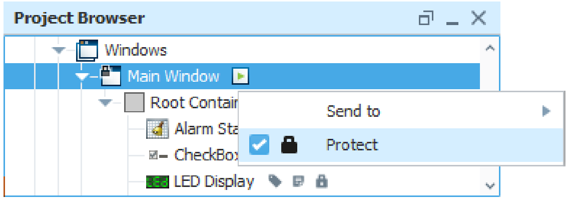
Watch the Video

The following table describes each of these five options:

Option	Affect
View	User must have at least one of these roles to view the project in the Designer.
Save	User must have at least one of these roles to save the project.
Delete	User must have at least one of these roles to delete the project.
Protect Resources	User must have at least one of these roles to access protected resources.

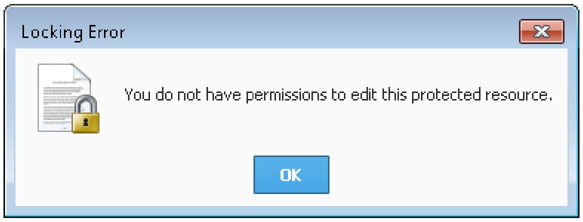
Protecting Project Resources

You can lock individual project resources from inside Designer by opening the Project Browser, and right clicking on any of the objects that you want to lock in. Select the **Protect** option to protect it. Once it's protected, it cannot be changed except by someone that has the permission to unprotect it, and modify it.



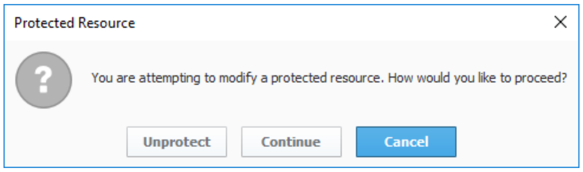
Protected resources are global or project resources that can only be edited by select users with the required roles. These roles are required to protect resources from being edited in the Designer, and do not apply to the clients. This means you can prevent a resource from being edited by other users who have Designer access. It is often used in scenarios where development work is finished on a window or object, and no further changes should be made to it. Other objects like Vision Templates or Alarm Pipelines are often protected so they may be used, but not modified.

Users without a required role will see the following message in the Designer when attempting to open a protected resource:



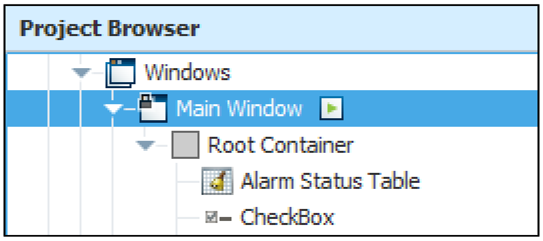
You do not have permissions to edit this protected resource.

Users with a required role are allowed to modify the resource, but a message will appear informing them that the resource is protected, and will be asked how to proceed:



You are attempting to modify a protected resource. How would you like to proceed?

Additionally, a lock icon will appear on the resource informing users that it is protected. An example can be seen on the 'Audit Events' window below:



To remove the protection, simply right click the object and select the **Protect** option to unprotect it.

INDUCTIVE UNIVERSITY

Locking Project Resources

Watch the Video

Using SSL

What Is SSL?

Secure Socket Layer (SSL) is a widely used security protocol for data as it goes across a network or the internet. SSL and TLS are protocols that secure the network traffic of the Gateway. This means using HTTPS for all traffic between the Designer, Vision Clients, and Perspective Sessions when connecting to the Gateway. It is highly recommended that SSL / TLS is always used as it will help protect against your data being vulnerable.

On this page

...

Enabling SSL

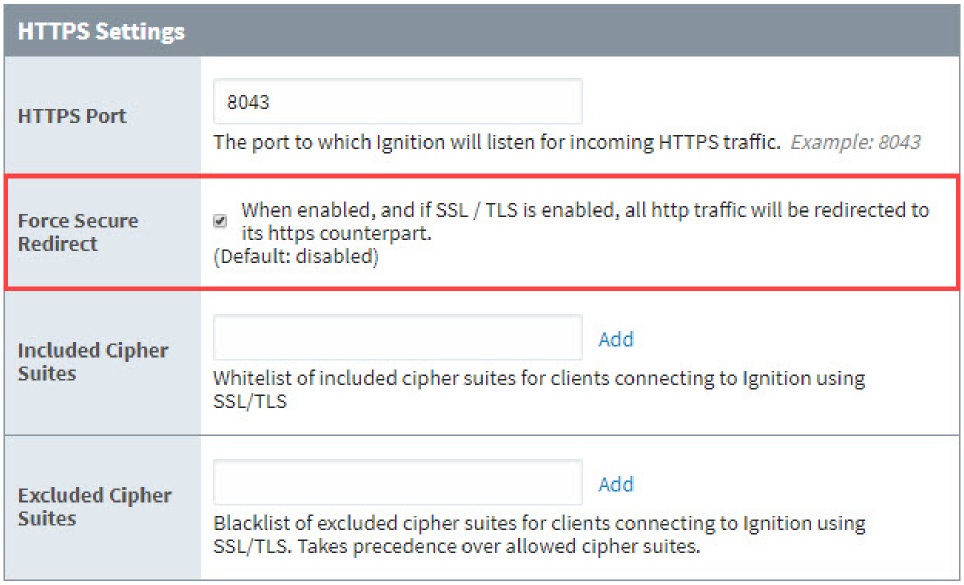
To enhance security in Ignition, you may opt to enable SSL encryption. This will affect all communication to and from the Gateway that is done over the HTTP protocol. This includes not only browsers interacting with the Gateway's web interface, but all Vision Client communication as well. Turning on SSL will encrypt all data sent over HTTP. This protects your installation from anyone "snooping" the data as it passes over the network. This may be important if data transferred between the Gateway and Clients is sensitive in nature. This also helps to thwart a security vulnerability known as "session hijacking".

Requiring SSL

Watch the Video

Turn on SSL

1. Go to the **Config** section of the Gateway Webpage.
2. Choose **Networking> Web Server** from the menus on the left.
3. Select the checkbox for Force Secure Redirect**,** and click the **Save** button at the bottom of the page.



HTTPS Settings

HTTPS Port	8043 The port to which Ignition will listen for incoming HTTPS traffic. *Example: 8043*
Force Secure Redirect	☑ When enabled, and if SSL / TLS is enabled, all http traffic will be redirected to its https counterpart. (Default: disabled)
Included Cipher Suites	Add Whitelist of included cipher suites for clients connecting to Ignition using SSL/TLS
Excluded Cipher Suites	Add Blacklist of excluded cipher suites for clients connecting to Ignition using SSL/TLS. Takes precedence over allowed cipher suites.

After SSL is enabled, all Clients, Designers, and web browsers are redirected to the SSL port if they try to use the standard HTTP port. By default, the SSL port is 8043. You can change it here to another port like the standard SSL port of 443. If you are using a firewall, make sure to open the appropriate ports.

Adding an SSL Certificate

Ignition supports using SSL communications to the Gateway Webpage as well as Client/Designer communication with the Gateway. It is highly recommended that you purchase an SSL certificate from a certificate authority if you turn this feature on. The procedures for how to install a genuine SSL Certificate are below.

When you turn on SSL in Ignition, the web browser uses what is called a "self-signed" certificate. This gives you the encryption benefits of SSL, but not the identity validation, and it isn't a "real" certificate. This is why a web browser will display nasty warnings to users that they shouldn't trust the website.

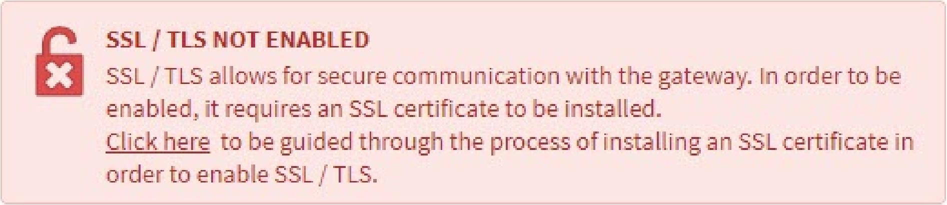
We are not able to ship a real certificate with Ignition because security certificates have to be obtained individually from a Certificate Authority (CA). Ignition supports certificates from both your organization's internal CA, as well as commercial CAs (Verisign, GoDaddy, Comodo, etc). In either case, the procedure for how to install a certificate is listed below.

> After you have added an SSL certificate, the KeyStore will automatically refresh every 15 minutes. You can disable this in the ignition.conf file by altering the ignition.ssl.refresh entry (Set to 0 to not refresh).

Get a Certificate Signing Request

Since SSL/TLS requires the installation of an SSL certificate, filling out the form below will generate a certificate signing request (CSR) to provide to a certificate authority. It is the first step required in getting an SSL certificate from a trusted Certificate Authority (CA), which is why details such as Organization and Location are being collected.

1. Go to the Config tab of the Gateway Webpage and choose **Networking > Web Server.**
2. You'll see a warning message indicating that SSL/TLS is not enabled. Click on the **Click here** link.



SSL / TLS NOT ENABLED

SSL / TLS allows for secure communication with the gateway. In order to be enabled, it requires an SSL certificate to be installed.
Click here to be guided through the process of installing an SSL certificate in order to enable SSL / TLS.

3. Click on the **I don't have all the items above button.** The Create Certificate screen is displayed.
4. Fill in the required fields on the screen, then click the **Generate Certificate Signing Request** button. This can be brought to a Certificate Authority.

Basic Details	
Field	**Definition**
Common Name	Full DNS name (required). This is typically what you type in your browser URL bar in order to navigate to this gateway, for example: yourdomain.com
Organization Name	Name of company (required). For example: Inductive Automation.
Organization Department	Department or section (required). For example: Engineering.
Email	Email address. For example: your@email.com.
Country	Typically an ISO 3166 2 character code (required). For example: US.
State / Province	State, province or region, for example: California.
Locality (City)	Name of city. For example: Folsom
Street	Street number and street name. For example: 90 Parkshore Dr

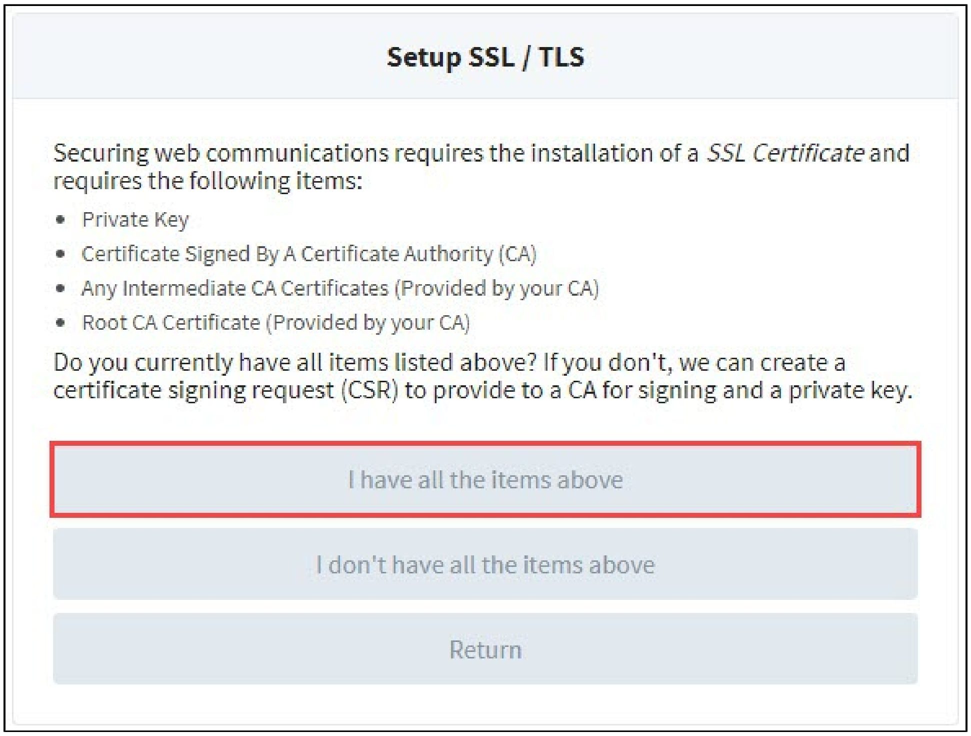
Postal Code	Postal Code Example: 95630
Key Type	The algorithm of the key pair which will be generated for the self signed certificate. Options are RSA or EC. Recommended: RSA
Key Size	The strength of the generated Key. Recommended: 2048 bits
Expires in	The number of days the generated Certificate will be valid. Only applies to the self-signed certificate.
Subject Alternative Names	
Field	**Definition**
IP Addresses	The IP addresses of all the servers you plan on installing the certificate. Click the **Add** button for each additional IP address.
DNS Names	DNS names which map to the list of IP addresses above. Click the Add button for each additional IP address.

Install an SSL Certificate

Once you have an SSL certificate, it needs to be added to Ignition.

1. Go to the Config tab of the Gateway Webpage and choose **Networking > Web Server.**
2. You'll see a warning message indicating that SSL/TLS is not enabled. Click on the **Click here** link.
3. The Setup SSL/TLS screen is displayed. Review the following list:
   - Private Key
   - Certificate Signed By A Certificate Authority (CA)
   - Any Intermediate CA Certificates (Provided by your CA)
   - Root CA Certificate (Provided by your CA)

4. If you have the items, click on the **I have all the items above** button. If you don't have all the items, click on the **I don't have all the items above button**, and follow the previous procedure, To Get an SSL Certificate from a CA.



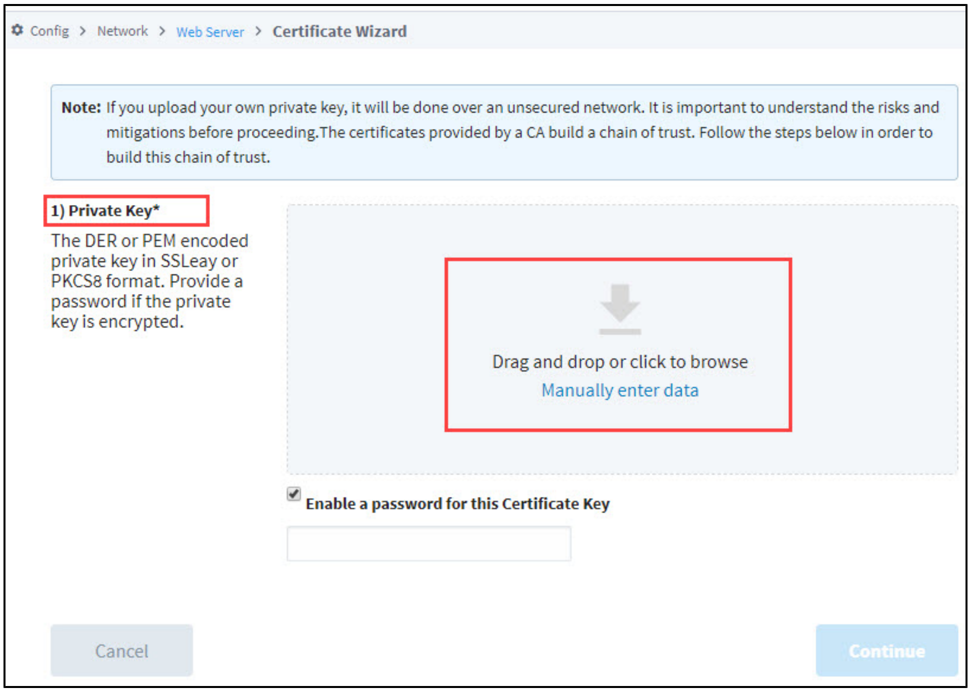
Setup SSL / TLS

Securing web communications requires the installation of a *SSL Certificate* and requires the following items:

- Private Key
- Certificate Signed By A Certificate Authority (CA)
- Any Intermediate CA Certificates (Provided by your CA)
- Root CA Certificate (Provided by your CA)

Do you currently have all items listed above? If you don't, we can create a certificate signing request (CSR) to provide to a CA for signing and a private key.

5. The Certificate Wizard is displayed. The first step is to import your private key in one of the following three ways.
   - Drag and Drop your certificate from your computer onto the screen.
   - Click anywhere on the grey box to browse for the private key.
   - Click **Manually enter data** button to type in the private key information



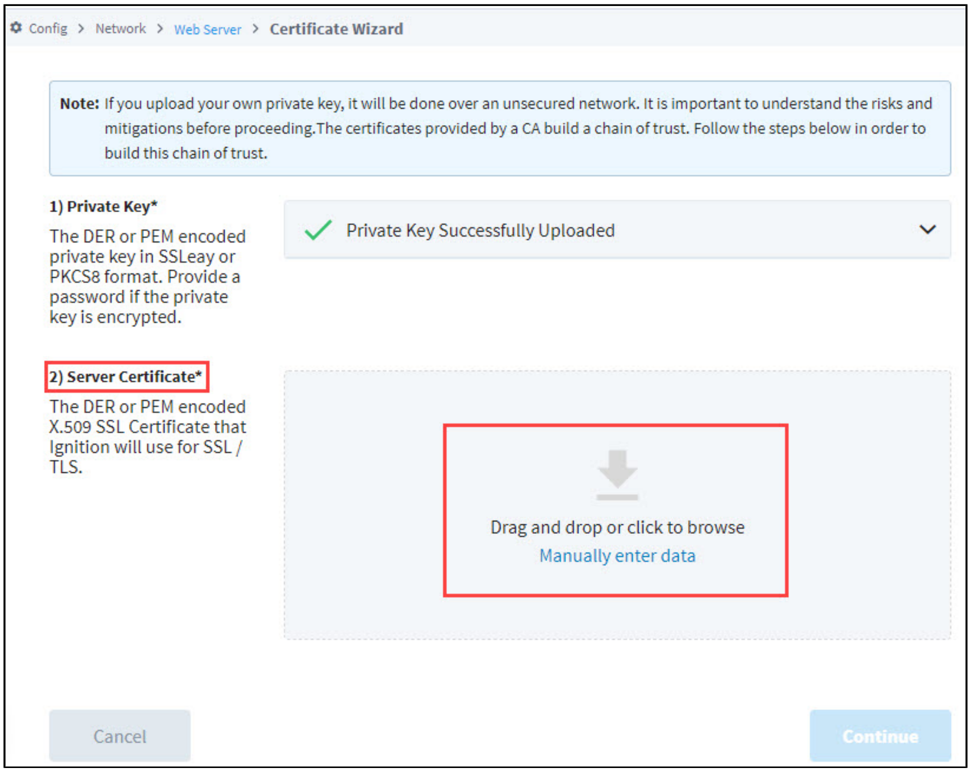
Note: If you upload your own private key, it will be done over an unsecured network. It is important to understand the risks and mitigations before proceeding.The certificates provided by a CA build a chain of trust. Follow the steps below in order to build this chain of trust.

1) Private Key*

The DER or PEM encoded private key in SSLeay or PKCS8 format. Provide a password if the private key is encrypted.

☑ **Enable a password for this Certificate Key**

6. If the private key is encrypted, click the **checkbox** to enable a password for this certificate and enter the password in the field. Click **Continue**.
7. The next step is to import the server certificate. This is the The DER or PEM encoded X.509 SSL Certificate that Ignition will use for SSL / TLS. Drag and drop the certificate file, browse for it, or manually enter the data.



Note: If you upload your own private key, it will be done over an unsecured network. It is important to understand the risks and mitigations before proceeding.The certificates provided by a CA build a chain of trust. Follow the steps below in order to build this chain of trust.

1) Private Key*

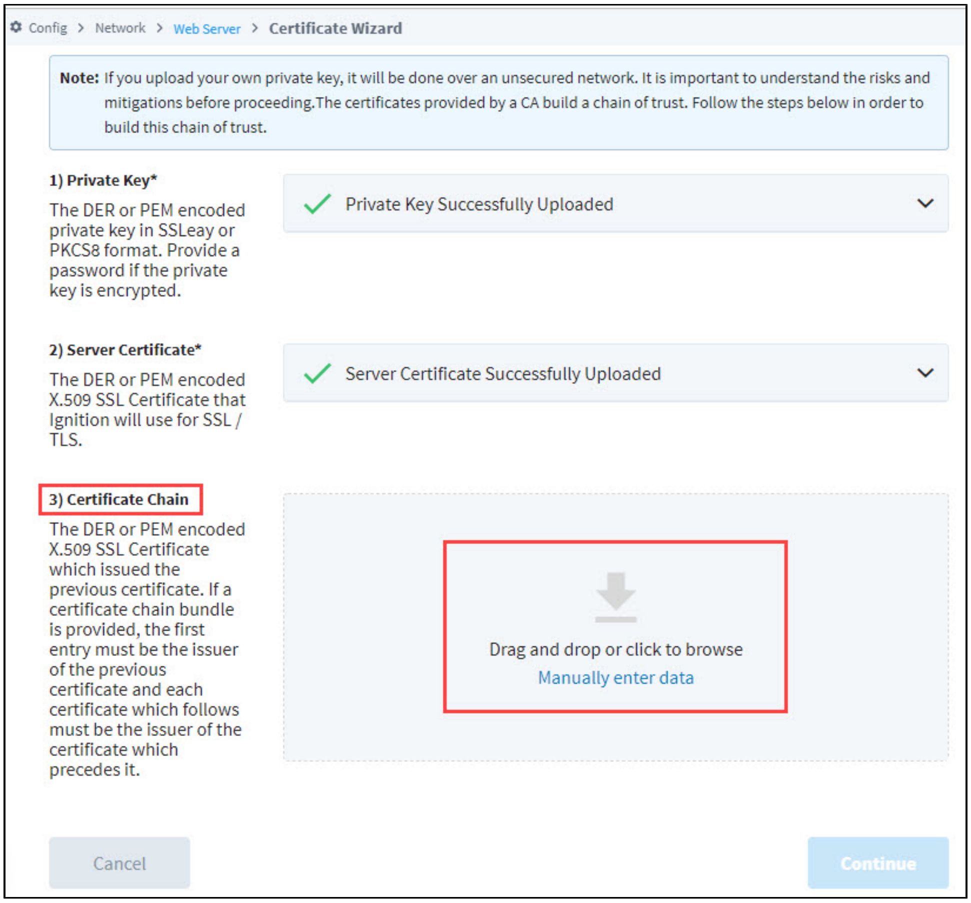
The DER or PEM encoded private key in SSLeay or PKCS8 format. Provide a password if the private key is encrypted.

✓ Private Key Successfully Uploaded

2) Server Certificate*

The DER or PEM encoded X.509 SSL Certificate that Ignition will use for SSL / TLS.

8. The next step is to import the certificate chain. This gives you the Intermediate CA Certificate. Drag and drop the certificate file or bundle, browse for it, or manually enter the data.
You'll see a message that the Intermediate CA Certificate was successfully uploaded.



Config > Network > Web Server > Certificate Wizard

Note: If you upload your own private key, it will be done over an unsecured network. It is important to understand the risks and mitigations before proceeding.The certificates provided by a CA build a chain of trust. Follow the steps below in order to build this chain of trust.

1) Private Key*

The DER or PEM encoded private key in SSLeay or PKCS8 format. Provide a password if the private key is encrypted.

✓ Private Key Successfully Uploaded

2) Server Certificate*

The DER or PEM encoded X.509 SSL Certificate that Ignition will use for SSL / TLS.

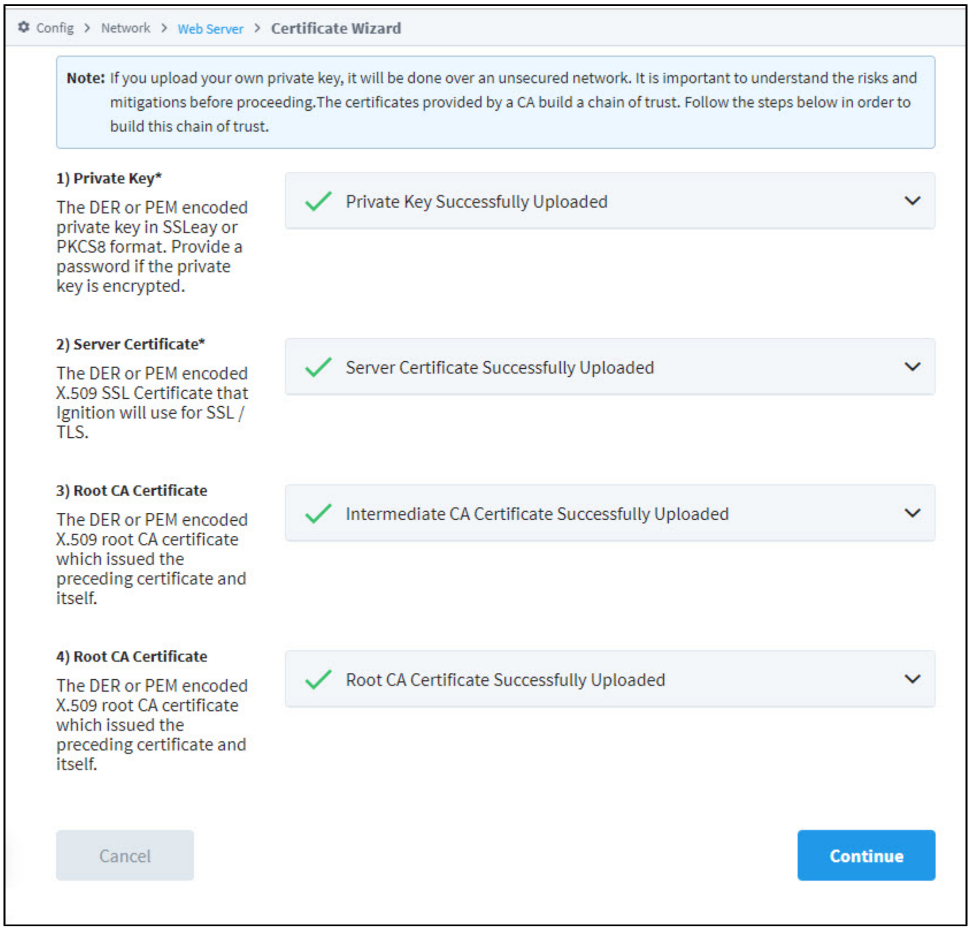
✓ Server Certificate Successfully Uploaded

3) Certificate Chain

The DER or PEM encoded X.509 SSL Certificate which issued the previous certificate. If a certificate chain bundle is provided, the first entry must be the issuer of the previous certificate and each certificate which follows must be the issuer of the certificate which precedes it.

9. Finally, import the root CA certificate: Drag and drop the certificate file, browse for it, or manually enter the data. You'll see a message that the Root CA Certificate was successfully uploaded.

> ***Editor notes are only visible to logged in users***
> Why does the description for the previous certificate, the intermediate one, change to say Root certificate now? - This is a question for Dev., probably a bug. The text is a duplicate of the step 4 too.This is a bug. Alex Lu submitted FB14677 for it.



Config > Network > Web Server > Certificate Wizard

Note: If you upload your own private key, it will be done over an unsecured network. It is important to understand the risks and mitigations before proceeding.The certificates provided by a CA build a chain of trust. Follow the steps below in order to build this chain of trust.

1) Private Key*

The DER or PEM encoded private key in SSLeay or PKCS8 format. Provide a password if the private key is encrypted.

✓ Private Key Successfully Uploaded

2) Server Certificate*

The DER or PEM encoded X.509 SSL Certificate that Ignition will use for SSL / TLS.

✓ Server Certificate Successfully Uploaded

3) Root CA Certificate

The DER or PEM encoded X.509 root CA certificate which issued the preceding certificate and itself.

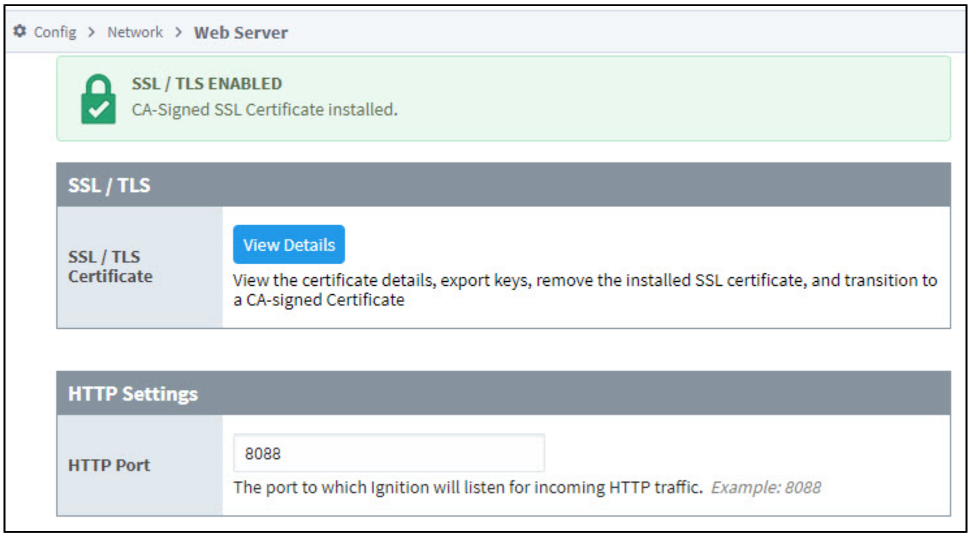
✓ Intermediate CA Certificate Successfully Uploaded

4) Root CA Certificate

The DER or PEM encoded X.509 root CA certificate which issued the preceding certificate and itself.

✓ Root CA Certificate Successfully Uploaded

10. Click the Continue button.
11. You'll see a confirmation message that the certificate is installed and SSL/TLS is enabled.



Config > Network > Web Server

SSL / TLS ENABLED
CA-Signed SSL Certificate installed.

SSL / TLS	
SSL / TLS Certificate	View Details View the certificate details, export keys, remove the installed SSL certificate, and transition to a CA-signed Certificate

HTTP Settings	
HTTP Port	8088 The port to which Ignition will listen for incoming HTTP traffic. *Example: 8088*

12. If you have a redundant installation, you'll need to repeat this procedure on your backup server.

Related Topics ...

- Security
- Security in Perspective

- Security in Vision

Common Security Tasks

This section contains examples for items we identified as "common" tasks. It includes methods and examples of Security features that many users are looking to utilize when first starting out with Ignition. Additionally, this section attempts to clarify some of the Security options available that can be used to protect your applications and data.

The examples in this section are self-contained explanations that may touch upon many other areas of Ignition. While they are typically focused on a single goal or end result, they can easily be expanded or modified after the fact. In essence, they serve as a great starting point for users new to Ignition, as well as experienced users that need to get acquainted with a new or unfamiliar feature.

Below is a list of common tasks related to the security features and setting up security for your Ignition system.

On this page ...

Configuring IdPs

With the introduction of Ignition 8 and Perspective 8.0, we implemented several new features called Security Levels and Identity Providers (IdPs) that improve authentication and authorization to the Ignition system. There are several types of Identity Providers (IdPs) such as Ignition, OpenID Connect 1.0, and Security Assertion Markup Language (SAML). An IdP offers a way for users to log into Ignition using credentials stored outside of Ignition as well as providing authentication services such as two-factor authentication (2FA). Security Levels, similar to roles that are assigned to users, are defined in the Gateway and enable you to define a hierarchy for access inside a Perspective Session. They provide a way for you to map roles from an IdP to Ignition roles.

The OpenID Connect 1.0 Example page will show you how to configure an external IdP that used OpenID Connect 1.0 with your Ignition system.

Troubleshooting IdPs

On the Troubleshooting Identity Providers page, you will find a few helpful examples to help you diagnose and troubleshoot issues with configuring IdPs. They typically include various scenarios that you may find yourself in, and what to do in those situations, as well as some error messages that will be helpful in figuring out what is going wrong.

In This Section ...

OpenID Connect 1.0 Example

This section provides an example of how to connect an Identity Provider that is using the OpenID Connect 1.0 protocol. This example uses the Okta IdP service. Your IdP vendor may differ and the specific links will differ.

On this page ...

Prerequisites

Before you begin configuring Ignition there are some preliminary requirements that need to be done outside of Ignition:

- A configured remote IdP (Okta in this example)
- The metadata file specific to your IdP
- The scope data specific to your IdP
- Login credentials to use as a test

Configured IdP

An IT department is usually the one to setup and configure a remote IdP. You need a configured remote IdP that is compatible with OpenID Connect 1.0. protocol.

At minimum there needs to be an account set up with the IdP, users added to the IdP account, and applications added to the IdP.

Metadata File

You will need the metadata file specific to your IdP. This document defines how to communicate with the IdP. It is usually a web page that allows the metadata file to be exported to a JSON file. Often it is a URL that ends with “.well-known/openid-configuration”.

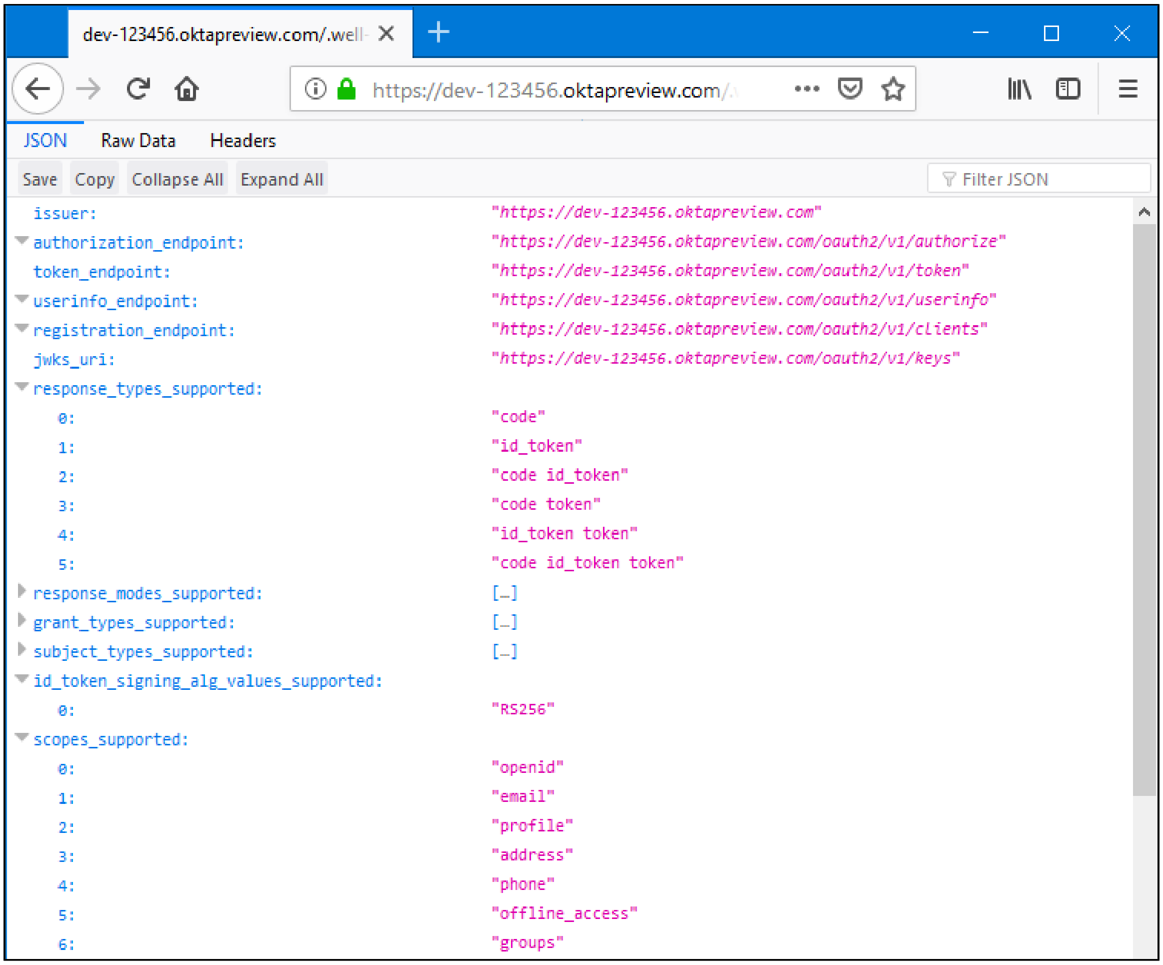
You will need the URL link to this page or a JSON export of this page. For example, if your IdP user login URL is something like this:

https://dev-123456.oktapreview.com/login/login.htm

Then the metadata import URL may look like something like this:

https://dev-123456.oktapreview.com/.well-known/openid-configuration

Here is an example of part of a metadata file for Okta. Notice that the URL link has "/.well-known/openid-configuration" at the end and is very similar to the login URL. It is recommended to use the URL specific to your IdP. Your IT department may choose to export this JSON file of this page and provide it to you. Either option will work.


dev-123456.oktapreview.com/.well-
https://dev-123456.oktapreview.com/
JSON
Raw Data
Headers
Save
Copy
Collapse All
Expand All
Filter JSON

```
issuer:                                   "https://dev-123456.oktapreview.com"
authorization_endpoint:                   "https://dev-123456.oktapreview.com/oauth2/v1/authorize"
token_endpoint:                           "https://dev-123456.oktapreview.com/oauth2/v1/token"
userinfo_endpoint:                        "https://dev-123456.oktapreview.com/oauth2/v1/userinfo"
registration_endpoint:                    "https://dev-123456.oktapreview.com/oauth2/v1/clients"
jwks_uri:                                 "https://dev-123456.oktapreview.com/oauth2/v1/keys"
response_types_supported:
  0:                                      "code"
  1:                                      "id_token"
  2:                                      "code id_token"
  3:                                      "code token"
  4:                                      "id_token token"
  5:                                      "code id_token token"
response_modes_supported:                 […]
grant_types_supported:                    […]
subject_types_supported:                  […]
id_token_signing_alg_values_supported:
  0:                                      "RS256"
scopes_supported:
  0:                                      "openid"
  1:                                      "email"
  2:                                      "profile"
  3:                                      "address"
  4:                                      "phone"
  5:                                      "offline_access"
  6:                                      "groups"
```

⚠ It is recommended to use the URL specific to your IdP. Your IT dept. may choose to export this JSON file of this page and provide it to you. Either option will work.

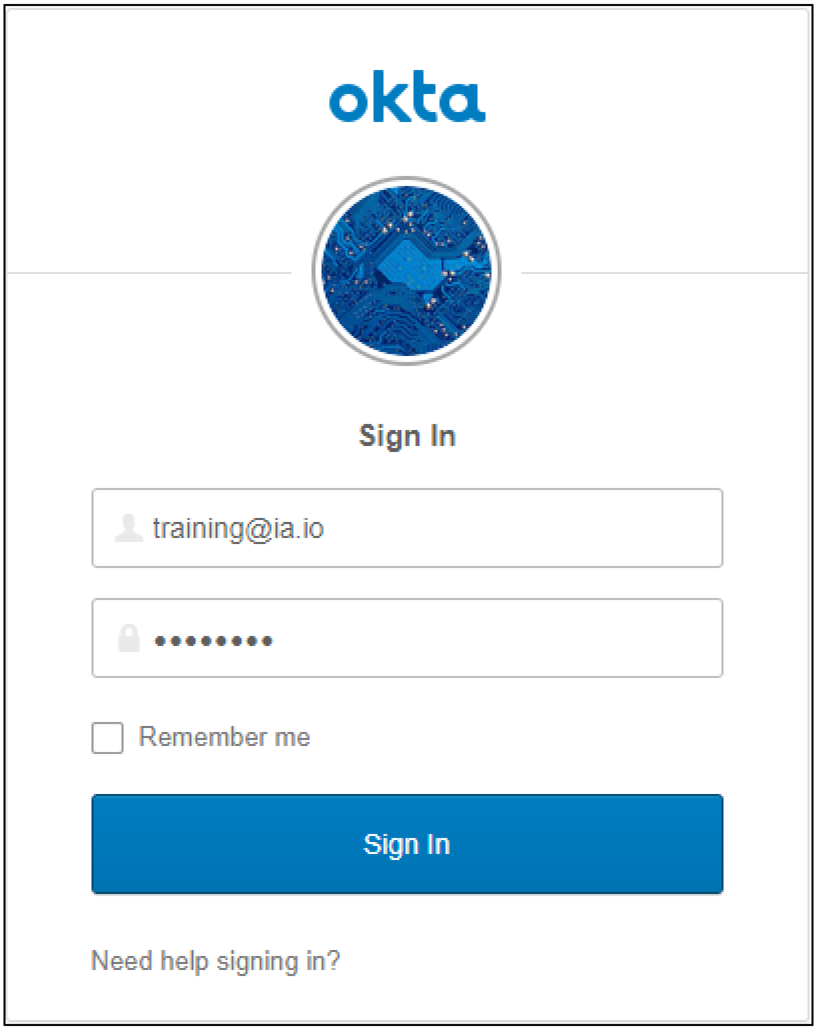
Scope Data

When a user is verified by the IdP a lot of the user specific data is not returned in the response file by default (i.e., username, email, firstname, lastname, etc). This user specific data is called the scope, and it can be returned if the **Scope** section of the Ignition Gateway is configured. The list of available scope definitions may be available from your IT deptartment or available from the developer documentation of the IdP you are using.

For this Okta example, the scope data is in the developer notes at https://developer.okta.com/docs/api/resources/oidc#scope-dependent-claims-not-always-returned

Test Login Credentials

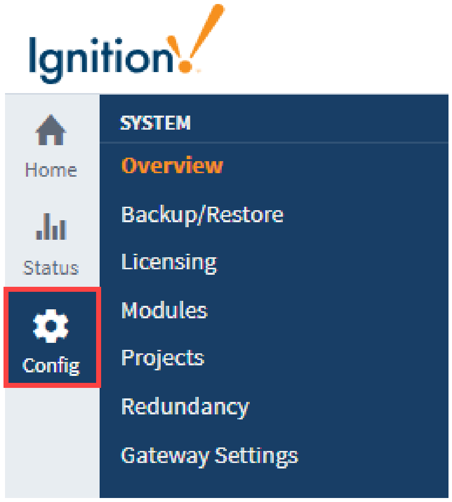
You need an account specific to the IdP for testing purposes (Okta in this example). To test and verify the IdP account, login to your IdP. For our example, the Okta login page is shown here:



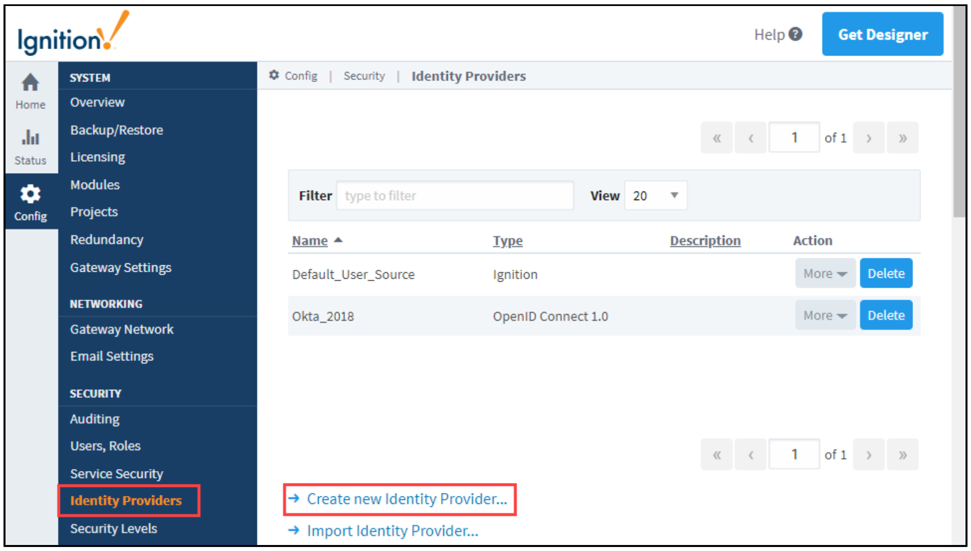
You should now have a IdP credentials to test with, a metaData URL or metaData JSON file, and a list of scope parameters to reference for your project. The next step is to configure Ignition to communicate with your IdP.

Configure Ignition Gateway

1. On the Gateway Webpage, click on the **Config** tab. You will need to log in if you aren't already.



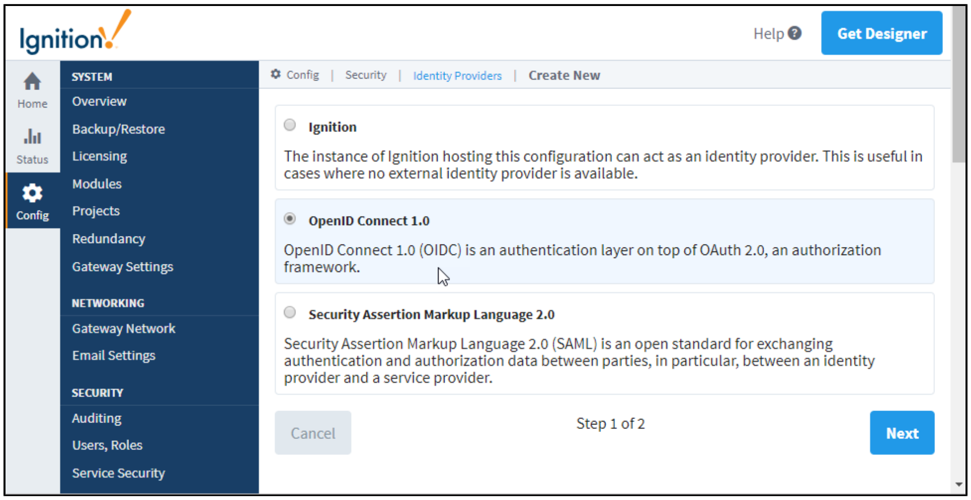
2. Under the Security section, click on **Identity Providers**. The Identity Providers screen is displayed. This screen will list all IdPs that have been configured. You can filter by name or adjust the number of IdPs displayed per page in the view.



Filter type to filter View 20

Name ▲	Type	Description	Action
Default_User_Source	Ignition		More Delete
Okta_2018	OpenID Connect 1.0		More Delete

3. Click on the **Create a New Identity Provider...** link.
4. Select the **OpenID Connect 1.0** option and press **Next**.



○ **Ignition**

The instance of Ignition hosting this configuration can act as an identity provider. This is useful in cases where no external identity provider is available.

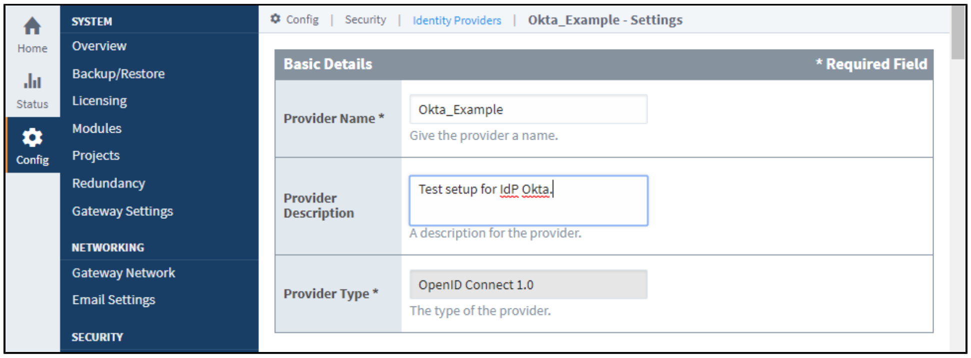
● **OpenID Connect 1.0**

OpenID Connect 1.0 (OIDC) is an authentication layer on top of OAuth 2.0, an authorization framework.

○ **Security Assertion Markup Language 2.0**

Security Assertion Markup Language 2.0 (SAML) is an open standard for exchanging authentication and authorization data between parties, in particular, between an identity provider and a service provider.

5. On the Basic Details screen, provide an **Adapter Name**. You can also add an **Adapter Description** if desired. The Provider Type field will fill in automatically from the previous screen.

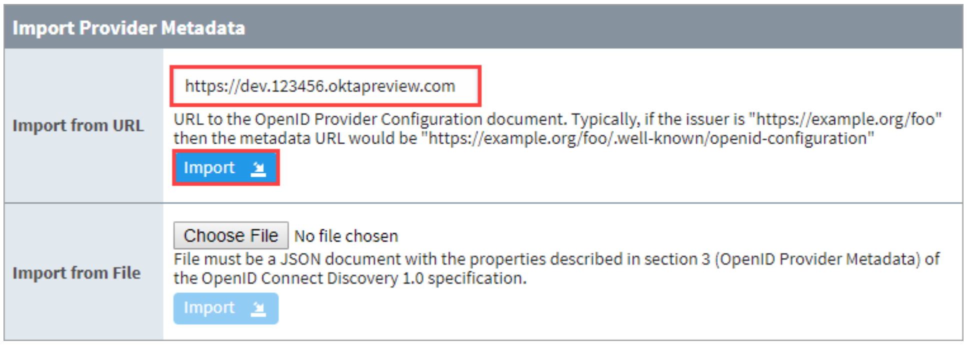


Basic Details * Required Field

Provider Name *	Okta_Example Give the provider a name.
Provider Description	Test setup for IdP Okta. A description for the provider.
Provider Type *	OpenID Connect 1.0 The type of the provider.

6. The next section is **Import Provider Metadata**. In the **Import from URL** section, enter in the URL from earlier that shows the ".well-known/openid-configuration" link specific to your IdP. You can also import a file below if it was provided by your IT department.

7. Click on the **Import** button.



Import Provider Metadata

Import from URL

https://dev.123456.oktapreview.com

URL to the OpenID Provider Configuration document. Typically, if the issuer is "https://example.org/foo" then the metadata URL would be "https://example.org/foo/.well-known/openid-configuration"

Import

Import from File

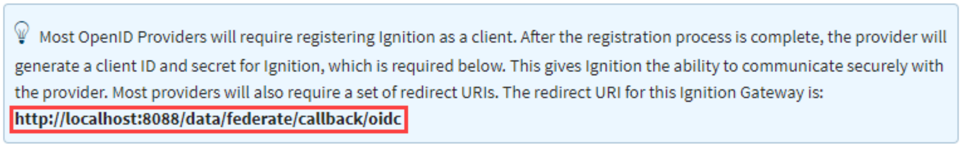
Choose File No file chosen

File must be a JSON document with the properties described in section 3 (OpenID Provider Metadata) of the OpenID Connect Discovery 1.0 specification.

Import

8. Ignition will now generate a URI redirect address for your Ignition server. It is listed just below the “Import Provider Metadata” area of the configuration page.
In our example it is http://10.10.115.3:8088/data/federate/callback/oidc. You need to provide this URI to your IdP (usually this means giving it to your IT department).

⚠ The URI should be a web address that is accessible from the end user's web browser.

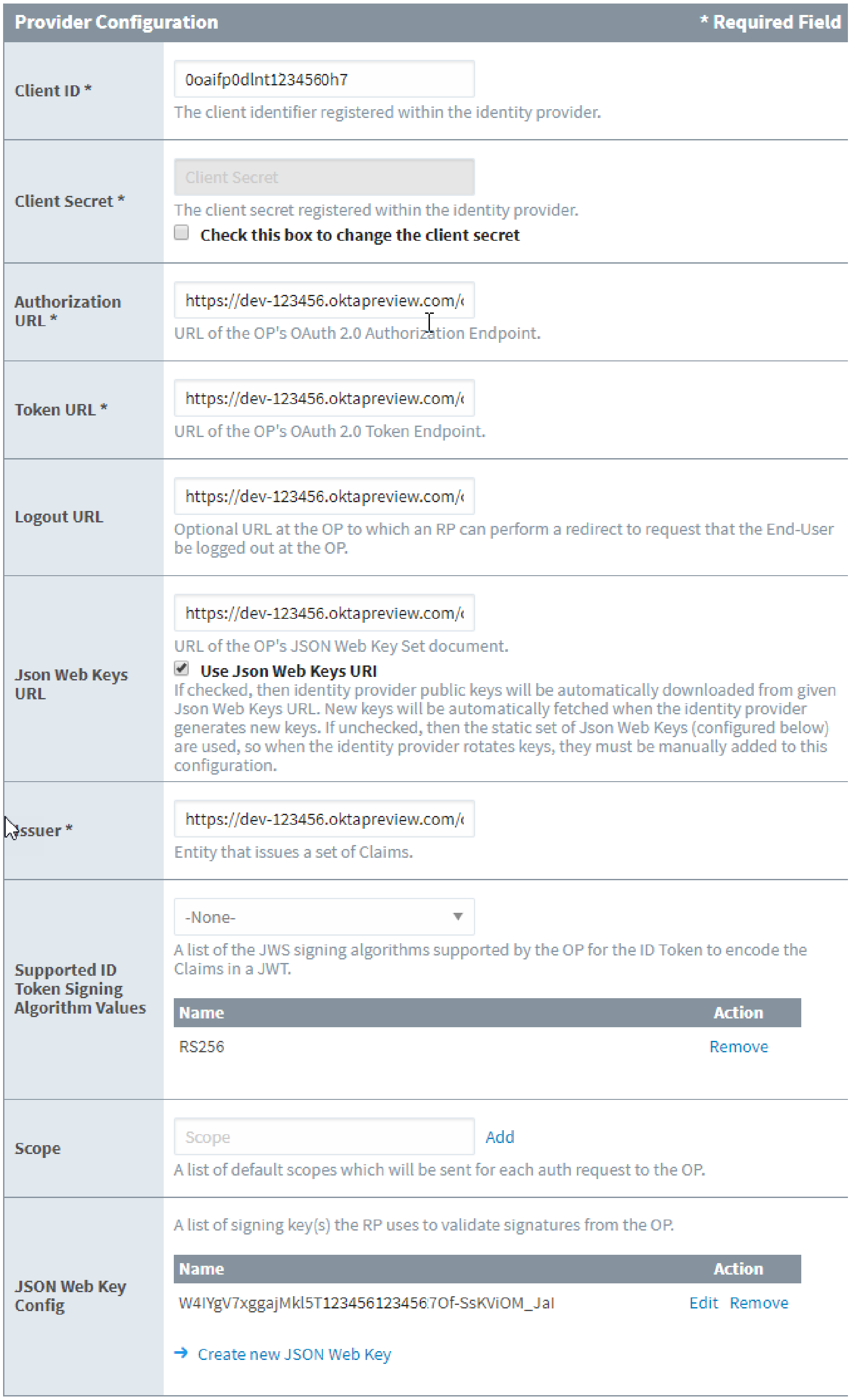


Most OpenID Providers will require registering Ignition as a client. After the registration process is complete, the provider will generate a client ID and secret for Ignition, which is required below. This gives Ignition the ability to communicate securely with the provider. Most providers will also require a set of redirect URIs. The redirect URI for this Ignition Gateway is:
http://localhost:8088/data/federate/callback/oidc

9. Once you've given your IT department the redirect address, they can add your Ignition server as an application to the IdP.
Once they have done this, they can provide you with a “Client ID” and “Client Secret”. This is needed for the Ignition Gateway to properly communicate with the IdP.

⚠ The IdP can use the same redirect address for the Login, Logout, and Initiate Login.

10. The next section is Provider Configuration. Most of the fields below should now be filled in when you imported the IdP Metadata. Fill in the **Client ID** and **Client Secret** fields with the information obtained from your IdP (or IT department). If you don't know them yet, you can put in bogus values for now and edit them later once the correct values are provided to you.



Provider Configuration *** Required Field**

Client ID *
0oaifp0dlnt1234560h7
The client identifier registered within the identity provider.

Client Secret *
Client Secret
The client secret registered within the identity provider.
□ **Check this box to change the client secret**

Authorization URL *
https://dev-123456.oktapreview.com/
URL of the OP's OAuth 2.0 Authorization Endpoint.

Token URL *
https://dev-123456.oktapreview.com/
URL of the OP's OAuth 2.0 Token Endpoint.

Logout URL
https://dev-123456.oktapreview.com/
Optional URL at the OP to which an RP can perform a redirect to request that the End-User be logged out at the OP.

Json Web Keys URL
https://dev-123456.oktapreview.com/
URL of the OP's JSON Web Key Set document.
☑ **Use Json Web Keys URI**
If checked, then identity provider public keys will be automatically downloaded from given Json Web Keys URL. New keys will be automatically fetched when the identity provider generates new keys. If unchecked, then the static set of Json Web Keys (configured below) are used, so when the identity provider rotates keys, they must be manually added to this configuration.

Issuer *
https://dev-123456.oktapreview.com/
Entity that issues a set of Claims.

Supported ID Token Signing Algorithm Values
-None-
A list of the JWS signing algorithms supported by the OP for the ID Token to encode the Claims in a JWT.

Name	Action
RS256	Remove

Scope
Scope Add
A list of default scopes which will be sent for each auth request to the OP.

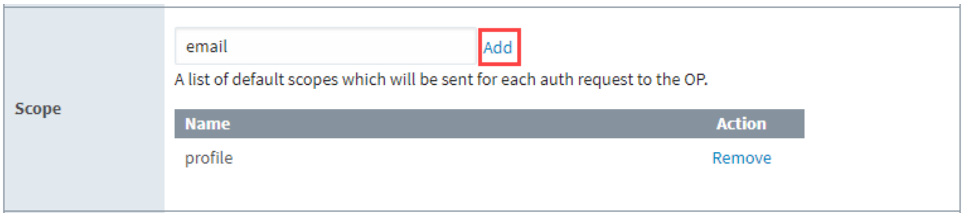
JSON Web Key Config
A list of signing key(s) the RP uses to validate signatures from the OP.

Name	Action
W4IYgV7xggajMkl5T1234561234567Of-SsKViOM_JaI	Edit Remove

→ Create new JSON Web Key

11. Providing scope is optional. These fields are specific to your IdP, and you may need to find the developer documentation specific to your IdP.

    Enter "email" in the **Scope** field and press the **Add** button. Repeat for each scope you want returned.
    For our example, the list of available scopes is in the Okta developer documentation: https://developer.okta.com/docs/api/resources/oidc#scope-dependent-claims-not-always-returned.



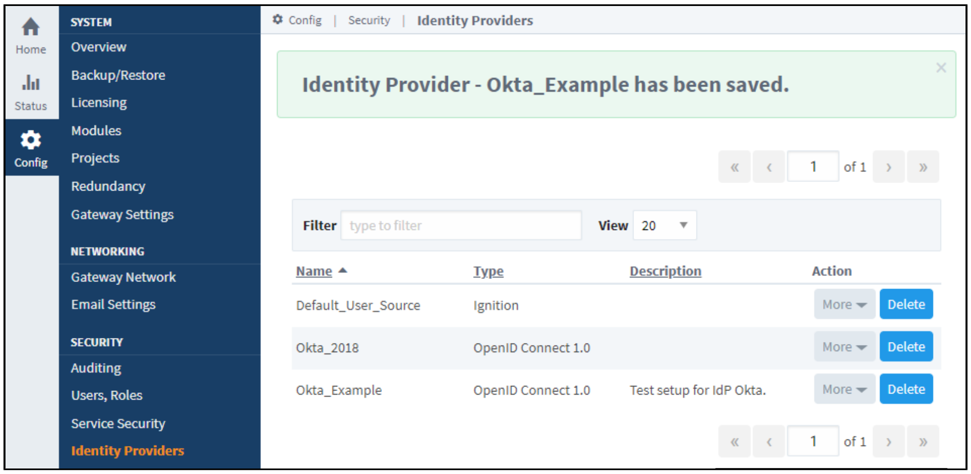
Scope

email

A list of default scopes which will be sent for each auth request to the OP.

Name	Action
profile	Remove

12. Press the **Save** button at the bottom right of the page. You'll see a confirmation message.

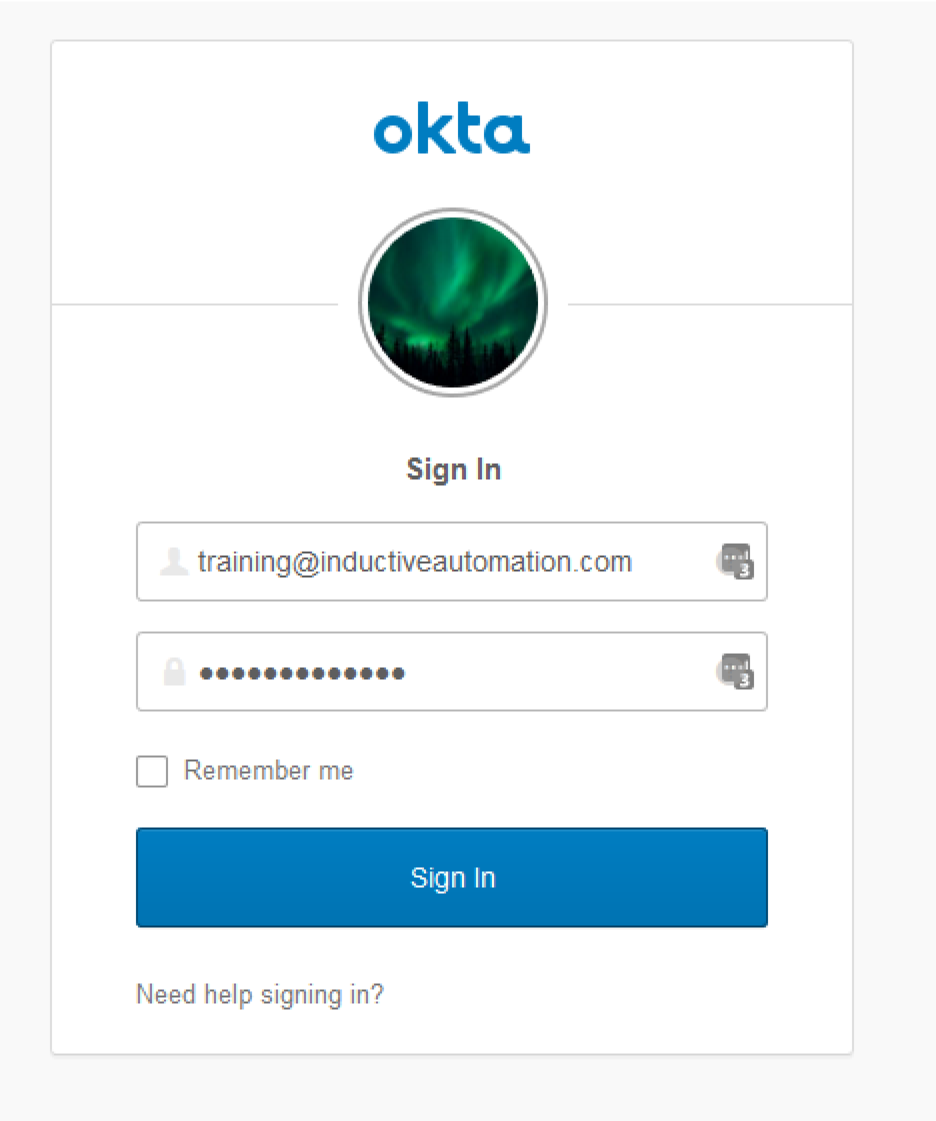


Identity Provider - Okta_Example has been saved.

Name	Type	Description	Action
Default_User_Source	Ignition		More Delete
Okta_2018	OpenID Connect 1.0		More Delete
Okta_Example	OpenID Connect 1.0	Test setup for IdP Okta.	More Delete

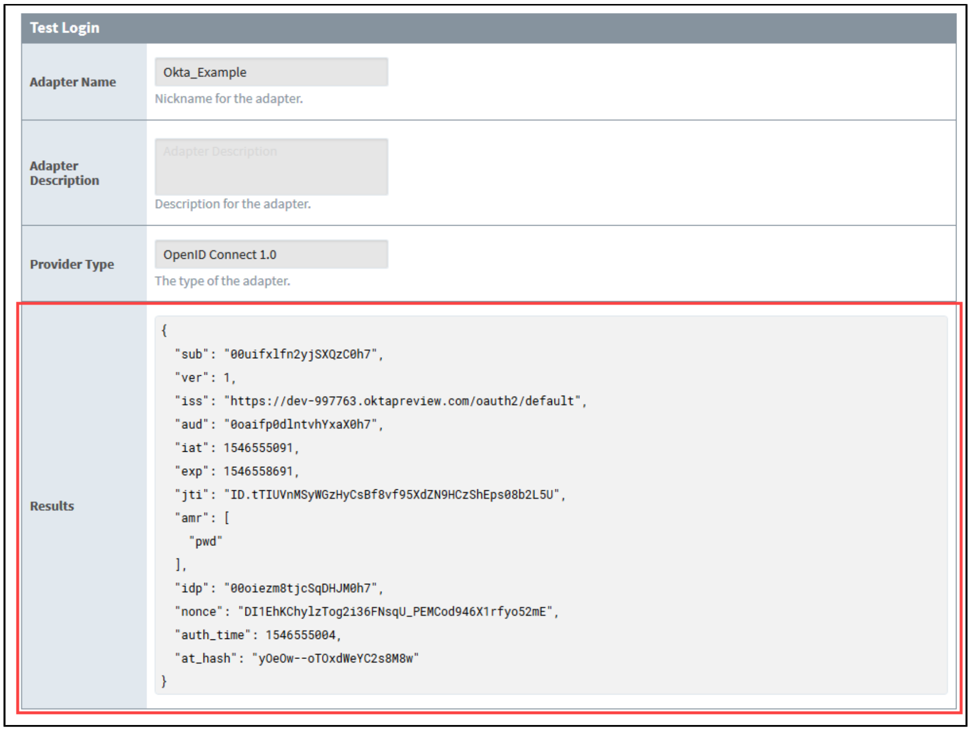
13. The next step is to perform a test login. From the Identity Providers screen, select **More** and then **Test Login.**

14. You will be re-directed to the Okta login. Enter in your test login credentials and click the **Sign In** button.



15. If the login is successful, you will be returned to the Identity Provider Test Login screen. The returned results will be displayed in the Results section.

> ⚠ In this example, we did not add the username or email to our scope. Thus they have not been returned.



Test Login

Adapter Name	Okta_Example Nickname for the adapter.
Adapter Description	Adapter Description Description for the adapter.
Provider Type	OpenID Connect 1.0 The type of the adapter.

Results

```
{
  "sub": "00uifxlfn2yjSXQzC0h7",
  "ver": 1,
  "iss": "https://dev-997763.oktapreview.com/oauth2/default",
  "aud": "0oaifp0dlntvhYxaX0h7",
  "iat": 1546555091,
  "exp": 1546558691,
  "jti": "ID.tTIUVnMSyWGzHyCsBf8vf95XdZN9HCzShEps08b2L5U",
  "amr": [
    "pwd"
  ],
  "idp": "00oiezm8tjcSqDHJM0h7",
  "nonce": "DI1EhKChylzTog2i36FNsqU_PEMCod946X1rfyo52mE",
  "auth_time": 1546555004,
  "at_hash": "yOeOw--oTOxdWeYC2s8M8w"
}
```

Editor notes are only visible to logged in users

Admin Username/password for Training's OKTA instance are available through LastPass. See Bobby.

OKTA USER LOGIN INFO:

Okta Login Page: https://dev-997763.oktapreview.com/login/login.htm

Okta ia User: training@ia.io

Okta ia Pass: P4ssw0rd

Okta Metadata Definition: https://dev-997763.oktapreview.com/.well-known/openid-configuration

Okta Scope Definition: https://developer.okta.com/docs/api/resources/oidc#scope-dependent-claims-not-always-returned

Okta Client ID: 0oaifp0dlntvhYxaX0h7

Okta Client Secret: ZvMy_gt-XqiILB4qQH6AacpsEqA_M2Y8gwkvoj3Q

SAML Example

This section provides an example of how to connect an Identity Provider that is using the SAML protocol. This example uses the Okta IdP service. Your IdP vendor may differ and the specific links will differ.

On this page ...

Prerequisites

Before you begin configuring Ignition there are some preliminary requirements that need to be done outside of Ignition:

- A configured remote IdP (Okta in this example)
- The metadata file specific to your IdP
- The scope data specific to your IdP
- Login credentials to use as a test

Configured IdP

An IT department is usually the one to setup and configure a remote IdP. You need a configured remote IdP that is compatible with SAML protocol.

At minimum there needs to be an account set up with the IdP, users added to the IdP account, and applications added to the IdP.

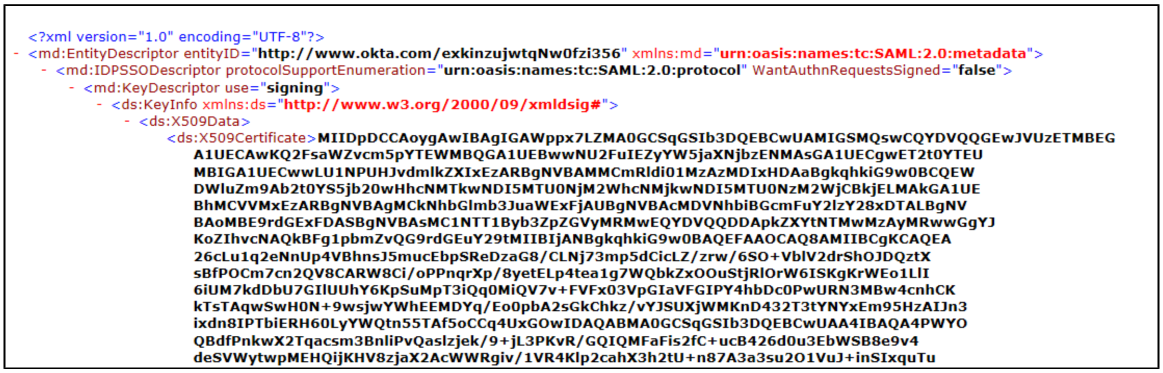
Metadata File

You will need the metadata file specific to your IdP. This document defines how to communicate with the IdP. It is usually a web page that allows the metadata file to be exported to an XML file.

You will need the URL link to this page or an XML export of this page. For example, the metadata import URL may look like something like this:

```
https://dev-123456.okta.com/app/esdfsdf7886sd6723hjkdf/sso/saml/metadata
```

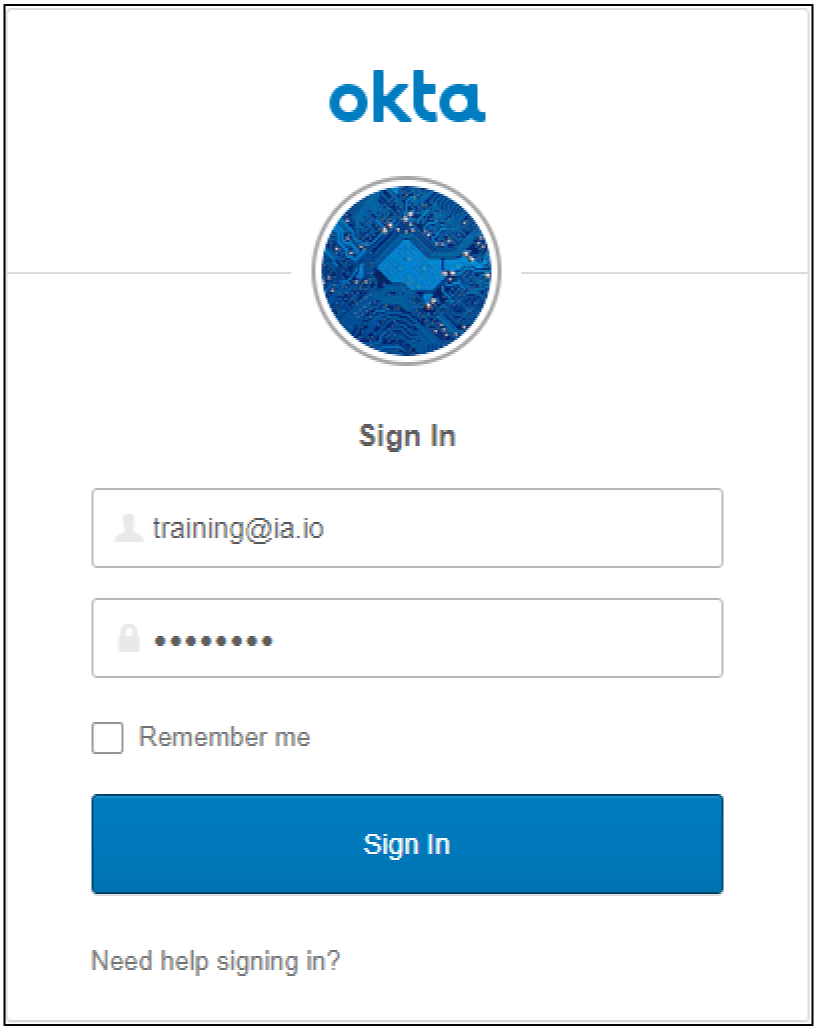
Here is an example of part of a metadata file for Okta. Notice that file is XML format. You can use the file or the URL to automatically import the configuration into Ignition. Otherwise it will need to be manually typed in.



```
  <?xml version="1.0" encoding="UTF-8"?>
- <md:EntityDescriptor entityID="http://www.okta.com/exkinzujwtqNw0fzi356" xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata">
  - <md:IDPSSODescriptor protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol" WantAuthnRequestsSigned="false">
    - <md:KeyDescriptor use="signing">
      - <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        - <ds:X509Data>
            <ds:X509Certificate>MIIDpDCCAoygAwIBAgIGAWppx7LZMA0GCSqGSIb3DQEBCwUAMIGSMQswCQYDVQQGEwJVUzETMBEG
              A1UECAwKQ2FsaWZvcm5pYTEWMBQGA1UEBwwNU2FuIEZyYW5jaXNjbzENMAsGA1UECgwET2t0YTEU
              MBIGA1UECwwLU1NPUHJvdmlkZXIxEzARBgNVBAMMCmRldi01MzAzMDIxHDAaBgkqhkiG9w0BCQEW
              DWluZm9Ab2t0YS5jb20wHhcNMTkwNDI5MTU0NjM2WhcNMjkwNDI5MTU0NzM2WjCBkjELMAkGA1UE
              BhMCVVMxEzARBgNVBAgMCkNhbGlmb3JuaWExFjAUBgNVBAcMDVNhbiBGcmFuY2lzY28xDTALBgNV
              BAoMBE9rdGExFDASBgNVBAsMC1NTT1Byb3ZpZGVyMRMwEQYDVQQDDApkZXYtNTMwMzAyMRwwGgYJ
              KoZIhvcNAQkBFg1pbmZvQG9rdGEuY29tMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA
              26cLu1q2eNnUp4VBhnsJ5mucEbpSReDzaG8/CLNj73mp5dCicLZ/zrw/6SO+VblV2drShOJDQztX
              sBfPOCm7cn2QV8CARW8Ci/oPPnqrXp/8yetELp4tea1g7WQbkZxOOuStjRlOrW6ISKgKrWEo1LlI
              6iUM7kdDbU7GIlUUhY6KpSuMpT3iQq0MiQV7v+FVFx03VpGIaVFGIPY4hbDc0PwURN3MBw4cnhCK
              kTsTAqwSwH0N+9wsjwYWhEEMDYq/Eo0pbA2sGkChkz/vYJSUXjWMKnD432T3tYNYxEm95HzAIJn3
              ixdn8IPTbiERH60LyYWQtn55TAf5oCCq4UxGOwIDAQABMA0GCSqGSIb3DQEBCwUAA4IBAQA4PWYO
              QBdfPnkwX2Tqacsm3BnliPvQaslzjek/9+jL3PKvR/GQIQMFaFis2fC+ucB426d0u3EbWSB8e9v4
              deSVWytwpMEHQijKHV8zjaX2AcWWRgiv/1VR4Klp2cahX3h2tU+n87A3a3su2O1VuJ+inSIxquTu
```

Test Login Credentials

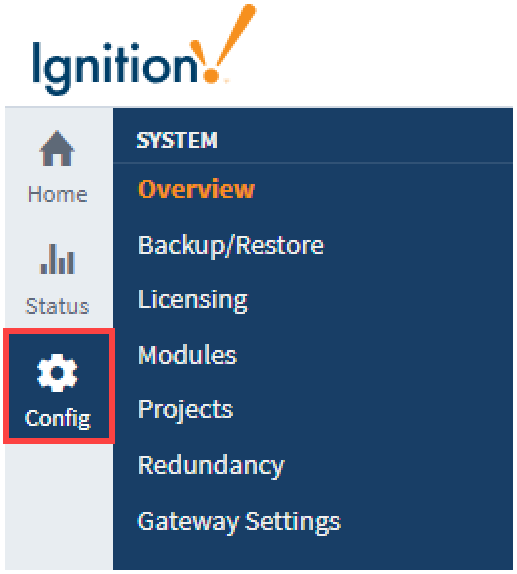
You need an account specific to the IdP for testing purposes (Okta in this example). To test and verify the IdP account, login to your IdP. For our example, the Okta login page is shown here:



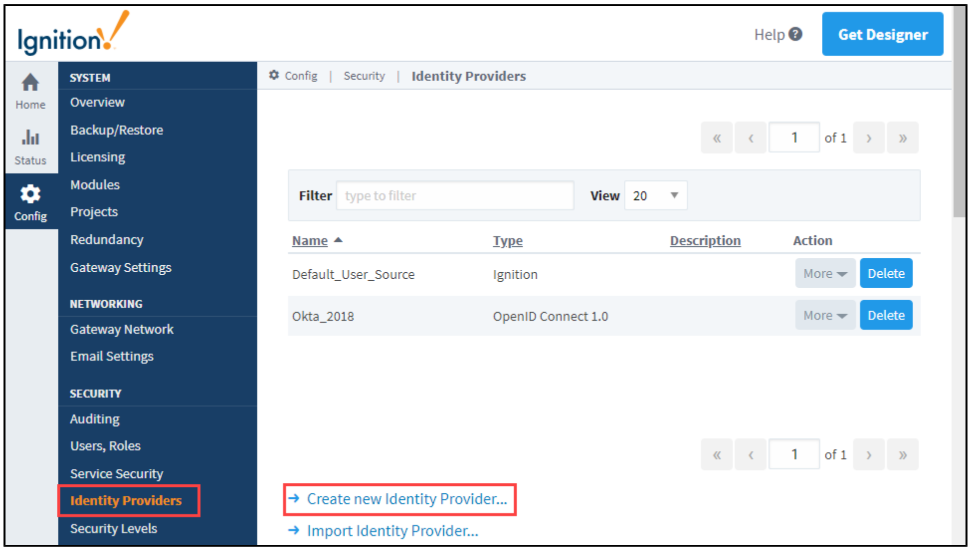
You should now have a IdP credentials to test with, a metadata URL or metadata XML file. The next step is to configure Ignition to communicate with your IdP.

Configure Ignition Gateway

1. On the Gateway Webpage, click on the **Config** tab. You will need to log in if you aren't already.



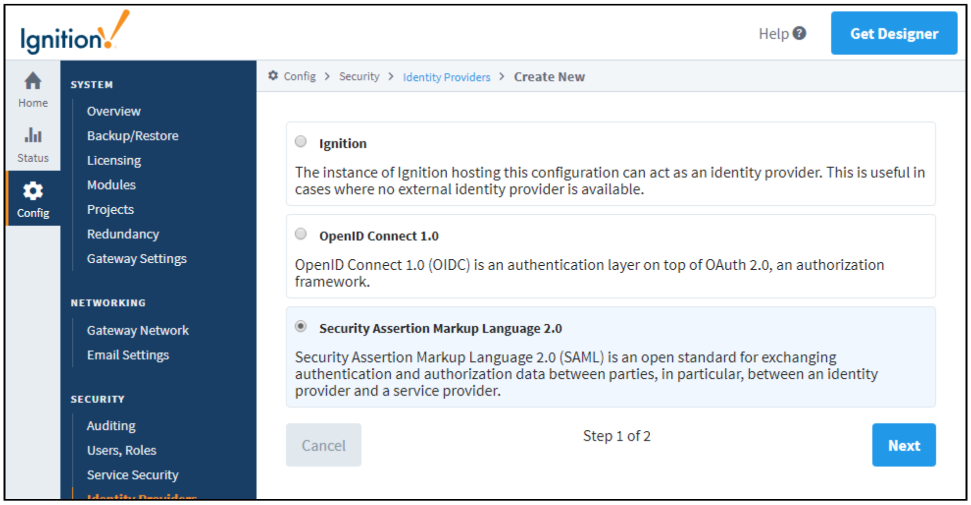
2. Under the Security section, click on **Identity Providers**. The Identity Providers screen is displayed. This screen will list all IdPs that have been configured. You can filter by name or adjust the number of IdPs displayed per page in the view.



Name	Type	Description	Action
Default_User_Source	Ignition		More Delete
Okta_2018	OpenID Connect 1.0		More Delete

1 of 1
→ Create new Identity Provider...
→ Import Identity Provider...

3. Click on the **Create a New Identity Provider...** link.
4. Select the **Security Assertion Markup Language 2.0** option and press **Next**.



○ **Ignition**

The instance of Ignition hosting this configuration can act as an identity provider. This is useful in cases where no external identity provider is available.

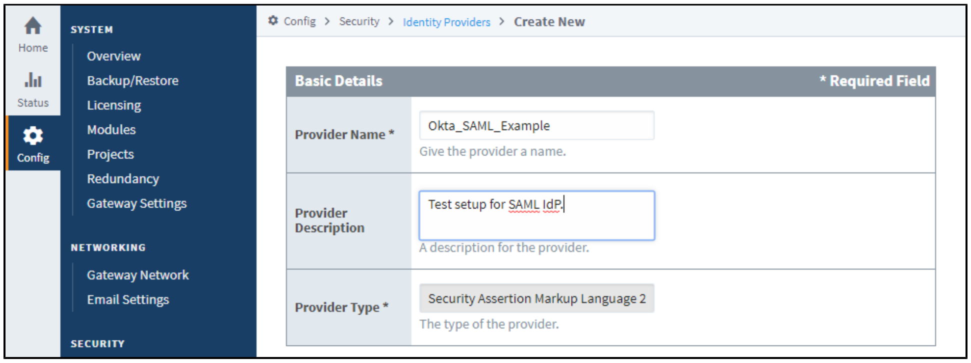
○ **OpenID Connect 1.0**

OpenID Connect 1.0 (OIDC) is an authentication layer on top of OAuth 2.0, an authorization framework.

● **Security Assertion Markup Language 2.0**

Security Assertion Markup Language 2.0 (SAML) is an open standard for exchanging authentication and authorization data between parties, in particular, between an identity provider and a service provider.

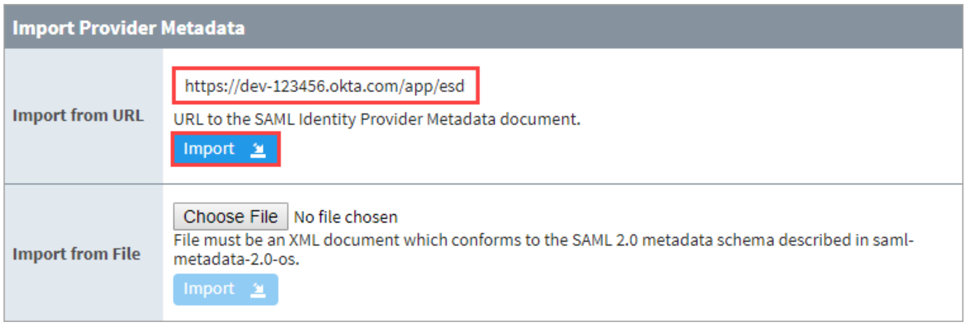
5. On the Basic Details screen, provide an **Adapter Name**. You can also add an **Adapter Description** if desired. The Provider Type field will fill in automatically from the previous screen.



Basic Details * Required Field

Provider Name *	Okta_SAML_Example Give the provider a name.
Provider Description	Test setup for SAML IdP. A description for the provider.
Provider Type *	Security Assertion Markup Language 2 The type of the provider.

6. The next section is **Import Provider Metadata**. In the **Import from URL** section, enter in the URL from earlier specific to your IdP. You can also import a file below if it was provided by your IT department.
7. Click on the **Import** button.

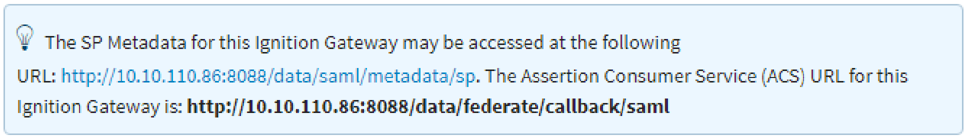


Import Provider Metadata

Import from URL	https://dev-123456.okta.com/app/esd URL to the SAML Identity Provider Metadata document.
Import from File	No file chosen File must be an XML document which conforms to the SAML 2.0 metadata schema described in saml-metadata-2.0-os.

8. Ignition will now generate a URI redirect address for your Ignition server. It is listed just below the "Import Provider Metadata" area of the configuration page.
In our example it is http://10.10.110.86:8088/data/federate/callback/saml. You need to provide this URI to your IdP (usually this means giving it to your IT department).

> ⚠ The URI should be a web address that is accessible from the IdP server.

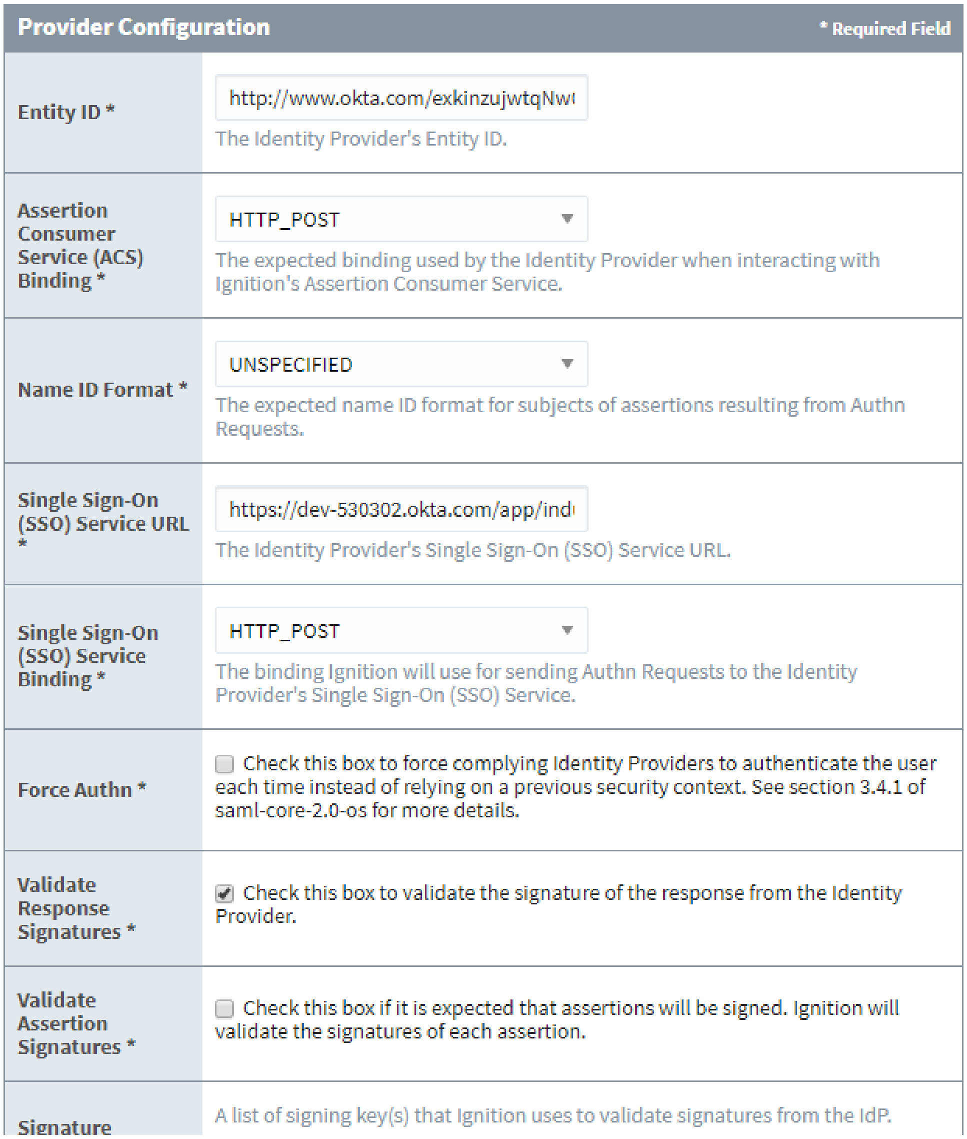


The SP Metadata for this Ignition Gateway may be accessed at the following URL: http://10.10.110.86:8088/data/saml/metadata/sp. The Assertion Consumer Service (ACS) URL for this Ignition Gateway is: **http://10.10.110.86:8088/data/federate/callback/saml**

9. Once you have given your IT department the redirect address, they can add your Ignition server as an application to the IdP.

> ⚠ The IdP can use the same redirect address for the Login, Logout, and Initiate Login.

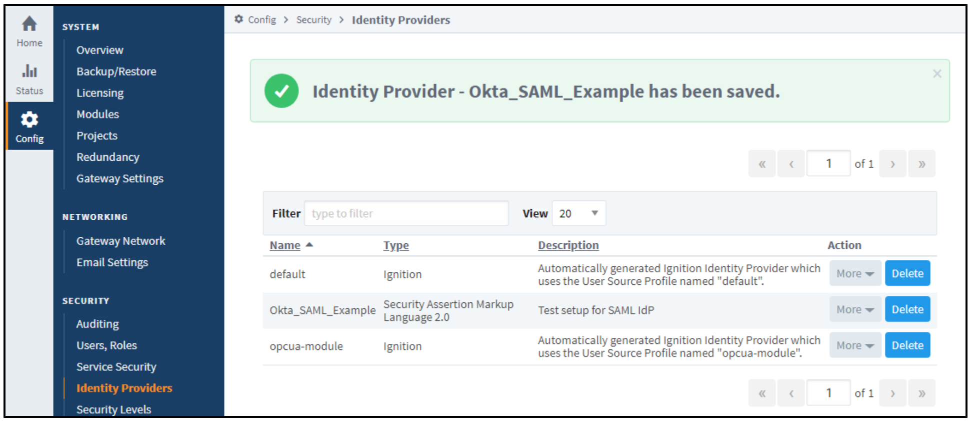
10. The next section is Provider Configuration. Most of the fields below should now be filled in when you imported the IdP Metadata.



Provider Configuration * Required Field

Entity ID *	http://www.okta.com/exkinzujwtqNw The Identity Provider's Entity ID.
Assertion Consumer Service (ACS) Binding *	HTTP_POST The expected binding used by the Identity Provider when interacting with Ignition's Assertion Consumer Service.
Name ID Format *	UNSPECIFIED The expected name ID format for subjects of assertions resulting from Authn Requests.
Single Sign-On (SSO) Service URL *	https://dev-530302.okta.com/app/ind The Identity Provider's Single Sign-On (SSO) Service URL.
Single Sign-On (SSO) Service Binding *	HTTP_POST The binding Ignition will use for sending Authn Requests to the Identity Provider's Single Sign-On (SSO) Service.
Force Authn *	□ Check this box to force complying Identity Providers to authenticate the user each time instead of relying on a previous security context. See section 3.4.1 of saml-core-2.0-os for more details.
Validate Response Signatures *	☑ Check this box to validate the signature of the response from the Identity Provider.
Validate Assertion Signatures *	□ Check this box if it is expected that assertions will be signed. Ignition will validate the signatures of each assertion.
Signature	A list of signing key(s) that Ignition uses to validate signatures from the IdP.

11. Press the Save button at the bottom right of the page. You'll see a confirmation message.



Config > Security > Identity Providers

✓ Identity Provider - Okta_SAML_Example has been saved.

« ‹ 1 of 1 › »

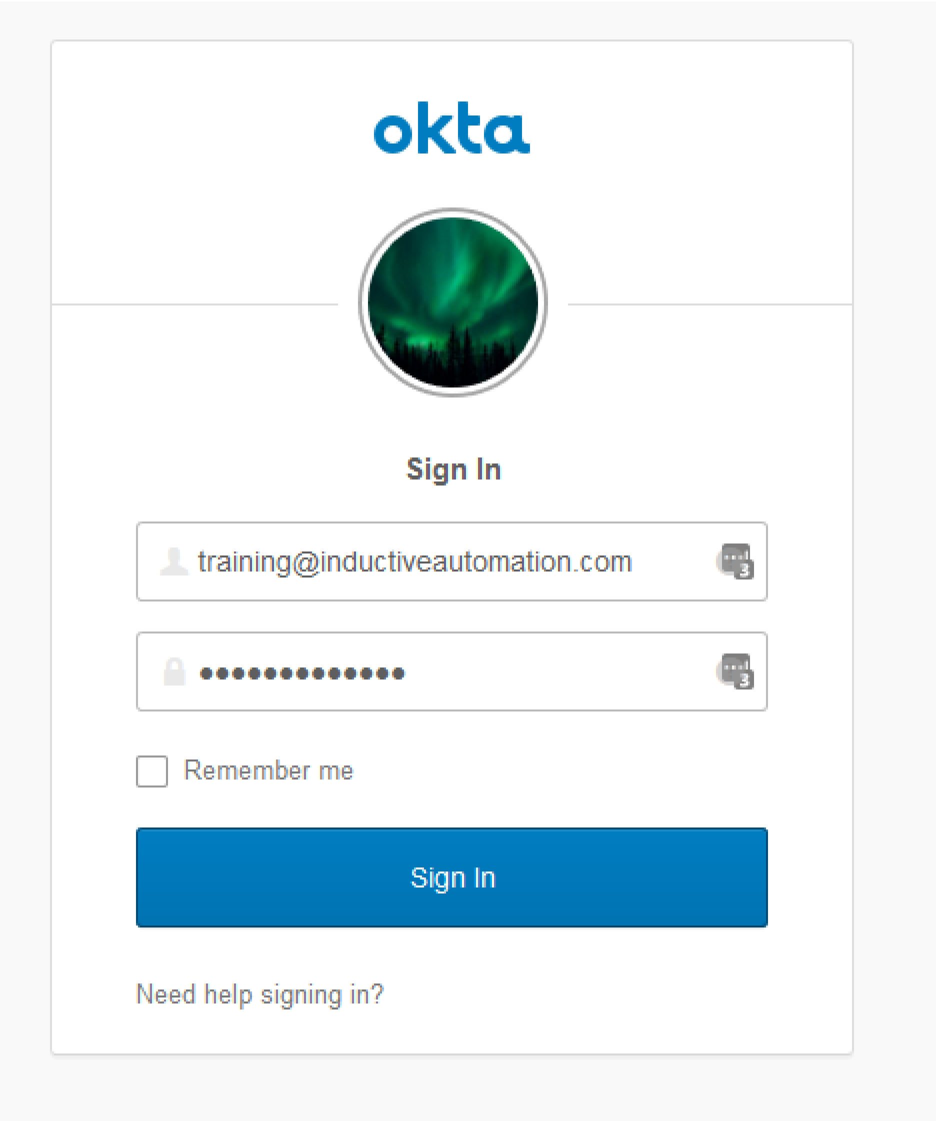
Filter type to filter View 20

Name ▲	Type	Description	Action
default	Ignition	Automatically generated Ignition Identity Provider which uses the User Source Profile named "default".	More ▾ Delete
Okta_SAML_Example	Security Assertion Markup Language 2.0	Test setup for SAML IdP	More ▾ Delete
opcua-module	Ignition	Automatically generated Ignition Identity Provider which uses the User Source Profile named "opcua-module".	More ▾ Delete

« ‹ 1 of 1 › »

12. The next step is to perform a test login. From the Identity Providers screen, select **More** and then **Test Login.**

13. You will be re-directed to the Okta login. Enter in your test login credentials and click the **Sign In** button.



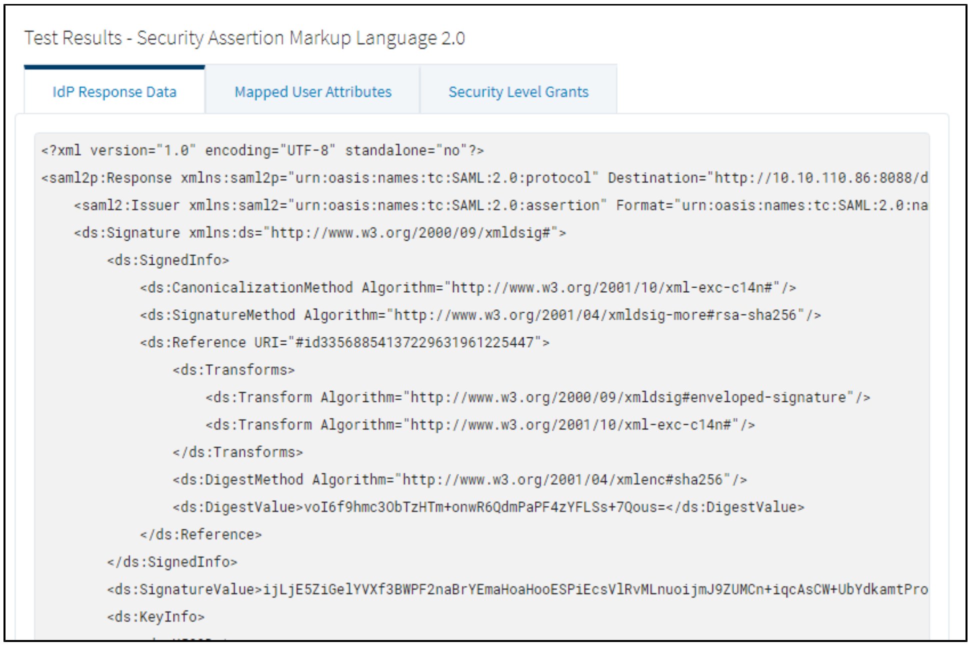
Sign In

training@inductiveautomation.com

••••••••••••••

□ Remember me

14. If the login is successful, you will be returned to the Identity Provider Test Login screen. The returned results will be displayed in the Results section.



```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<saml2p:Response xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol" Destination="http://10.10.110.86:8088/d
    <saml2:Issuer xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion" Format="urn:oasis:names:tc:SAML:2.0:na
    <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        <ds:SignedInfo>
            <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
            <ds:SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256"/>
            <ds:Reference URI="#id3356885413722963196l225447">
                <ds:Transforms>
                    <ds:Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
                    <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
                </ds:Transforms>
                <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
                <ds:DigestValue>voI6f9hmc3ObTzHTm+onwR6QdmPaPF4zYFLSs+7Qous=</ds:DigestValue>
            </ds:Reference>
        </ds:SignedInfo>
        <ds:SignatureValue>ijLjE5ZiGelYVXf3BWPF2naBrYEmaHoaHooESPiEcsVlRvMLnuoijmJ9ZUMCn+iqcAsCW+UbYdkamtPro
        <ds:KeyInfo>
```

Troubleshooting Identity Providers

Editor notes are only visible to logged in users
QA items:

Logout URL seems to have no effect

JSON Web Key seems to have no effect

This Troubleshooting section has a compilation of examples to help you diagnose and troubleshoot issues with configuring IdPs.

The Save Button Is not Selectable

All required fields must be entered on the Identity Providers screen before Save can be selected. Required fields have an asterisk (*) next to their name.

⚠ If you are waiting for values for the **ClientID** and **Client Secret** fields, you can enter fake values and return when you have the correct value.

After Importing Metadata from a File, Values Did not Auto Populate

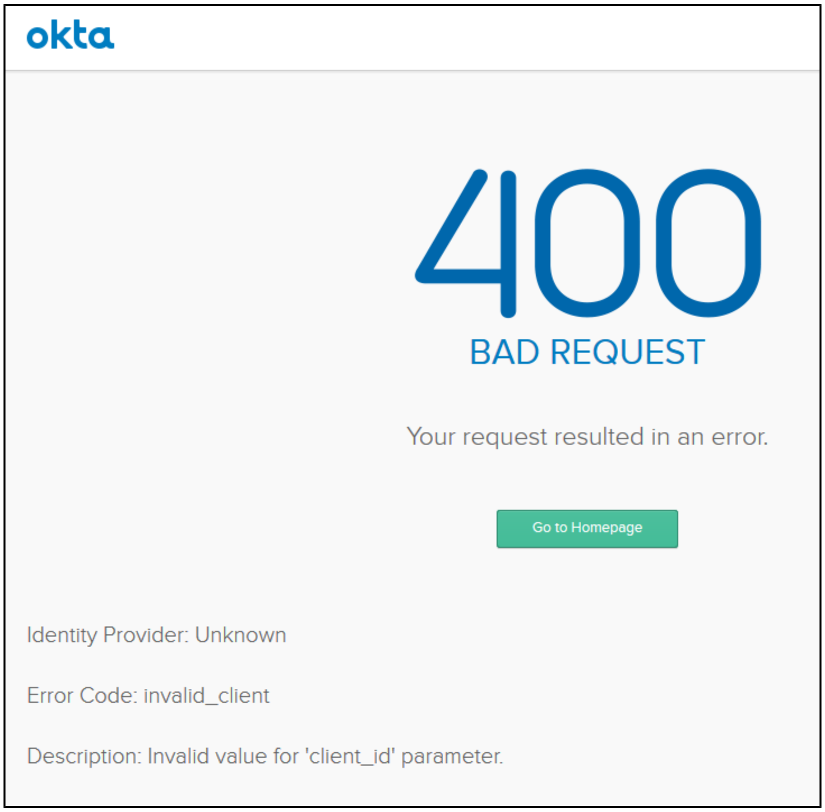
1. Verify that import file is JSON format.
2. Verify that the import URL is valid.
3. Re-import the file.

Login Testing: IdP Login Page Does not Appear

1. Confirm that the values in **Client ID** and **Client Secret** fields are correct. These come from the IdP when your Ignition Gateway is added as an application.
2. Check all other configuration settings.

Login Testing: The IdP login Is Displayed but the Login Attempt Fails

This issue is outside of Ignition. Check with your IT department and verify the login credentials (username and password) for your IdP.



400

BAD REQUEST

Your request resulted in an error.

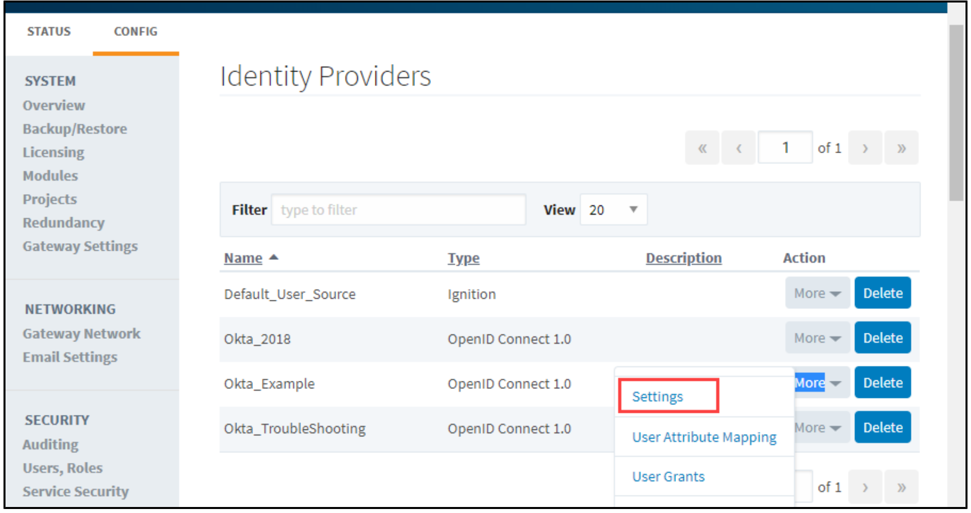
Identity Provider: Unknown

Error Code: invalid_client

Description: Invalid value for 'client_id' parameter.

Login Testing: The IdP Login Accepts the User but the IDP Redirect Fails (HTTP ERROR 500)

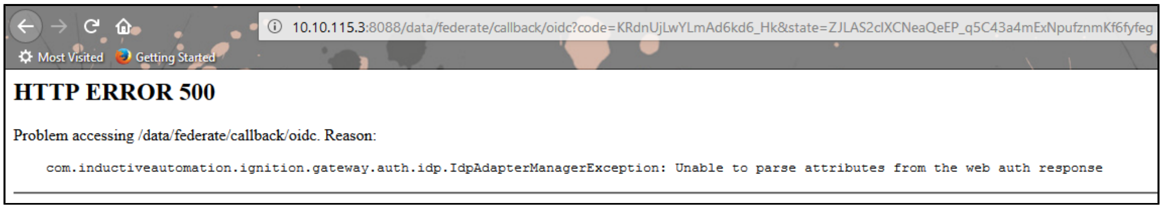
1. Go to Settings for you IdP.



Identity Providers

Name ▲	Type	Description	Action
Default_User_Source	Ignition		More Delete
Okta_2018	OpenID Connect 1.0		More Delete
Okta_Example	OpenID Connect 1.0		More Delete
Okta_TroubleShooting	OpenID Connect 1.0		More Delete

2. Verify the setting for **Supported ID Token Signing Algorithm Values**.
3. If the URL for the IdP's metadata is available, try re-importing it.
4. Verify and re-enter the Client Id and Client Secret.
5. Verify the Token URL. Then re-import the IdP's metadata.
6. Verify the JSON web keys URL. (Default is to leave the check box checked.) Then re-import the IdP's metadata.
7. Verify the Issuer URL, then re-import the IdP's metadata.



10.10.115.3:8088/data/federate/callback/oidc?code=KRdnUjLwYLmAd6kd6_Hk&state=ZJLAS2clXCNeaQeEP_q5C43a4mExNpufznmKf6fyfeg

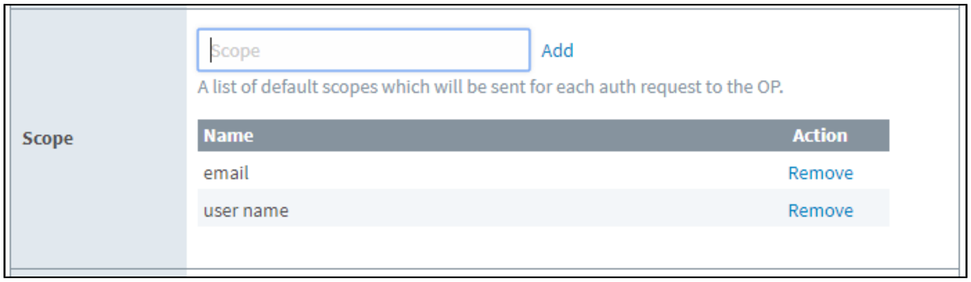
HTTP ERROR 500

Problem accessing /data/federate/callback/oidc. Reason:

```
    com.inductiveautomation.ignition.gateway.auth.idp.IdpAdapterManagerException: Unable to parse attributes from the web auth response
```

Login Testing: The Test Is Successful, but Results Do not Show Useful data (i.e., user name, eMail)

1. Go to settings for you IdP.
2. Add the desired fields to the Scope section. You may have to reference the developers documentation Scope document



Scope

A list of default scopes which will be sent for each auth request to the OP.

Name	Action
email	Remove
user name	Remove

3. Click Save.
4. Repeat the Login test.

Login Testing: Revised User Attributes Are not Shown in the Results of a Successful Test

1. Go to settings for you IdP.
2. Add the desired fields to the Scope section.
3. Click Save.
4. Repeat the Login test.

You Are not Re-directed Back to Ignition after a Successful IdP Login

1. Verify the Adapter Configuration: Authorization URL.
2. Re-import the IdP's metadata.

Designer

The Ignition Designer is where the majority of configuration and design work is done. The Designer uses web-launch technology to open and edit your projects. This is how you can configure your Perspective and Vision projects.

The Designer provides all the firepower to bring your projects to life. It uses a drag-and-drop configuration making screen development quick and intuitive. You can create user interfaces by dragging Perspective components onto a view, Vision components onto a window, and Tags onto your components to instantly bind data to tables, charts, and graphs. You can set up Tags and Transaction Groups to log data to your databases. You can set up Reports to generate and save data however you'd like. The Designer saves all your projects to the Gateway so everything is controlled in one place.

This page provides some good information about the Designer, Designer workspace, Tools, and how to create a project.

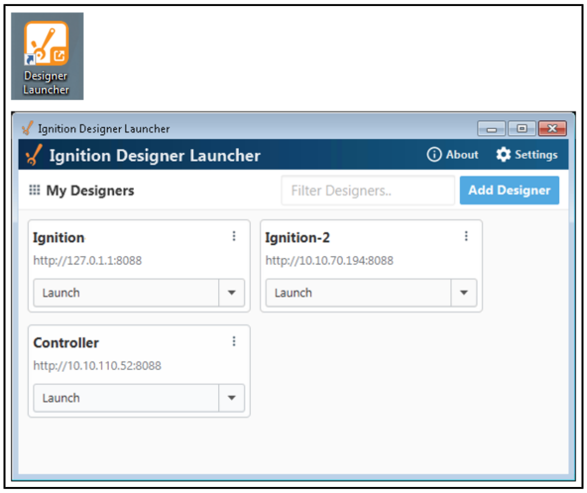
On this page ...

The Designer Launcher

Watch the Video

Opening the Designer

Opening the Designer requires running the Designer Launcher. Once the Designer Launcher is installed and configured, you'll have all your projects at your fingertips. If you created a desktop shortcut for the Designer Launcher at install, simply click to open it, and select a Designer to open, and from there you can create a new project or open an existing project.

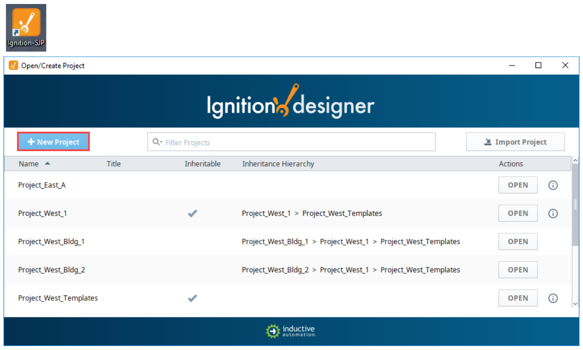


Ignition
http://127.0.0.1:8088
Launch

Ignition-2
http://10.10.70.194:8088
Launch

Controller
http://10.10.110.52:8088
Launch

If you have a Designer that has projects that you work on regularly, you can create a shortcut to that Designer and keep it on your desktop. When you click on your shortcut, it opens the Designer, then click on the project you want to edit. Now you can hit the ground running designing your project!



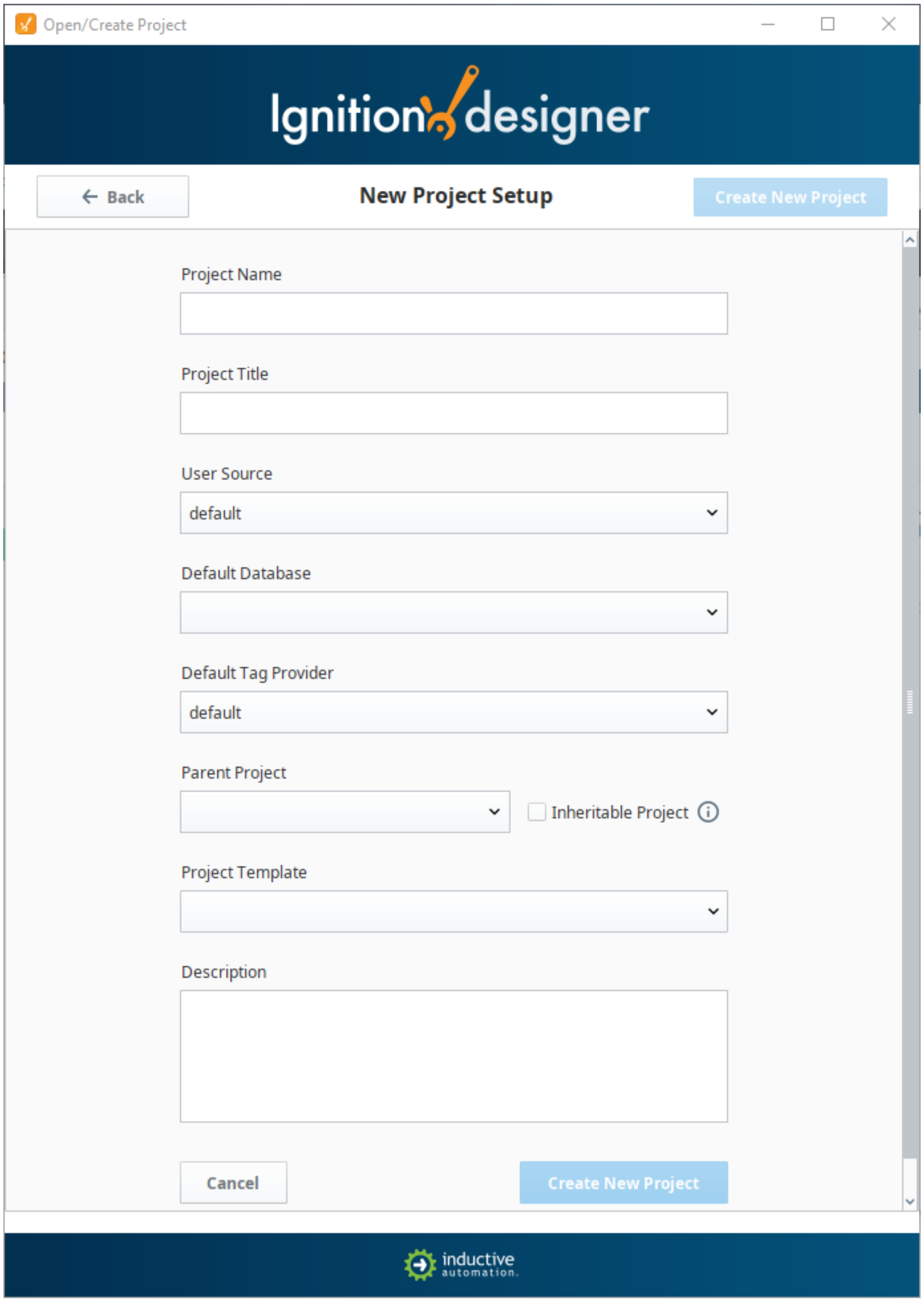
Name	Title	Inheritable	Inheritance Hierarchy	Actions
Project_East_A				OPEN ⓘ
Project_West_1		✓	Project_West_1 > Project_West_Templates	OPEN ⓘ
Project_West_Bldg_1			Project_West_Bldg_1 > Project_West_1 > Project_West_Templates	OPEN
Project_West_Bldg_2			Project_West_Bldg_2 > Project_West_1 > Project_West_Templates	OPEN
Project_West_Templates		✓		OPEN ⓘ

Refer to the Designer Launcher page for downloading, installing, and configuring the Designer Launcher.

Creating a Project

The first step in working with a project is creating one! Once you launch the Designer, the **Open/Create Project** window is displayed. Here you have the option to create a new project or open an existing project. Let's create a new project!

1. After opening the Designer, click on **Create a New Project** button. The New Project Setup window is displayed.



Project Name

Project Title

User Source: default

Default Database

Default Tag Provider: default

Parent Project

□ Inheritable Project

Project Template

Description

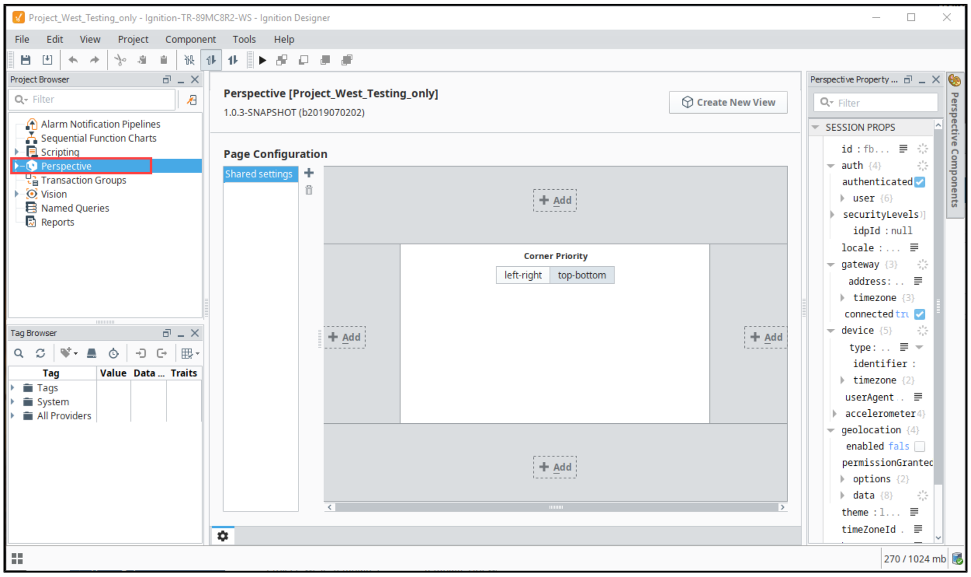
2. Enter the **Project Name** (required) and any other configuration settings you need for your project. Most settings are optional. Refer to the New Project Setup Properties Table below for a description of each property.

> ⚠ It is not advisable to change the Project Name after it's been created, instead, change the Title property in **Config > Projects** if you want to change how the project appears later.

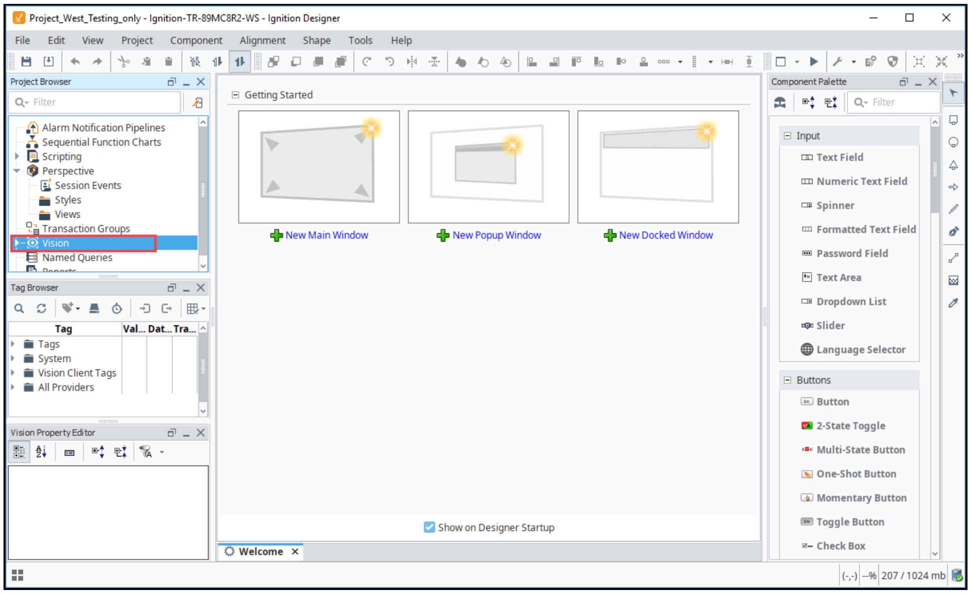
New Project Setup Properties

Property	Description
Project Name	The Project Name can only consist of alphanumeric characters and the '_' (underscore) character. Spaces and other special characters are not supported. Note: It is not advisable to change the Project Name after it's been created, instead, change the Title property in **Config > Projects** if you want to change how the project appears later.
Project Title	This is the name that will be displayed on the launch page of the Gateway and in the runtime Client or Session, (optional) There are no restrictions on using special characters or spaces. If you do not specify a title, the project name will be displayed on the Gateway launch page and in the runtime.
User Source	Controls access to the project. If this field is left blank, the **'default'** user source is set. For more information or to add users, see Security.
Default Database	Select a Default Database (optional). Any queries to the database will use this default database connection unless explicitly specified otherwise.
Default Tag Provider	Select a default Tag Provider (optional). Most installations will likely only have one provider, but in situations where there are more than one, this is the provider that will be used unless explicitly specified otherwise.
Parent Project	Each project may have a parent project, and will inherit all of the resources of that parent project, (optional). Click the dropdown list to see all the available options.
Project Template	Select a Project Template (optional). There are several pre-built project templates focused on navigation that support either Perspective or Vision: Perspective Menu Nav, Perspective Web Nav, Vision 2-Tier Tab Nav, Vision Tab Nav, and Vision Tree Nav. Click the dropdown to see all the available options.
Description	Enter a description of the project (optional). Note that once a project is created, this description can be viewed on the Open/Create Project screen when you hover over the Information ⓘ icon.

3. Click on **Create New Project**.
4. Your project is now created and opens on the **Perspective Configuration Page**. If you want to design your project in Vision, click on the **Vision** application in the **Project Browser** and the **Vision Getting Started window** will open.

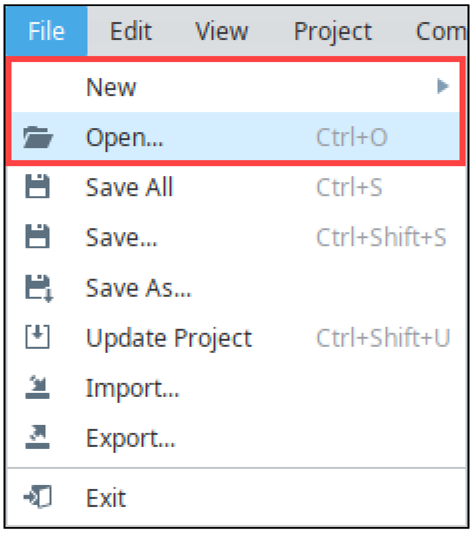


To design your project in Vision, expand **Vision** in the **Project Browser** to create windows and add components and Tags. For more information on getting started, refer to the Quick Start - Vision Client page.



Open or Create a Project from within the Designer

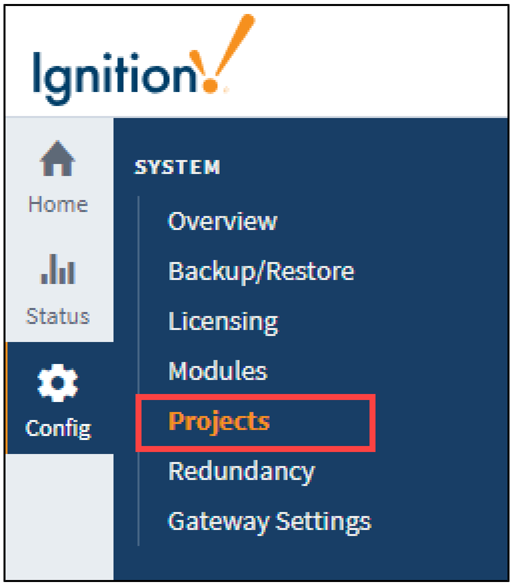
To create a new project or open a different project from within the Designer, use the **File > Open** menu in the top menubar.



The Open/Create Project screen will be displayed. You can choose from existing projects or create a new project.

Updating Project Settings

Project settings such as the title, description, connections, and inheritance are set through the Gateway Webpage **Config** Tab, under **System > Projects**. For more information, see Projects.

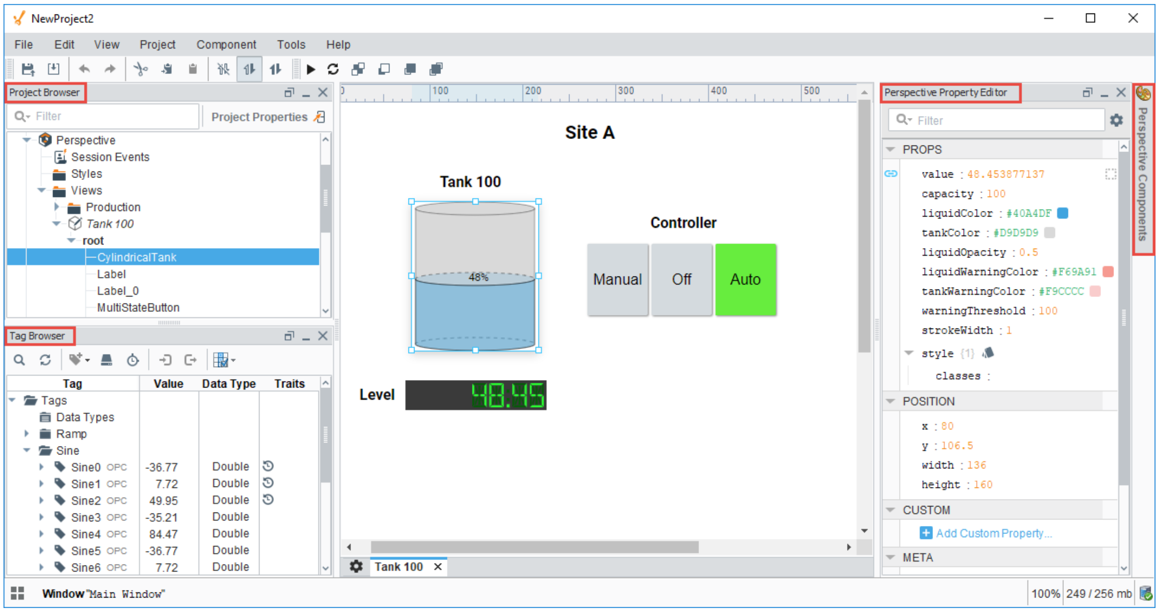


Designer's Workspace

The Designer workspace is centrally located and organized by panels. Some of the panels include a Project Browser, Tag Browser, Component Palette, and Property Editor. These panels can change depending on the type of resource you are currently editing. For example, if you are editing a Perspective view or Vision window, the Designer workspace has Component Palette and Property Editor panels. If you're editing an Alarm Notification Pipeline, your Designer workspace will be the Pipeline Block Editor. If you're editing a Report, your Designer workspace will be the Report Designer. Each type of workspace has panels that are only valid when that workspace is active.

Here are two images showing the Perspective Designer workspace and Vision Designer workspace, and an example displaying the same components. At a glance, they look very similar, but there are some differences, including each having their own Component Palettes. To learn more, refer to the Perspective Designer Interface and the Vision Designer Interface pages.

Perspective Designer's Workspace



Tag	Value	Data Type	Traits
Tags			
Data Types			
Ramp			
Sine			
Sine0 OPC	-36.77	Double	⟲
Sine1 OPC	7.72	Double	⟲
Sine2 OPC	49.95	Double	⟲
Sine3 OPC	-35.21	Double	
Sine4 OPC	84.47	Double	
Sine5 OPC	-36.77	Double	
Sine6 OPC	7.72	Double	

PROPS

```
value : 48.453877137
capacity : 100
liquidColor : #40A4DF
tankColor : #D9D9D9
liquidOpacity : 0.5
liquidWarningColor : #F69A91
tankWarningColor : #F9CCCC
warningThreshold : 100
strokeWidth : 1
style {1}
  classes :
```

POSITION

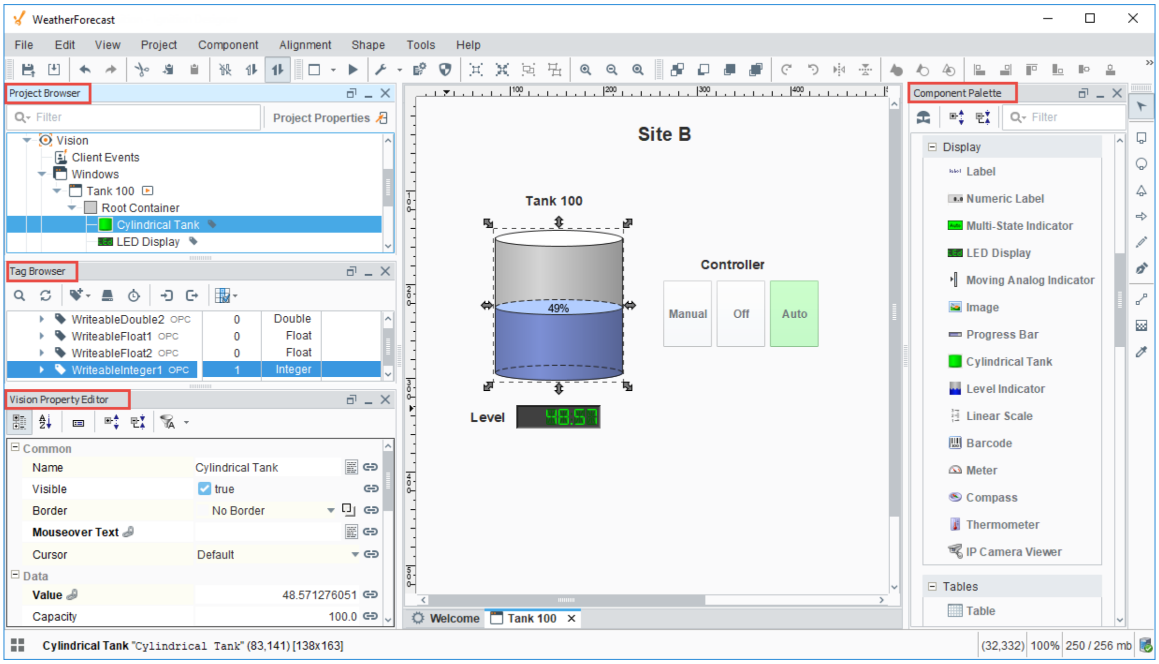
```
x : 80
y : 106.5
width : 136
height : 160
```

CUSTOM

Add Custom Property...

META

Vision Designer's Workspace



WriteableDouble2	OPC	0	Double
WriteableFloat1	OPC	0	Float
WriteableFloat2	OPC	0	Float
WriteableInteger1	OPC	1	Integer

Common

Name	Cylindrical Tank
Visible	☑ true
Border	No Border
Mouseover Text	
Cursor	Default

Data

Value	48.571276051
Capacity	100.0

Cylindrical Tank "Cylindrical Tank" (83,141) [138x163] (32,332) 100% 250 / 256 mb

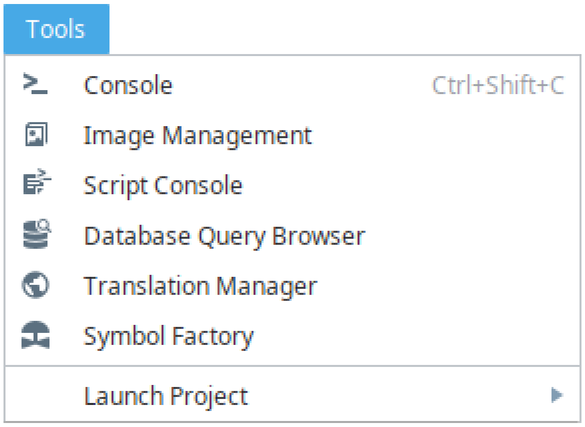
Designer Tools

The Designer has a host of tools to help you accelerate building, testing, and deploying your project. Let's talk about a few here.

Tools Menu

In addition to all the panels available in the Designer workspace, there is also a Tools menu to help you create your projects.

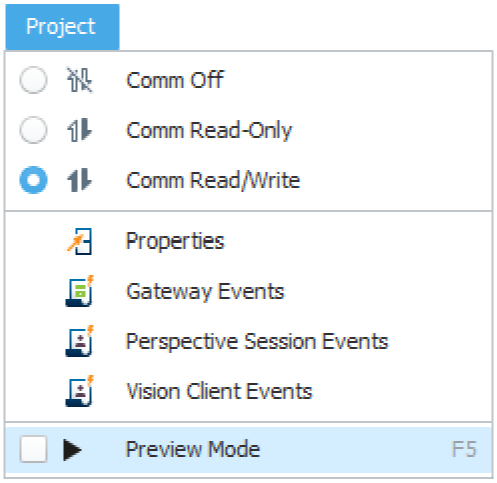
- **Console -** The Output Console is the script-writers best friend, and most frequently used to test and debug Python scripts in Ignition.
- **Image Management** - The Image Management tool manages and stores images used for your projects.
- **Script Console** - The Script Console is used to test and debug Python scripts.
- **Database Query Browser** - The Database Query Browser is a very convenient tool that lets you make simple selects and edits in a database table, and interact with all of the databases that Ignition is connected to (i.e., running queries, browsing tables and schemas). It is very common during the course of project design to inspect the database directly, or to experiment with a SQL query to get it just right.
- **Translation Manager** - Opens up the Translation Manager panel, and allows you to configure language translations. See also: Localization and Languages
- **Symbol Factory** - The Symbol Factory contains a variety of high quality vector graphics symbols that can bring your projects to life.



Previewing the Project

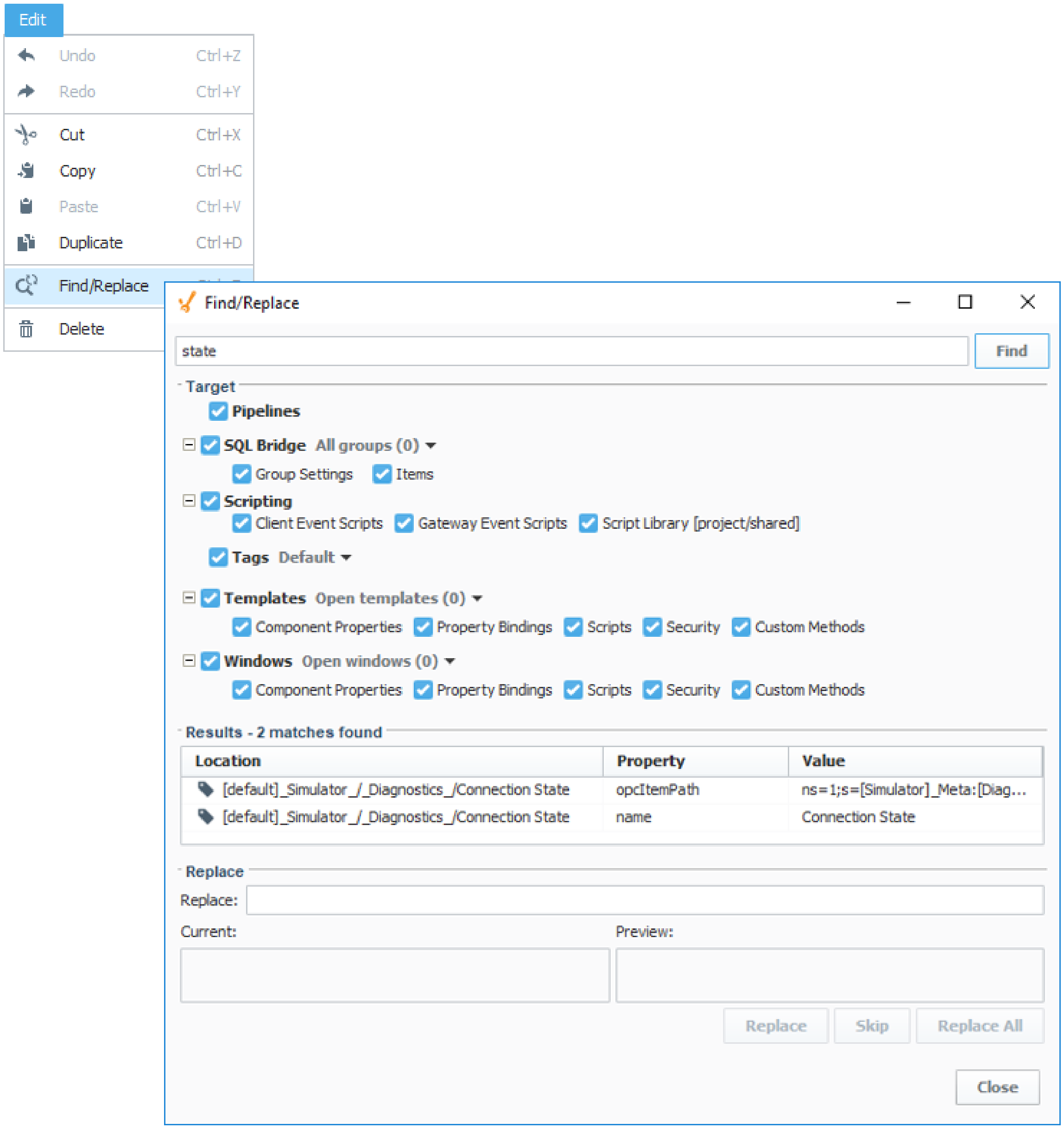
The Designer provides the capability to preview, test, and interact with the screens and functionality before you deploy your project. The Designer workspace operates in two distinct modes: Design mode and Preview mode. Designers can easily switch between these modes to make sure their project is working as expected during the course of development by simply clicking **Preview Mode** ▶ icon from the top menubar, or clicking **Project** also from the top menubar, and selecting **Preview Mode**.

⚠ The **Preview Mode** ▶ icon toggles to be a **Design Mode** ■ icon when it is pressed.



Find and Replace

The Find and Replace is a handy tool in the Designer workspace. You can search your entire project for specific components, properties, and scripts. You can even use the replace command to make mass changes expeditiously to a project with very little effort.



state

Target

☑ Pipelines

☑ SQL Bridge All groups (0)

☑ Group Settings ☑ Items

☑ Scripting

☑ Client Event Scripts ☑ Gateway Event Scripts ☑ Script Library [project/shared]

☑ Tags Default

☑ Templates Open templates (0)

☑ Component Properties ☑ Property Bindings ☑ Scripts ☑ Security ☑ Custom Methods

☑ Windows Open windows (0)

☑ Component Properties ☑ Property Bindings ☑ Scripts ☑ Security ☑ Custom Methods

Results - 2 matches found

Location	Property	Value
[default]_Simulator_/_Diagnostics_/Connection State	opcItemPath	ns=1;s=[Simulator]_Meta:[Diag...
[default]_Simulator_/_Diagnostics_/Connection State	name	Connection State

Replace

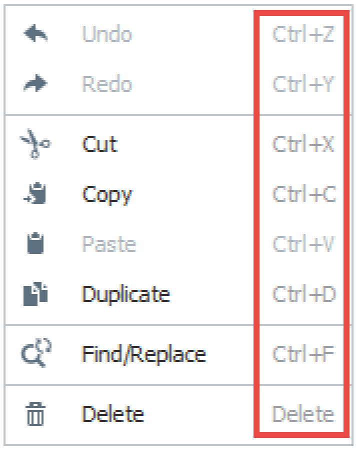
Replace:

Current:

Preview:

Keyboard Shortcuts

There are a lot of ways to speed up your development once you are familiar with how Ignition works. There are many keyboard shortcuts in Designer that are listed throughout the Designer interface alongside menu options.



Concurrent Editing and Conflict Resolution

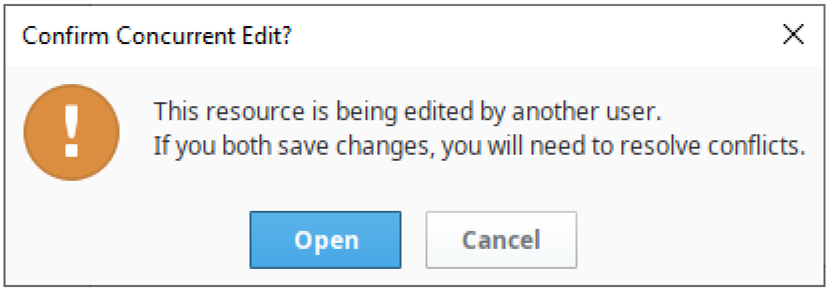
The Ignition Designer uses a lock-free strategy for handling concurrent editing. Multiple people can work together and make changes to a project at the same time. An unlimited number of Designers can be open concurrently, and modifying any resource in the Designer doesn't lock it. The Designer keeps track of the resources that are being edited, and any conflicting edits will be resolved at the time the project is saved. The developer who creates a saving conflict will be prompted to resolve the conflict by choosing whether to use their own changes, accept the other designer's changes, or cancel their save and figure out what to do in another way.

The following feature is new in Ignition version **8.0.10**
Click here to check out the other new features

Concurrent Users UI

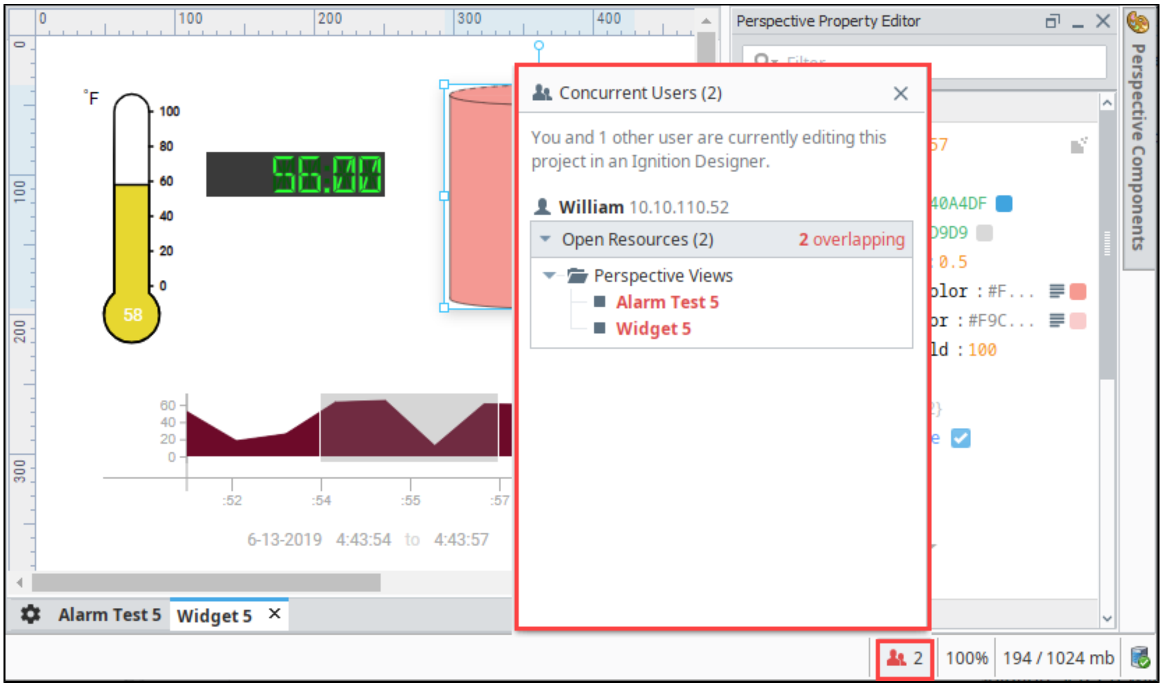
Version 8.0.10 added the Concurrent User Interface that allows users to see which project resources are open in other Designer instances, the names of the users that have them open, and when a project update is available. The new UI is located in the lower right corner of the Designer.

If you open a resource that is already open in another Designer, you will be greeted with a new popup confirming that you want to proceed.


Confirm Concurrent Edit?

This resource is being edited by another user.
If you both save changes, you will need to resolve conflicts.

The UI makes use of the color red to denote when there are conflicts with the changes in another Designer instance, or if their users are saving on overlapping resources (multiple users are making changes and saving while the same resource is open).



Concurrent Users (2)

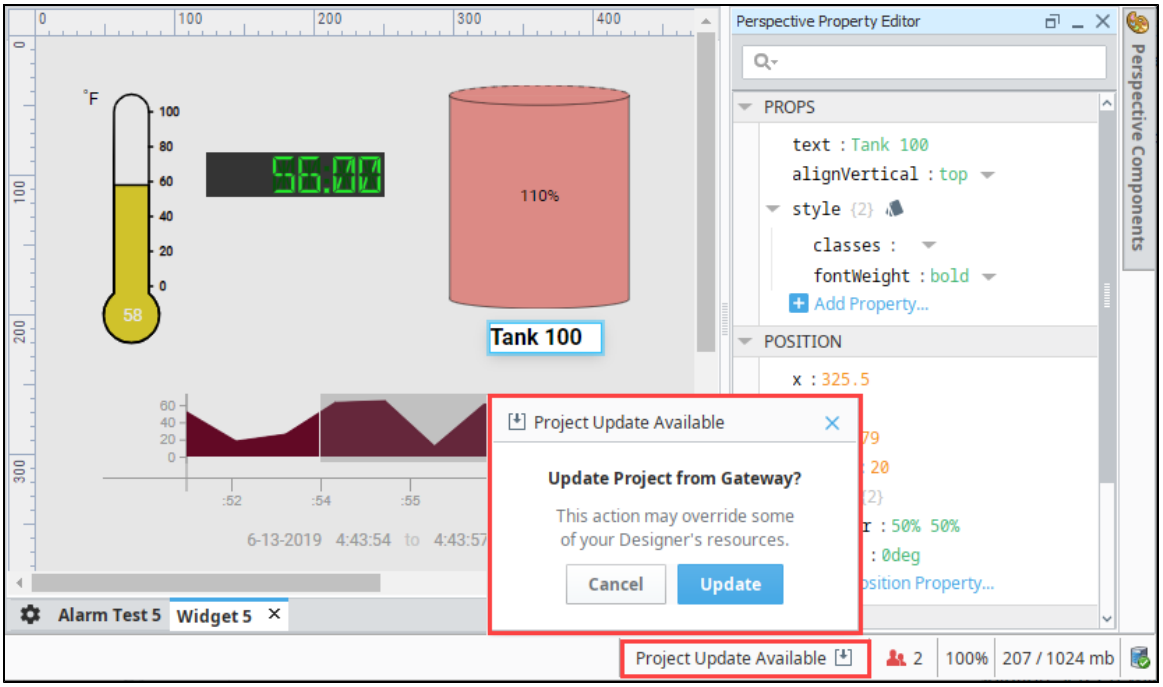
You and 1 other user are currently editing this project in an Ignition Designer.

William 10.10.110.52

Open Resources (2) — 2 overlapping

- Perspective Views
  - Alarm Test 5
  - Widget 5

If the user in the other Designer instance saves their project, the UI will display that a project update is available. Click on **Project Update Available** to either update or cancel the project update.



Project Update Available

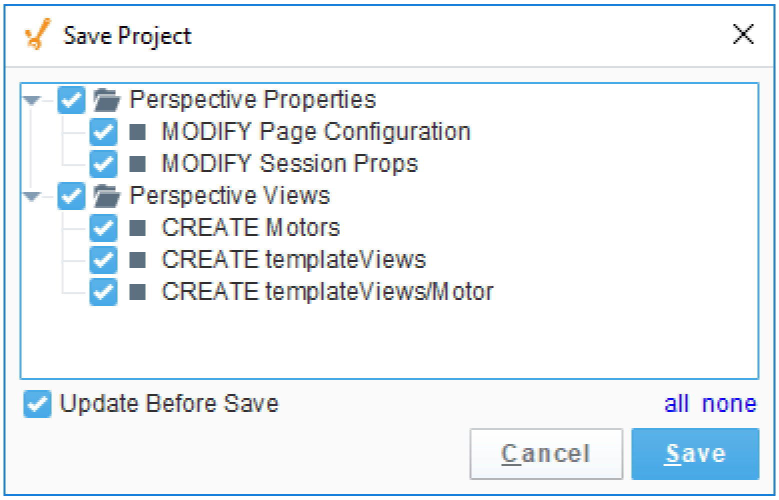
Update Project from Gateway?

This action may override some of your Designer's resources.

Cancel | Update

Conflict Resolution

When you're ready to save your project, go to the Menubar and select **File Save**. When the Save Project dialog pops up, select the items you want to save and click the **Save** button.



Save Project

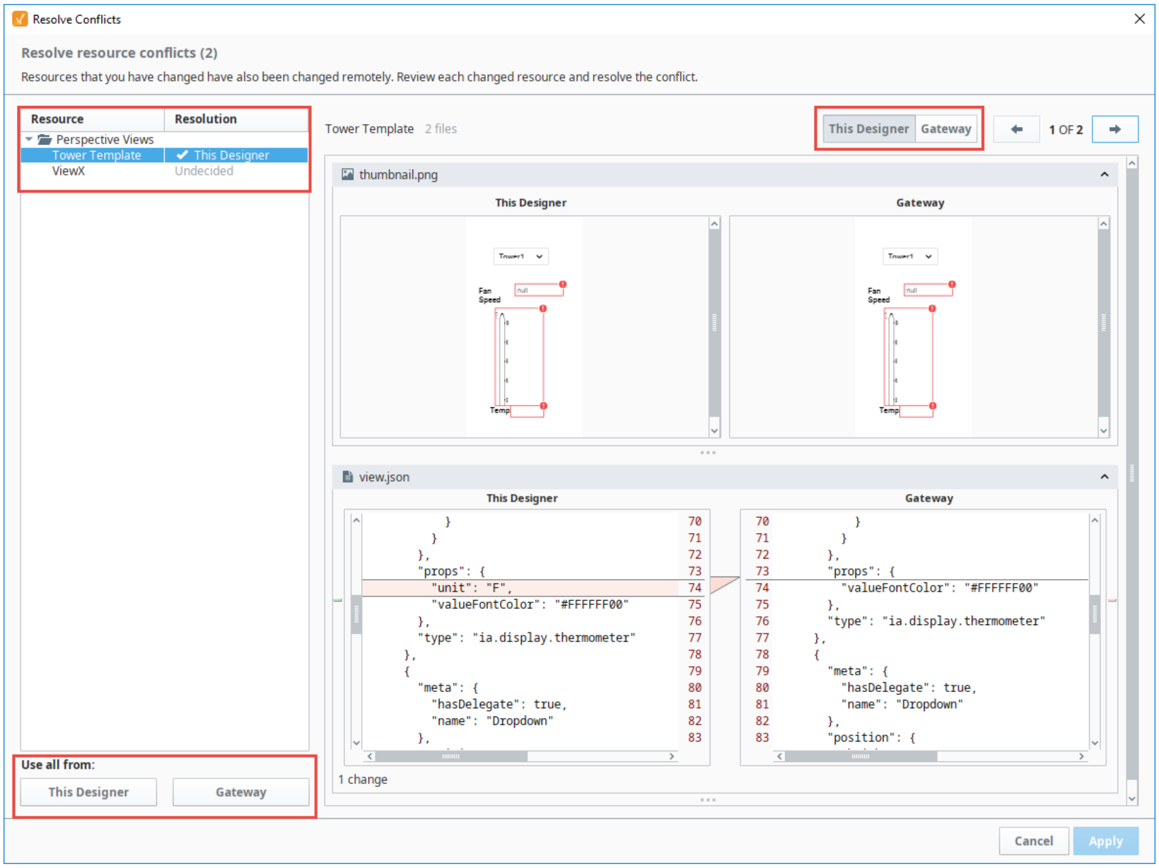
- ☑ Perspective Properties
  - ☑ MODIFY Page Configuration
  - ☑ MODIFY Session Props
- ☑ Perspective Views
  - ☑ CREATE Motors
  - ☑ CREATE templateViews
  - ☑ CREATE templateViews/Motor

☑ Update Before Save

If there are any conflicts, the Designer will ask you to update your project and the conflict resolution screen will open automatically. You will see a thumbnail image of both screens and the .json code showing the conflict. To resolve the conflict, you will be given a choice; accept your changes, accept the other developer's changes, or cancel the save and contact the other developer. Buttons are provided at the top and bottom of the screen for you to enter how you want to resolve the conflict.

> The following feature is new in Ignition version **8.0.3**
> Click here to check out the other new features

The Resource Tree displays the conflicts and the resolutions. The selected conflict will be highlighted in blue. The image below shows changes were made to the Thermometer. The line number and the code are highlighted at the bottom so you can quickly identify the conflict. Once the conflict is resolved, a checkmark will appear in the Resolution column next to the conflict and how it was resolved. You'll notice that there is second conflict. To resolve the conflict, select it in the Resource Tree or click the arrow to navigate to the next conflict.



Resolve Conflicts

Resolve resource conflicts (2)

Resources that you have changed have also been changed remotely. Review each changed resource and resolve the conflict.

Resource	Resolution
Perspective Views	
Tower Template	✓ This Designer
ViewX	Undecided

Tower Template 2 files

thumbnail.png

view.json

This Designer

```
            }
          }
        },
        "props": {
          "unit": "F",
          "valueFontColor": "#FFFFFF00"
        },
        "type": "ia.display.thermometer"
      },
      {
        "meta": {
          "hasDelegate": true,
          "name": "Dropdown"
        },
```

Gateway

```
            }
          }
        },
        "props": {
          "valueFontColor": "#FFFFFF00"
        },
        "type": "ia.display.thermometer"
      },
      {
        "meta": {
          "hasDelegate": true,
          "name": "Dropdown"
        },
        "position": {
```

1 change

Related Topics ...

- Designer Launcher
- Quick Start Guide
- Quick Start - Perspective Session
- Quick Start - Vision Client
- Perspective Designer Interface
- Vision Designer Interface

In This Section ...

Designer Launcher

Overview

The Designer Launcher makes it extremely easy for you to hit the ground running on your projects. The Designer Launcher locates all Gateways that are available on a local network. Each Designer that is added to the My Designers list represents a shortcut to a particular Gateway, and once that Designer is opened you can access or create new projects on that Gateway. The same Designer can access all the projects that are available on that Gateway, and you don't need to add the Designer multiple times for different Gateways!

Create a Designer Launcher shortcut on your desktop and you can open your Designer, and create or edit a project in a flash!

Login credentials/permissions

If the Designer Launcher locates and adds a Gateway to your list of designers, you must have the right credentials/privileges to login to that particular Designer.

Best Practice

It's good practice to download and install the Designer Launcher every time you install a new release of Ignition. The Designer Launcher is a separate application and is not part of an Ignition release, so it's important not to have an outdated Designer Launcher shortcut residing on your desktop. Each time you install a new version of Ignition, download and install the Designer Launcher, and let the install process create a new desktop shortcut for you automatically.

The Designer Launcher for Windows, Linux, and Mac OS are available on the **Home** tab of the Gateway Webpage. Ignition knows what platform you're running, and will display it on the Download button. There is section on how to download and install the Designer Launcher for each operating system. Simply click on the link in each section to expand the panel to guide you through the steps.

On this page

...

INDUCTIVE UNIVERSITY

The Designer Launcher

Watch the Video

Windows Designer Launcher

The following feature is new in Ignition version **8.0.10**
Click here to check out the other new features

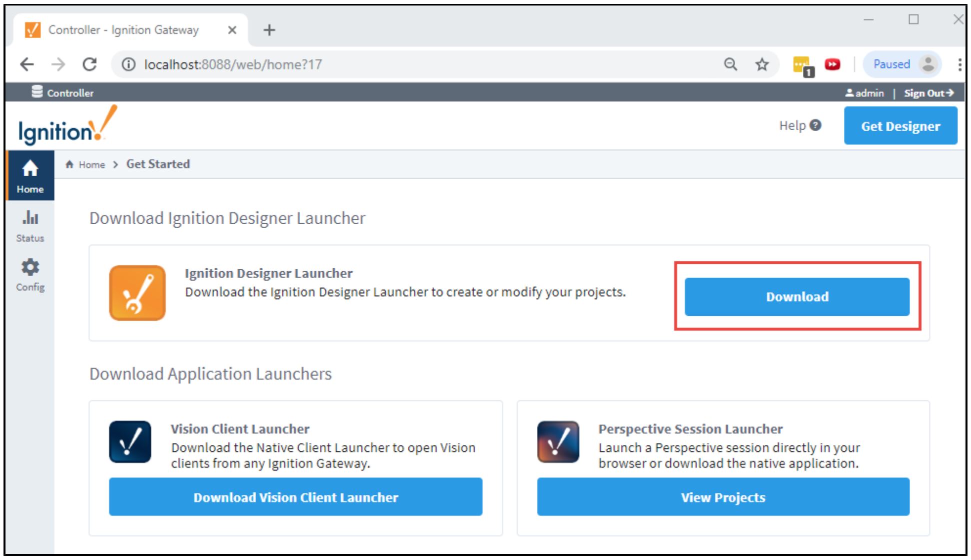
The Designer Launcher for Windows can now be installed as an "all user" application on a Windows operating system providing launcher access to all users on the same machine from a single installation. The first time you install the Designer Launcher, you will be prompted to select your install mode: "Install for me only" or to "Install for all users." If the Designer Launcher was already installed, subsequent install attempts are treated as an upgrade and you will not be prompted to select the install mode. This new option will only appear on new launcher installations, or after uninstalling a prior version of the installer.

Designer Launcher Destination Folder

When installing for "all users," the installer will be placed in the following folder: "%Ignition Installation Directory%\Program Files\Inductive Automation."
When installing for "me only," the installer will be found under the "\AppData\Roaming\Inductive Automation" folder.

Use the following steps to download and install the Designer Launcher application for the **Windows** platform.

1. You can access the Designer Launcher on the **Home** tab of the Gateway Webpage. Click **Download** to start the download process for the Designer Launcher.

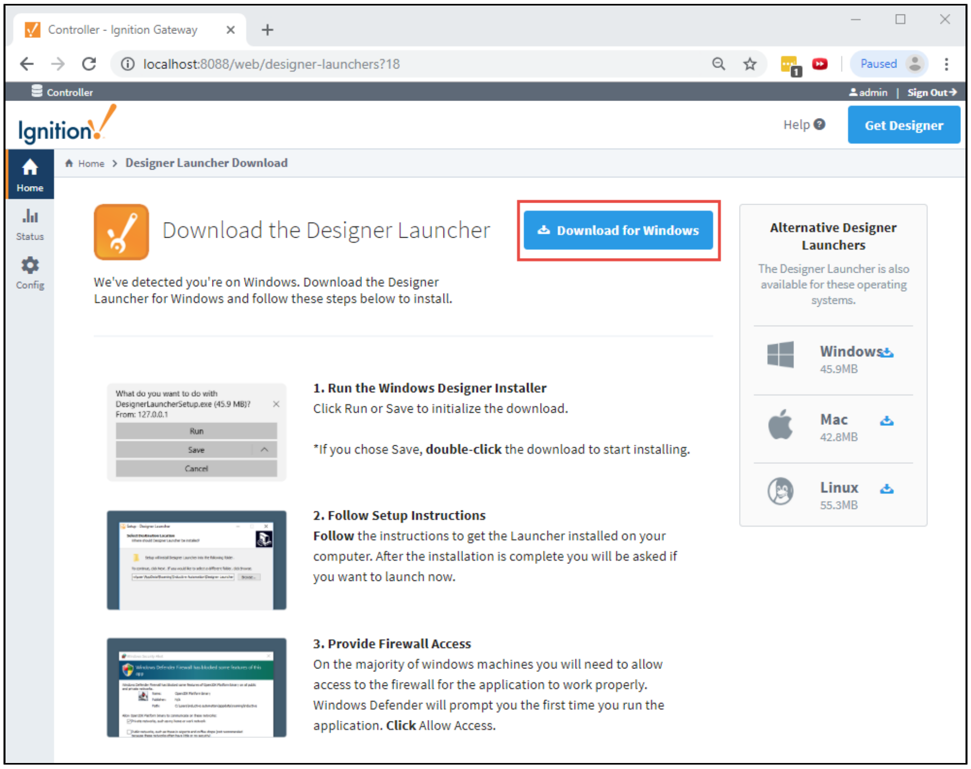

Controller - Ignition Gateway
localhost:8088/web/home?17
Paused
Controller
admin | Sign Out
Ignition
Help
Get Designer
Home > Get Started
Home
Status
Config
Download Ignition Designer Launcher
Ignition Designer Launcher
Download
Download Application Launchers
Vision Client Launcher
Download Vision Client Launcher
Perspective Session Launcher
View Projects

Download the Ignition Designer Launcher to create or modify your projects.

Download the Native Client Launcher to open Vision clients from any Ignition Gateway.

Launch a Perspective session directly in your browser or download the native application.

2. Your web browser will detect the operating system, and suggest the appropriate download for the **Download** button.


Controller - Ignition Gateway
localhost:8088/web/designer-launchers?18
Paused
Controller
admin | Sign Out
Ignition
Help
Get Designer
Home > Designer Launcher Download
Home
Status
Config
Download the Designer Launcher
Download for Windows
Alternative Designer Launchers
Windows
45.9MB
Mac
42.8MB
Linux
55.3MB
What do you want to do with DesignerLauncherSetup.exe (45.9 MB)?
From: 127.0.0.1
Run
Save
Cancel

We've detected you're on Windows. Download the Designer Launcher for Windows and follow these steps below to install.

The Designer Launcher is also available for these operating systems.

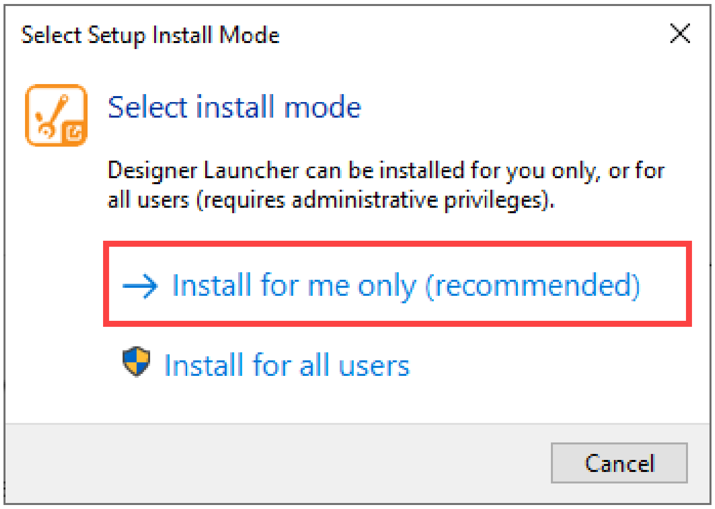
1. Run the Windows Designer Installer
Click Run or Save to initialize the download.

*If you chose Save, **double-click** the download to start installing.

2. Follow Setup Instructions
Follow the instructions to get the Launcher installed on your computer. After the installation is complete you will be asked if you want to launch now.

3. Provide Firewall Access
On the majority of windows machines you will need to allow access to the firewall for the application to work properly. Windows Defender will prompt you the first time you run the application. **Click** Allow Access.

3. Run the downloaded file (**DesignerLauncherSetup.exe**) found on the lower-left of the window, or go to your **Downloads** folder and double-click the file to run it.
4. If this is the first time installing the Designer Launcher, you will be prompted to select an install mode. Select either **"Install for me only,"** or **"Install for all users".**

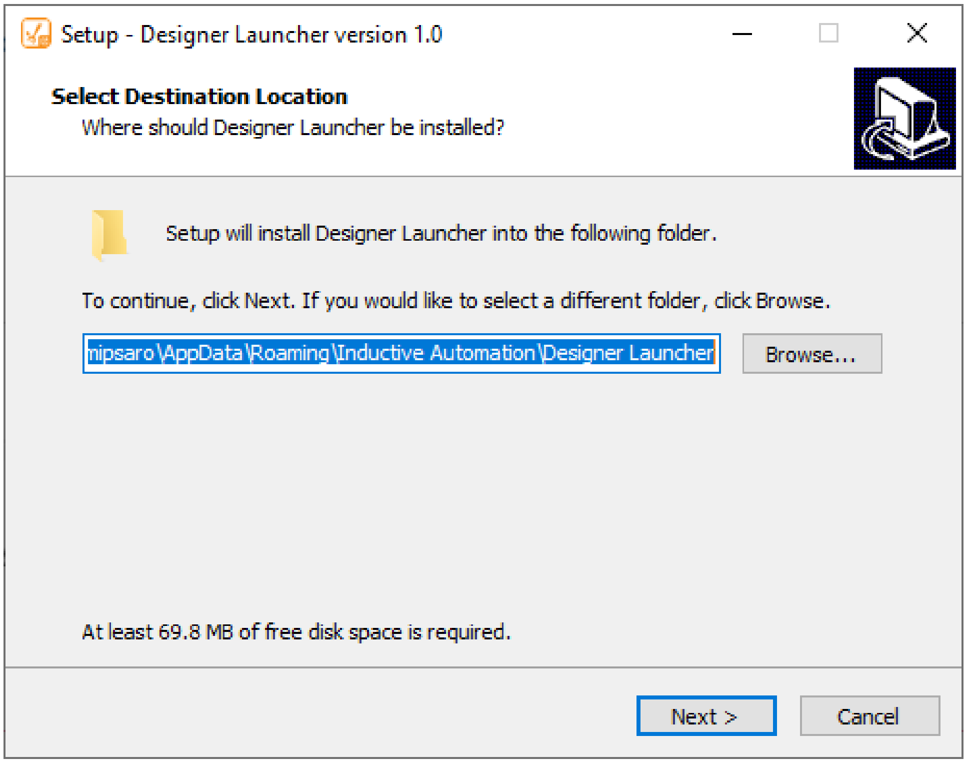


Select Setup Install Mode

Select install mode

Designer Launcher can be installed for you only, or for all users (requires administrative privileges).

5. The Install Wizard will select a default destination location folder for the Designer Launcher application files. If you are "installing for me only", the destination folder will be different from the "all users" destination folder.



Setup - Designer Launcher version 1.0

Select Destination Location

Where should Designer Launcher be installed?

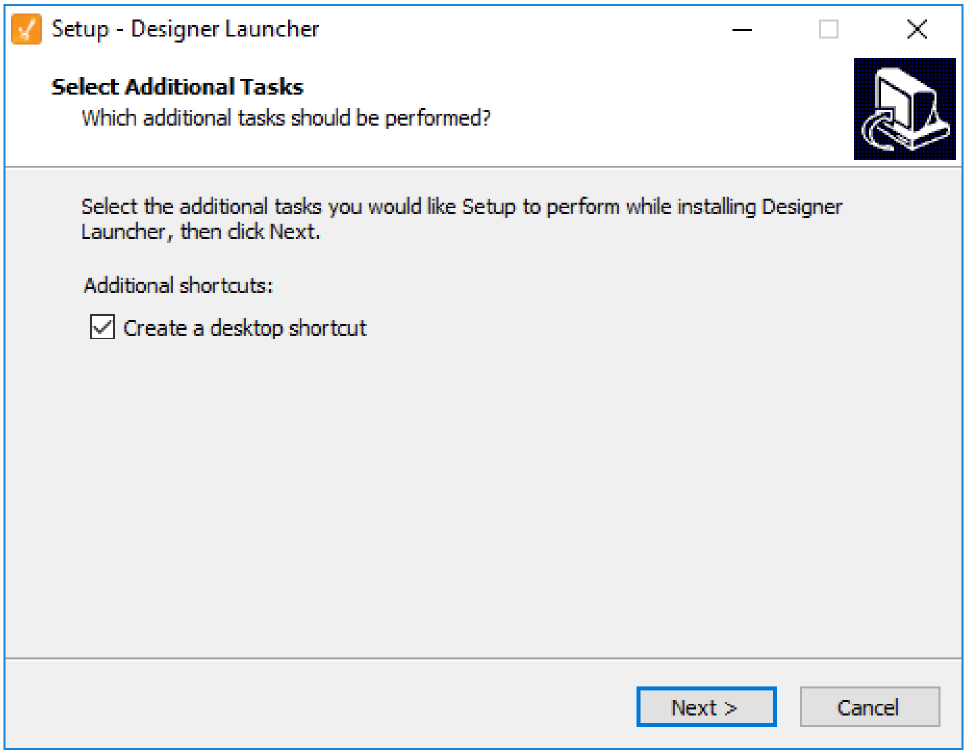
Setup will install Designer Launcher into the following folder.

To continue, click Next. If you would like to select a different folder, click Browse.

nipsaro\AppData\Roaming\Inductive Automation\Designer Launcher

At least 69.8 MB of free disk space is required.

6. The Select Additional Tasks window will open. You'll notice that a desktop shortcut is checked and will be created so you don't have to run the Designer Launcher each time you want to open the Designer. Click **Next**.



Setup - Designer Launcher

Select Additional Tasks

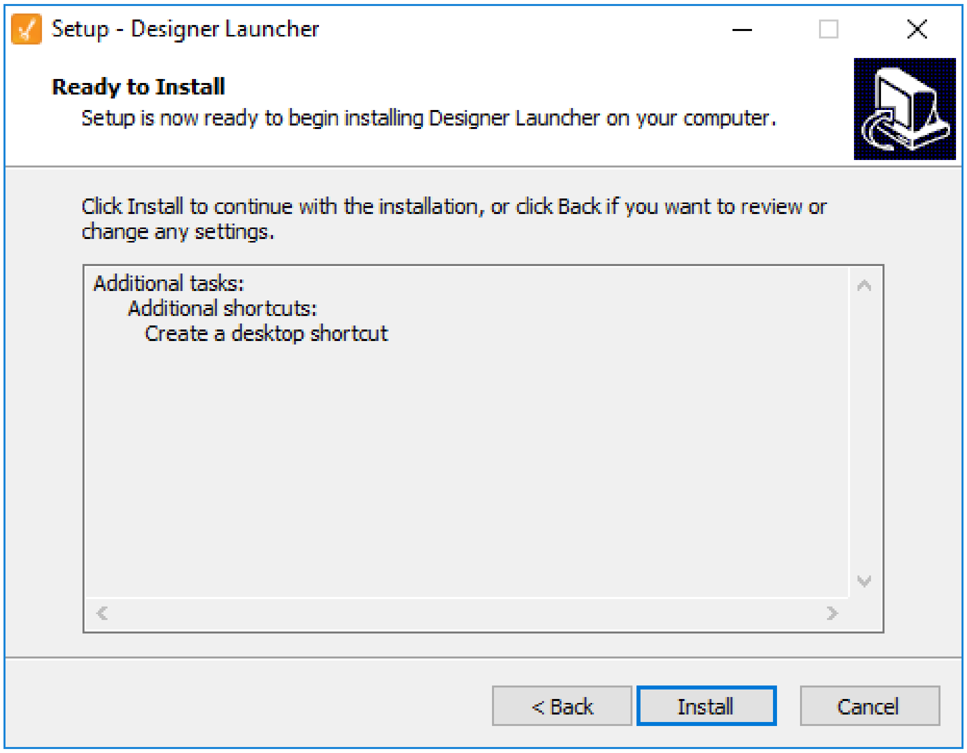
Which additional tasks should be performed?

Select the additional tasks you would like Setup to perform while installing Designer Launcher, then click Next.

Additional shortcuts:

☑ Create a desktop shortcut

7. Now, you're Ready to Install the Designer Launcher on your computer. Click **Install** to continue.



Setup - Designer Launcher

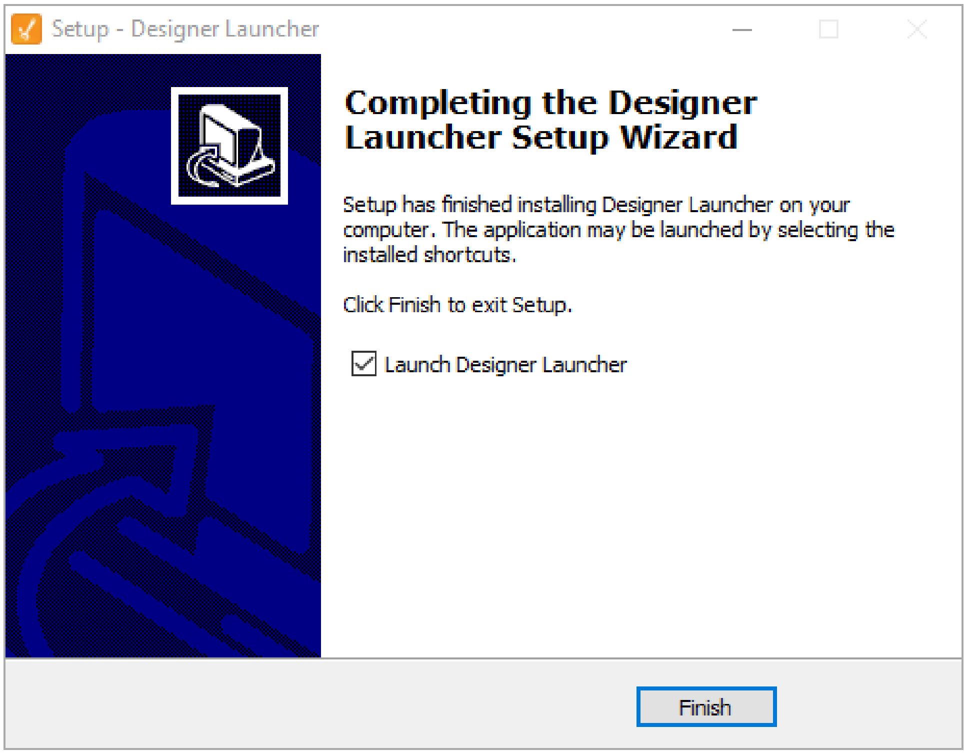
Ready to Install

Setup is now ready to begin installing Designer Launcher on your computer.

Click Install to continue with the installation, or click Back if you want to review or change any settings.

```
Additional tasks:
    Additional shortcuts:
      Create a desktop shortcut
```

8. Once the Designer Launcher is installed, click **Finish**.



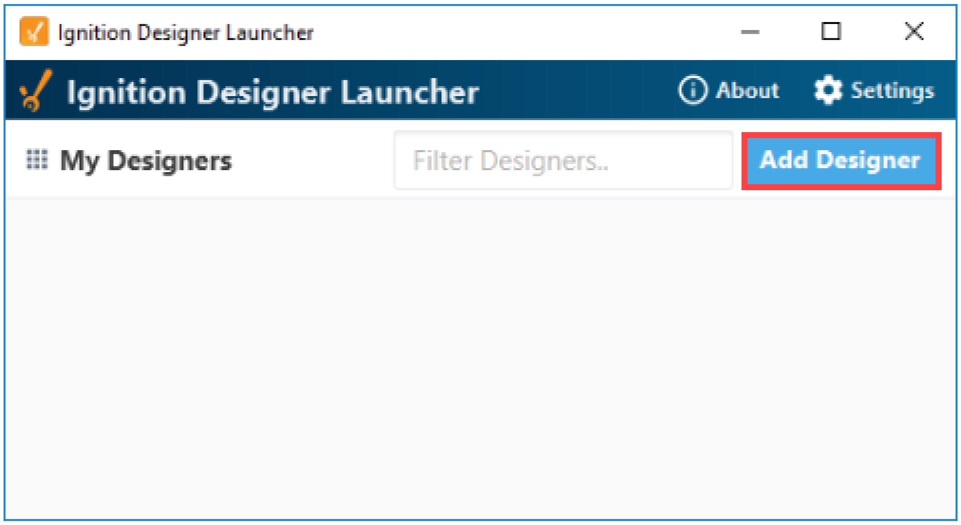
Completing the Designer Launcher Setup Wizard

Setup has finished installing Designer Launcher on your computer. The application may be launched by selecting the installed shortcuts.

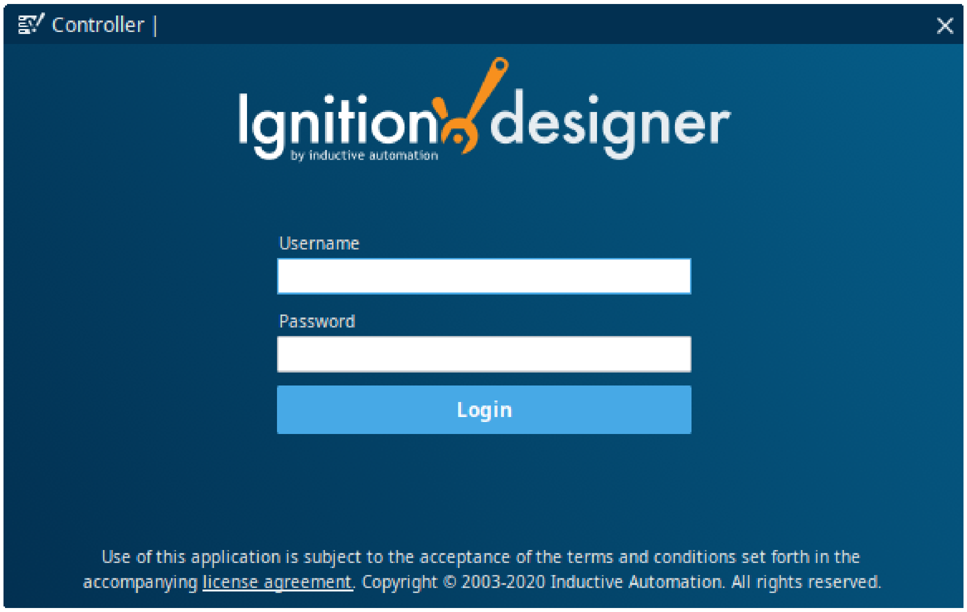
Click Finish to exit Setup.

☑ Launch Designer Launcher

9. The Ignition Designer Launcher window will open. Here you can select any Designers you have permission to launch. If you are running this for the first time, you will need to add your Designer(s). Adding Designers is addressed in the Adding Additional Designers section below.

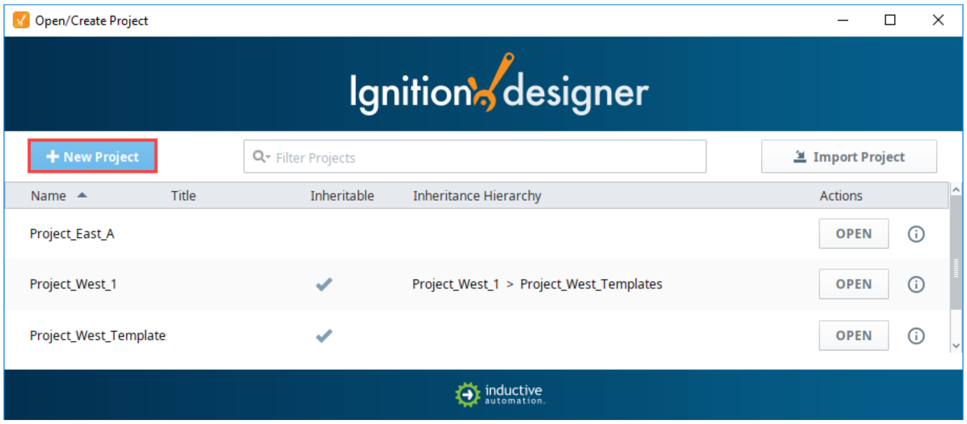


10. The Designer starts up and the login window is displayed. Enter your credentials for the Designer, and click **Login**. Logging into the designer requires a user account in a User Sources, the user source must be set as the System User Source under Gateway Settings, and the user account must have the at least one of the designer roles (again, under Gateway Settings).



Use of this application is subject to the acceptance of the terms and conditions set forth in the accompanying license agreement. Copyright © 2003-2020 Inductive Automation. All rights reserved.

11. The **Open/Create Project** window is displayed giving you two options: create a new project or open an existing one. Enter a project name and then click the **New Project** button.

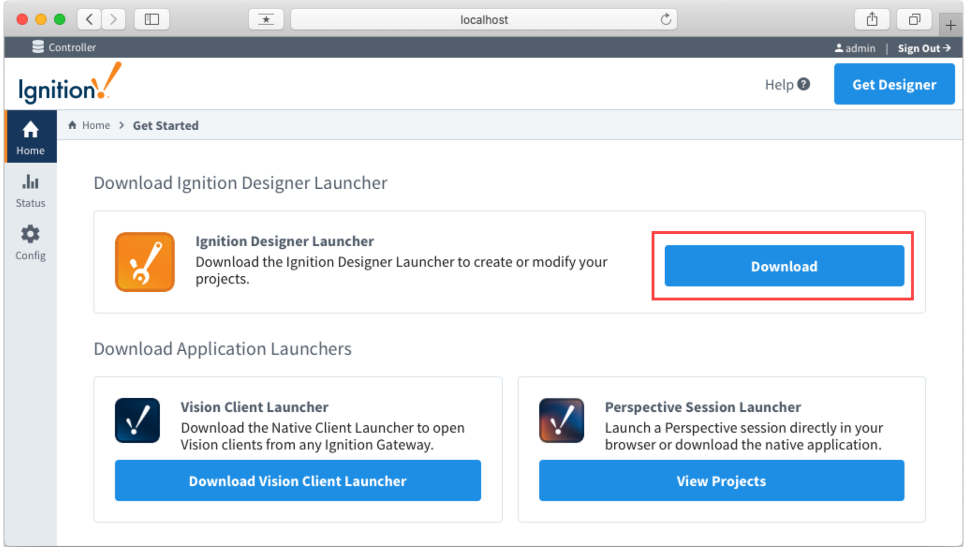


Name	Title	Inheritable	Inheritance Hierarchy	Actions
Project_East_A				OPEN
Project_West_1		✓	Project_West_1 > Project_West_Templates	OPEN
Project_West_Template		✓		OPEN

Mac OS Designer Launcher

Use the following steps to download and install the Designer Launcher application for the **Mac OS** platform.

1. From the **Home** tab on the Gateway Webpage, click the **Download** button for the Ignition Designer Launcher.



Home > Get Started

Download Ignition Designer Launcher

Ignition Designer Launcher
Download the Ignition Designer Launcher to create or modify your projects.

Download

Download Application Launchers

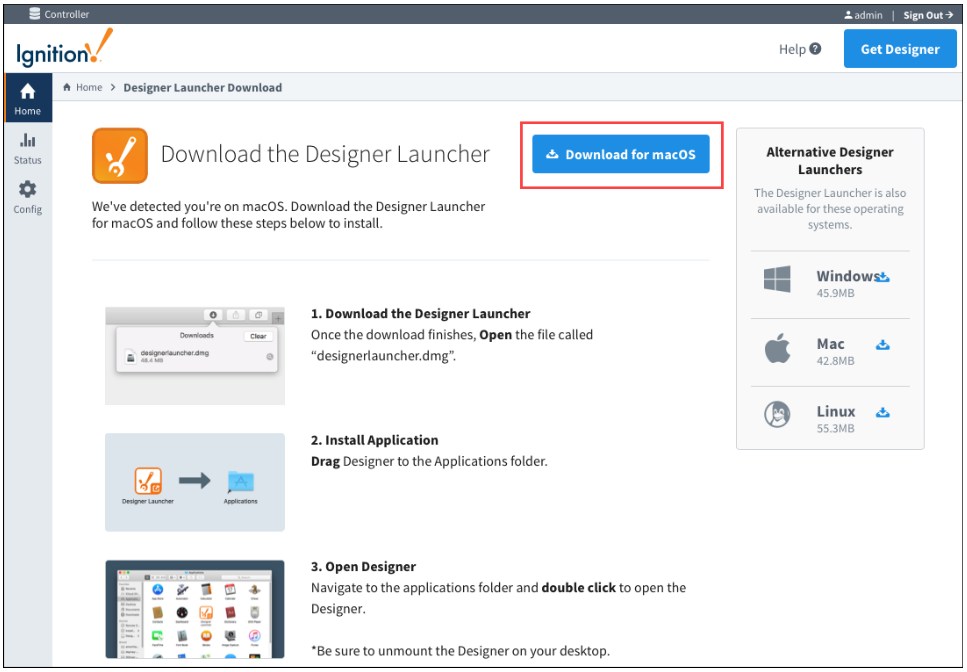
Vision Client Launcher
Download the Native Client Launcher to open Vision clients from any Ignition Gateway.

Download Vision Client Launcher

Perspective Session Launcher
Launch a Perspective session directly in your browser or download the native application.

View Projects

2. Your web browser will detect the operating system, and will display **"macOS"** on the Download button. Click on the **Download** button to download the Designer Launcher.



Home > Designer Launcher Download

Download the Designer Launcher

Download for macOS

We've detected you're on macOS. Download the Designer Launcher for macOS and follow these steps below to install.

1. Download the Designer Launcher
Once the download finishes, **Open** the file called "designerlauncher.dmg".

2. Install Application
Drag Designer to the Applications folder.

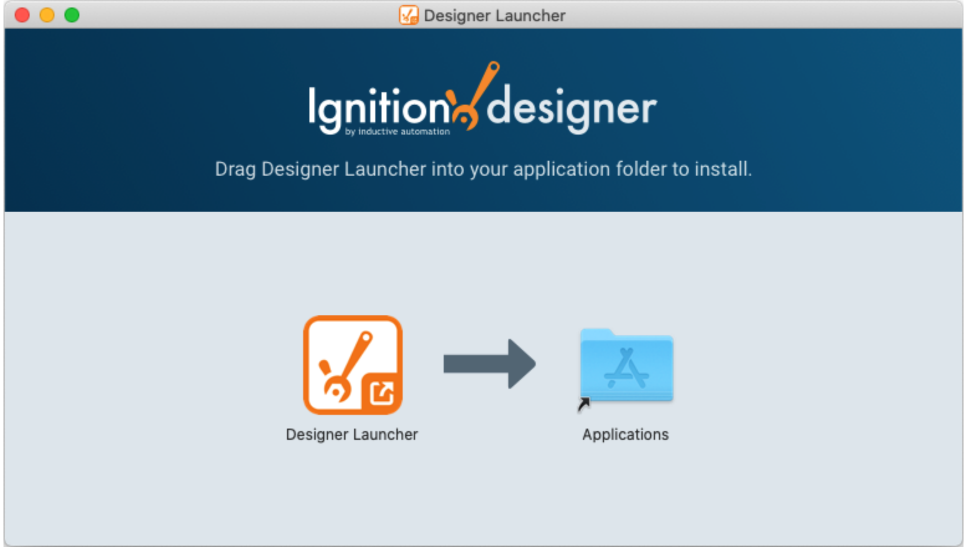
3. Open Designer
Navigate to the applications folder and **double click** to open the Designer.

*Be sure to unmount the Designer on your desktop.

Alternative Designer Launchers
The Designer Launcher is also available for these operating systems.

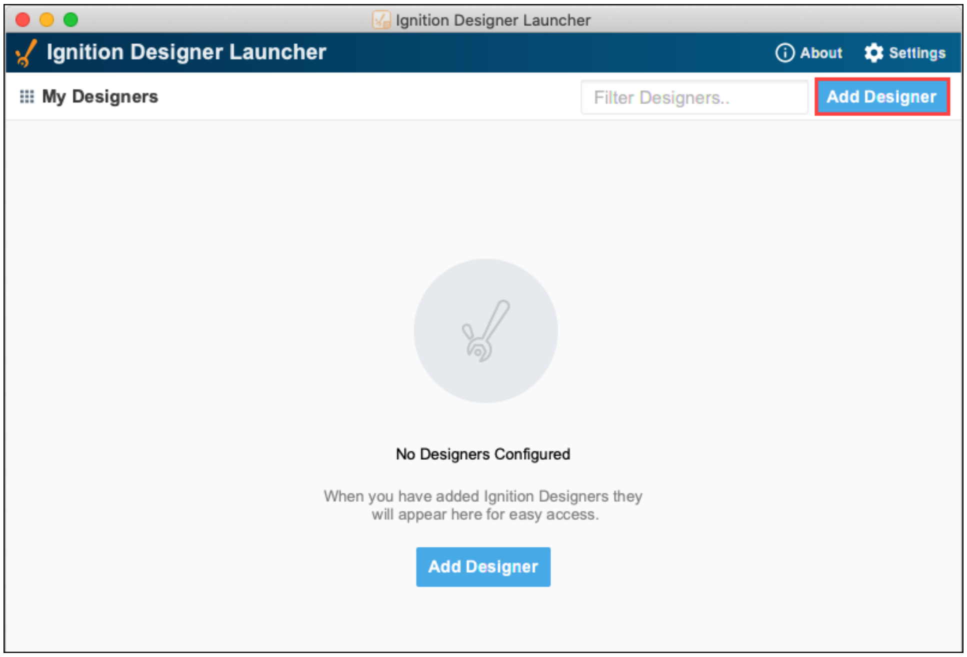
Windows	45.9MB
Mac	42.8MB
Linux	55.3MB

3. Run the downloaded file (**designerlauncher.dmg**) found in your **Downloads** folder.
4. To install the Designer Launcher application, drag the Designer Launcher icon to the Applications folder.



Drag Designer Launcher into your application folder to install.

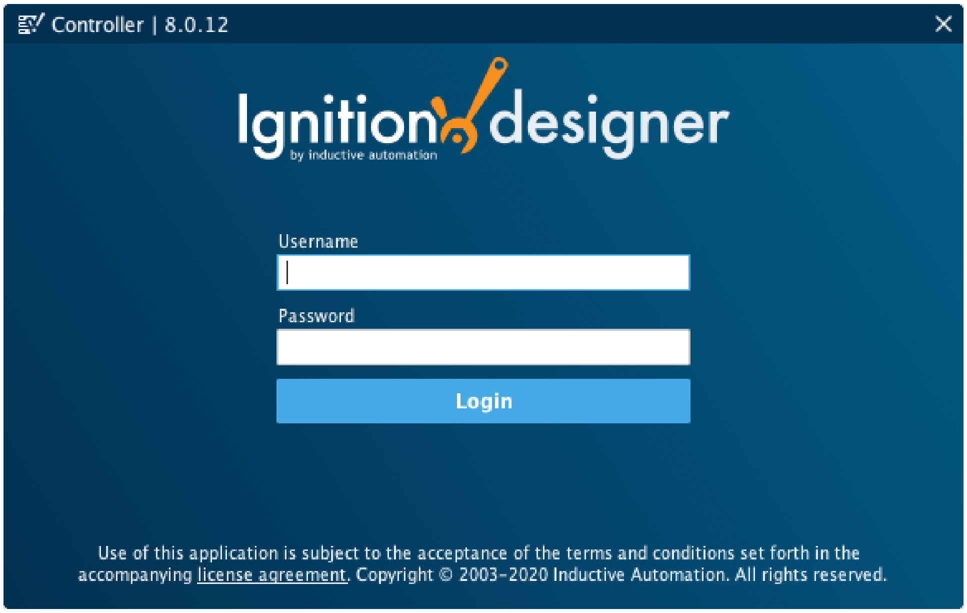
5. Navigate to the applications folder and double click on the Designer Launcher icon to open the Designer.
6. The Ignition Designer Launcher window will open. Here you can select any Designers you have permission to launch. If you are running this for the first time, you will need to add your Designer(s). Adding Designers is addressed in the Adding Additional Designers section.



No Designers Configured

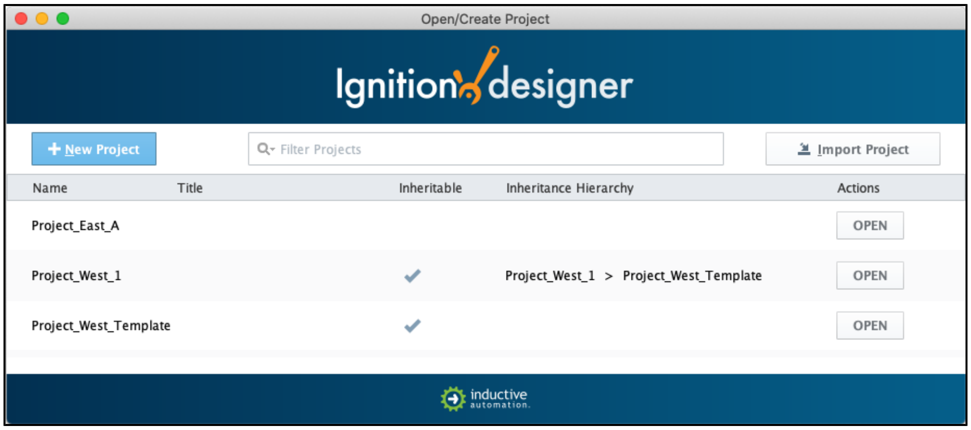
When you have added Ignition Designers they will appear here for easy access.

7. The Designer starts up and the login window is displayed. Enter your credentials for the Designer, and click **Login**. Logging into the designer requires a user account in a User Sources, the user source must be set as the System User Source under Gateway Settings, and the user account must have the at least one of the designer roles (again, under Gateway Settings).



Use of this application is subject to the acceptance of the terms and conditions set forth in the accompanying license agreement. Copyright © 2003-2020 Inductive Automation. All rights reserved.

8. The **Open/Create Project** window is displayed giving you two options: create a new project or open an existing one. Enter a project name and then click the **New Project** button.

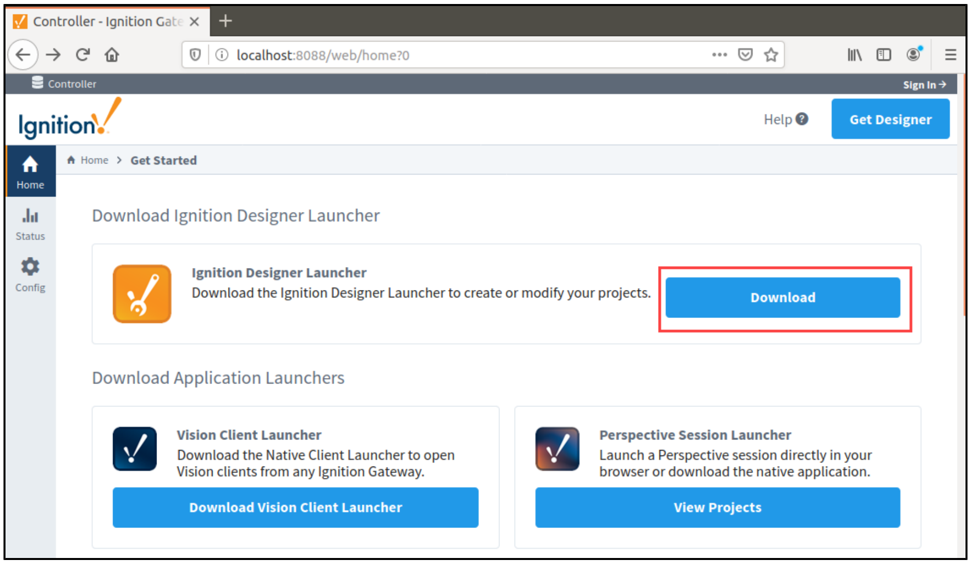


Name	Title	Inheritable	Inheritance Hierarchy	Actions
Project_East_A				OPEN
Project_West_1		✓	Project_West_1 > Project_West_Template	OPEN
Project_West_Template		✓		OPEN

Linux Designer Launcher

Use the following steps to download and install the Designer Launcher application for the **Linux** platform.

1. From the **Home** tab on the Gateway Webpage, click the **Download** button for the Ignition Designer Launcher.



Download Ignition Designer Launcher

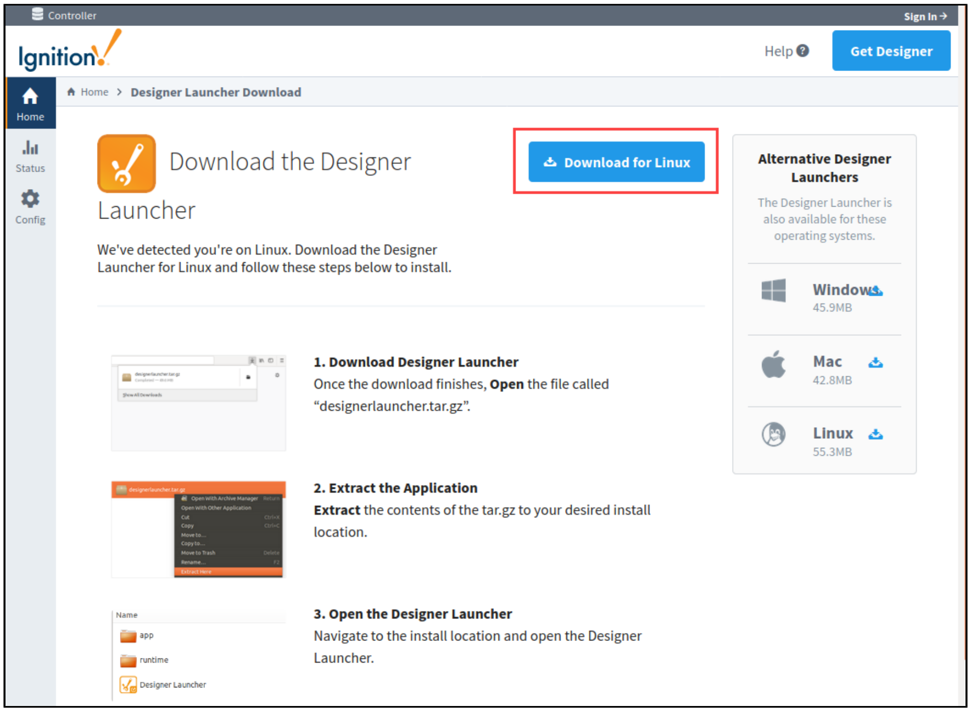
Ignition Designer Launcher
Download the Ignition Designer Launcher to create or modify your projects.

Download Application Launchers

Vision Client Launcher
Download the Native Client Launcher to open Vision clients from any Ignition Gateway.

Perspective Session Launcher
Launch a Perspective session directly in your browser or download the native application.

2. Your web browser will detect the operating system, and will display **"Linux"** on the Download button. Click the **Download** button to download the Designer Launcher.



Download the Designer Launcher

We've detected you're on Linux. Download the Designer Launcher for Linux and follow these steps below to install.

1. Download Designer Launcher
Once the download finishes, **Open** the file called "designerlauncher.tar.gz".

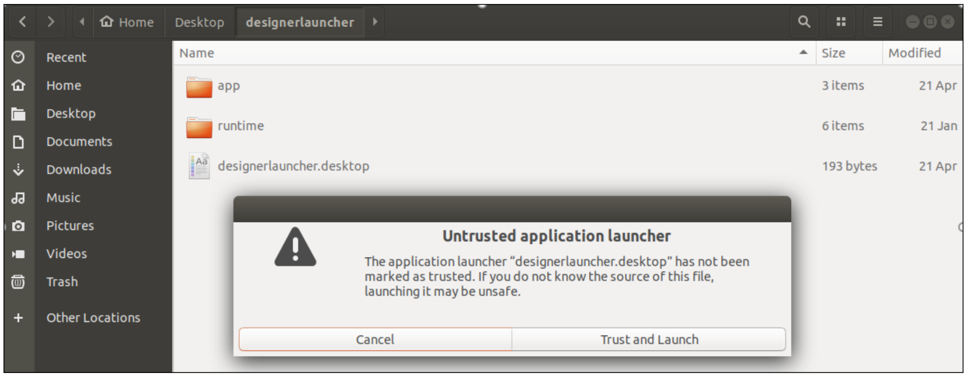
2. Extract the Application
Extract the contents of the tar.gz to your desired install location.

3. Open the Designer Launcher
Navigate to the install location and open the Designer Launcher.

Alternative Designer Launchers
The Designer Launcher is also available for these operating systems.

Windows	45.9MB
Mac	42.8MB
Linux	55.3MB

3. Once the download completes, open the file called **"designerlauncher.tar.gz"**.
4. Extract the contents of the **'tar.gz'** file to your desired install location.
5. Navigate to your install folder, and click the **'designerlauncher.desktop'** file. A message box may popup about the Designer Launcher being an untrusted application. Click **Trust and Launch**.

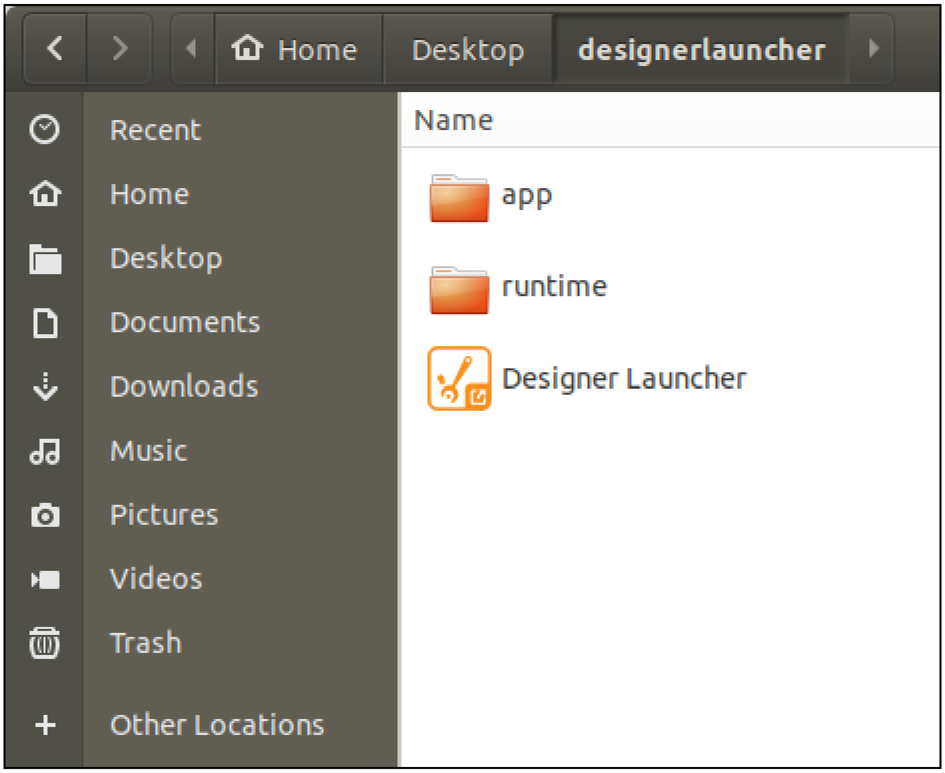


Name	Size	Modified
app	3 items	21 Apr
runtime	6 items	21 Jan
designerlauncher.desktop	193 bytes	21 Apr

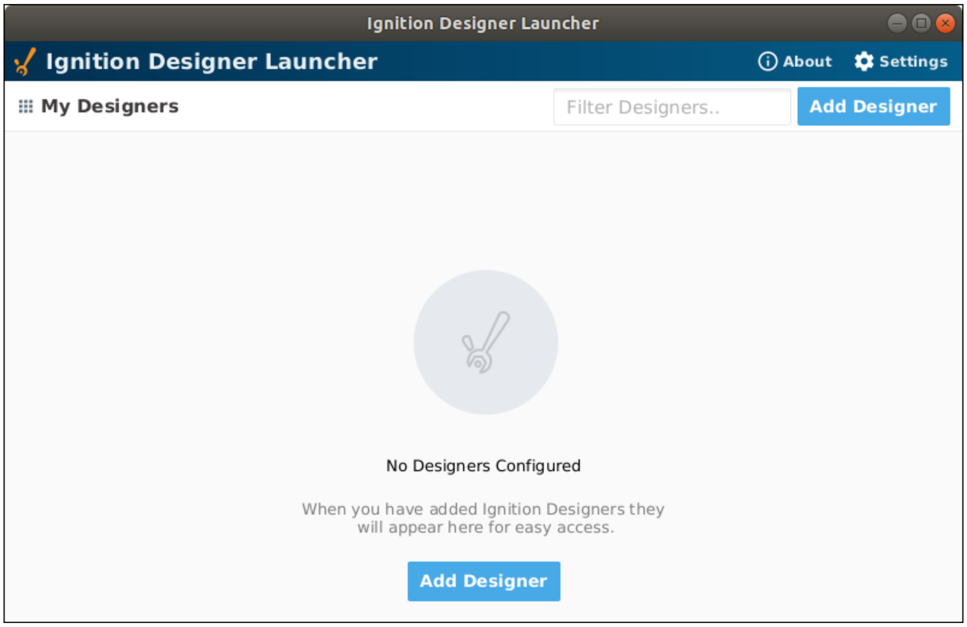
Untrusted application launcher

The application launcher “designerlauncher.desktop” has not been marked as trusted. If you do not know the source of this file, launching it may be unsafe.

6. Click the Designer Launcher icon to launch the Designer Launcher.



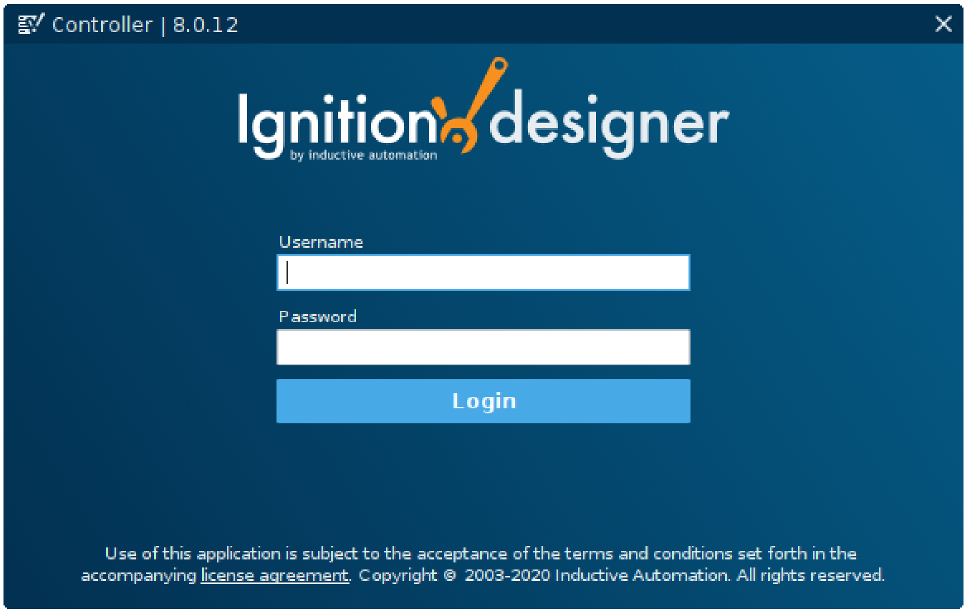
7. The Ignition Designer Launcher window will open. Here you can select any Designers you have permission to launch. If you are running this for the first time, you will need to add your Designer(s). Adding Designers is addressed in the Adding Additional Designers section.



No Designers Configured

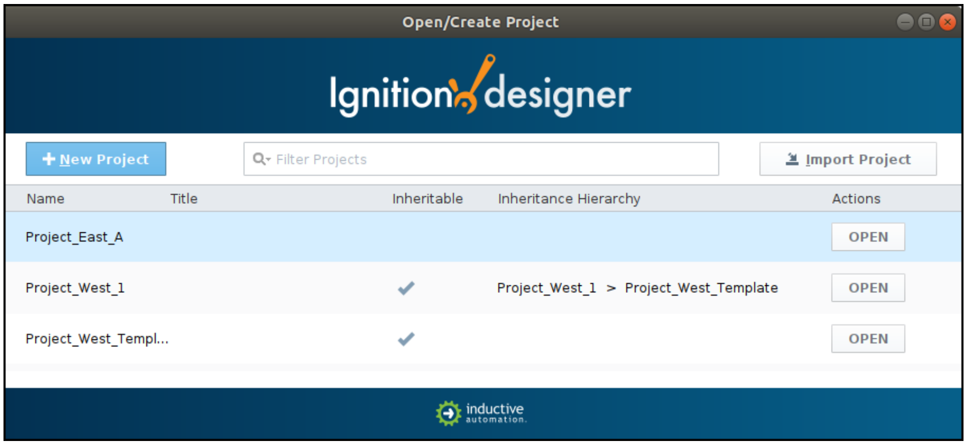
When you have added Ignition Designers they will appear here for easy access.

8. The Designer starts up and the login window is displayed. Enter your credentials for the Designer, and click **Login**. Logging into the designer requires a user account in a User Sources, the user source must be set as the System User Source under Gateway Settings, and the user account must have the at least one of the designer roles (again, under Gateway Settings).



Use of this application is subject to the acceptance of the terms and conditions set forth in the accompanying license agreement. Copyright © 2003-2020 Inductive Automation. All rights reserved.

9. The **Open/Create Project** window is displayed giving you two options: create a new project or open an existing one. Enter a project name and then click the **New Project** button.



Name	Title	Inheritable	Inheritance Hierarchy	Actions
Project_East_A				OPEN
Project_West_1		✓	Project_West_1 > Project_West_Template	OPEN
Project_West_Templ...		✓		OPEN

Configuring the Designer Launcher

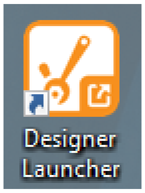
Configuring the Designer Launcher is fairly simple no matter what platform you're working in. The process steps for configuring the launcher is the same for all three platforms: Windows, Mac OS, or Linux.

> The steps and screenshots in this section were taken from a Windows platform. These steps are the same for Mac OS and Linux, but the screenshots for Mac OS and Linux may have a slightly different appearance. This section is used to configure the Designer Launcher for all three platforms: Windows, Mac OS, and Linux.

The Designer Launcher finds the Gateways, you add the Gateways to My Designers, and the only thing left to do is create your Designer shortcuts. You can create a shortcut for each Designer, or use the Designer Launcher to display all your Designers. Creating shortcuts makes them readily accessible for opening a Designer quickly. If you are running this for the first time, you'll need to add your Designer(s). Adding Designers is addressed in the Adding Additional Designers section below.

Let's open the Designer Launcher to create a few shortcuts.

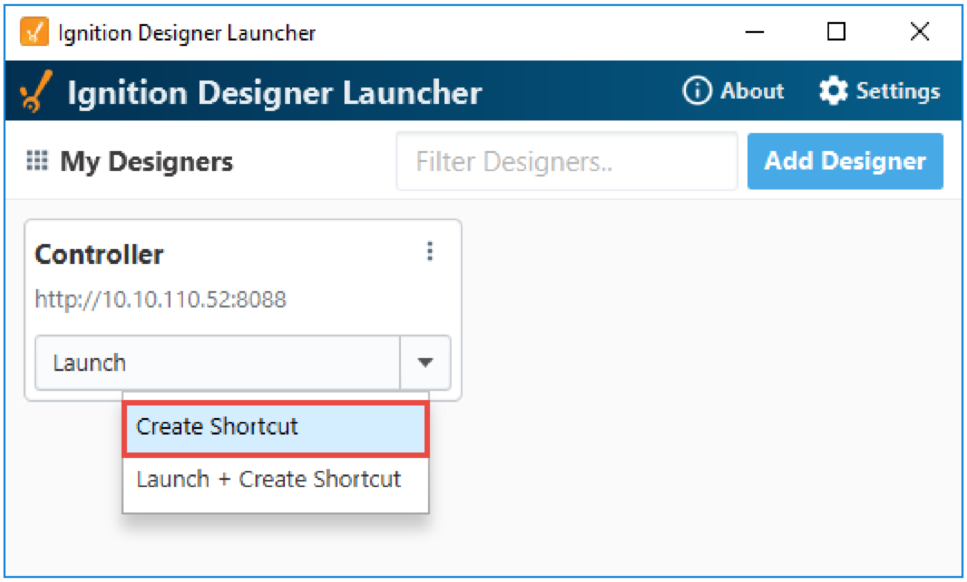
1. If you followed the Designer Launcher Download and Install steps for your platform, then you probably already have the Designer Launcher shortcut created which looks like the image below.

   If you don't, then go back to the Gateway Webpage to download and install the Designer Launcher again, following the steps documented for your platform.

2. Find the Designer Launcher shortcut and click it to open the Designer Launcher window. A list of Designers will be displayed.

3. Now let's create a shortcut on your desktop for your Designer. You have the option to **Create Shortcut**, or **Launch + Create a Shortcut** for that Designer. Click the dropdown box, select **Create Shortcut**, and a shortcut will be placed on your desktop.



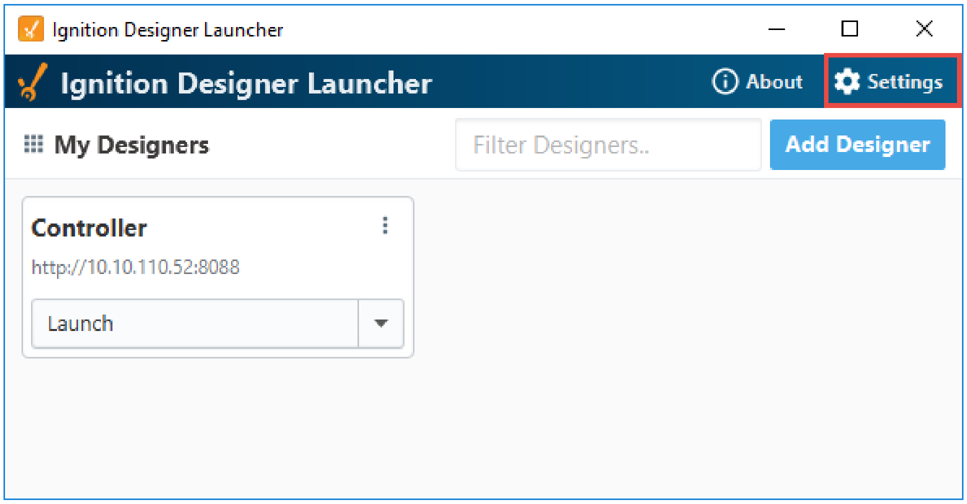
4. Go to your desktop and click the shortcut to open your Designer. You'll notice that name of your Designer is also the name of the shortcut (i.e., **Controller**).

The Designer Launcher will now notify users if it detects that an updated version of the launcher is available from either the app or a shortcut. Depending on the change, the launcher may allow you to upgrade at a later time, or require a new designer launcher before launching the application.

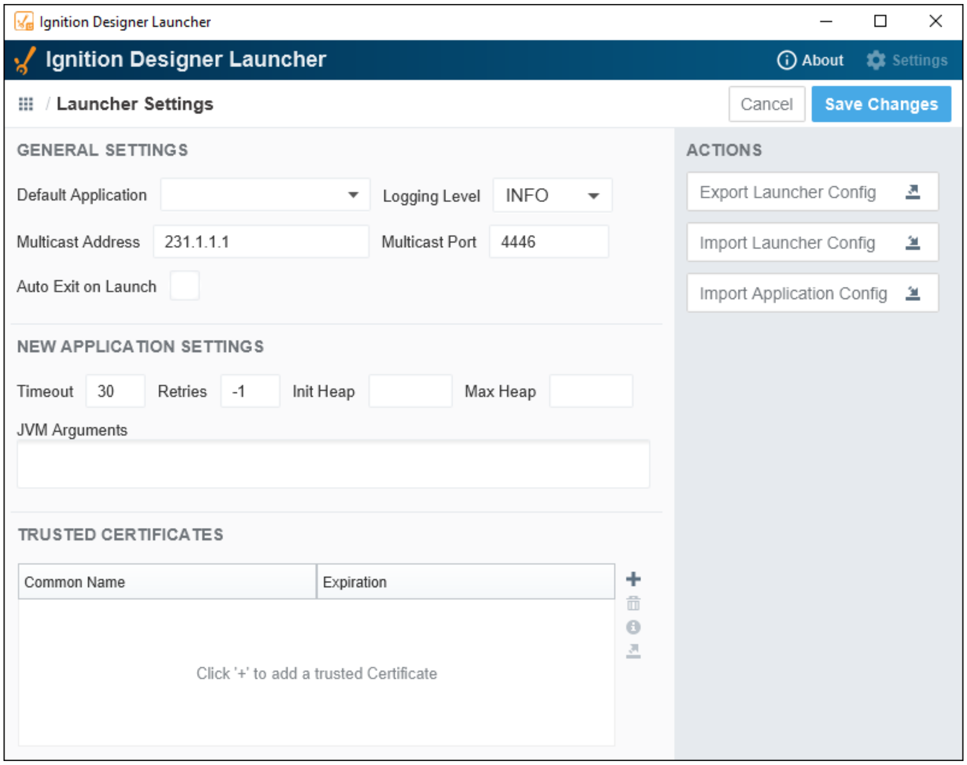
Designer Launcher Settings

You have the option of customizing your Designer Launcher Settings.

1. To access Designer Launcher Settings, click **Settings** on the top right of any Designer Launcher window.



2. When the Designer Launcher Settings window opens, you can choose to use the default settings or customize each Designer. When finished, click **Save Changes**.



GENERAL SETTINGS

Default Application:
Logging Level: INFO
Multicast Address: 231.1.1.1
Multicast Port: 4446
Auto Exit on Launch: □

NEW APPLICATION SETTINGS

Timeout: 30
Retries: -1
Init Heap:
Max Heap:
JVM Arguments:

TRUSTED CERTIFICATES

Common Name	Expiration

Click '+' to add a trusted Certificate

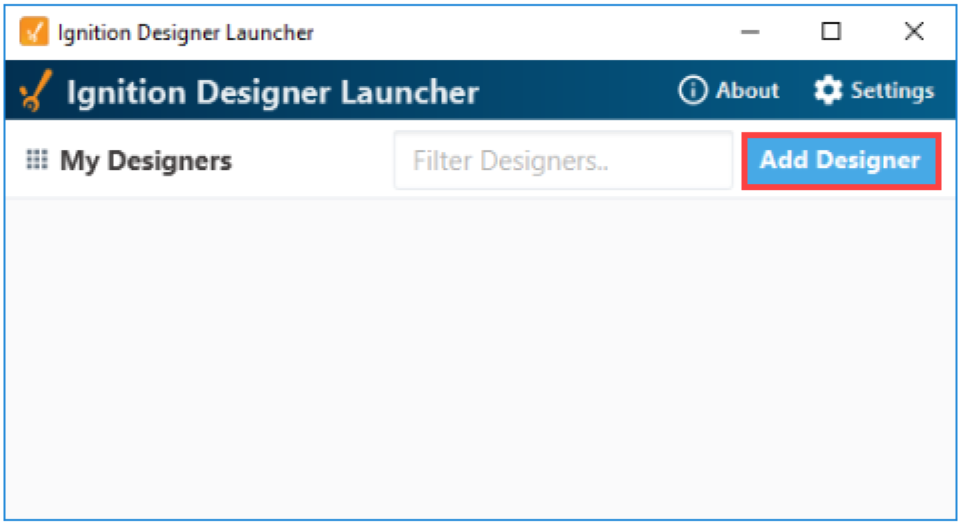
Launcher Settings - Configurations for the Launcher		
Property Setting	**Description**	**JSON Name**
Default Application	When set, the Launcher will attempt to automatically launch the specified Application when running. If left blank/null, an Application will not automatically start	`default.application`
Logging Level	Level of logging that will be used for the launcher: INFO, TRACE, DEBUG, WARN, and ERROR.	`logger.level`
Multicast Address	The address that will be used to listen for multicast broadcasts from Gateways.	`multicast.address`
Multicast Port	The port that will be used to listen for multicast broadcasts from Gateways.	`multicast.receive.port`
Auto Exit on Launch	Automatically close the launcher window when a client is started.	`autoexit`
Application Defaults - Default Configurations for New Applications		
Property Setting	**Description**	**JSON Name**
Timeout	Maximum number of seconds allowed for any Gateway communication. Any communication that exceeds this amount will cause the client launcher to abort the communication and try again if configured.	`timeout`
Retries	The number of times a Client will to attempt to contact a gateway again if an error occurred during communication. Available values are: • -1 : Retry indefinitely, or until the launcher is manually closed. • 0 : Zero retries, or abort after the first failure. • 1 (or more): Determines the number of retries: i.e., a value of "5" means five retries.	`retries`

Initial Heap	Amount of memory to be allocated at startup.	`init.heap`
Max Heap	Maximum amount of memory allocation for heap size.	`max.heap`
JVM Arguments	Arguments to append to client startup. These should start with a '-D' and be separated by semi-colons.	`jvm.arguments`

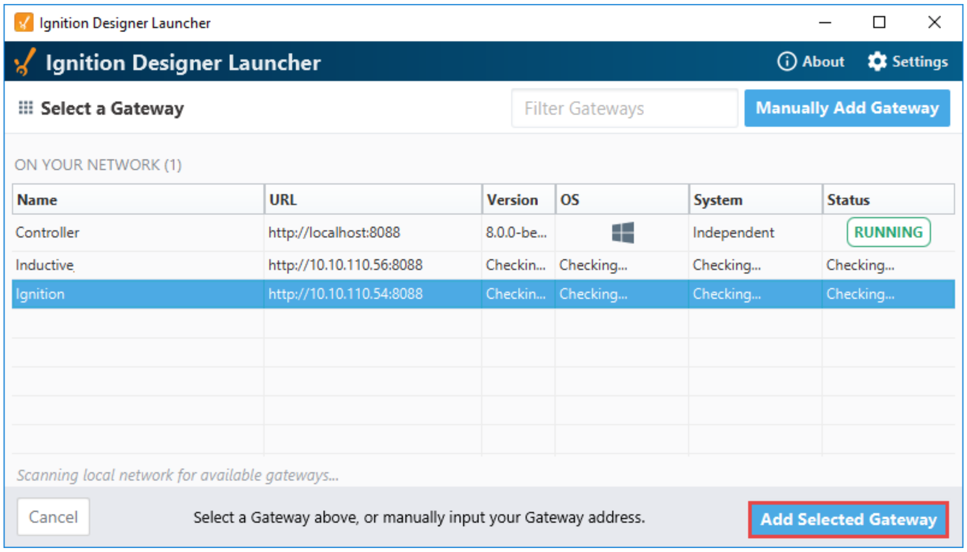
Adding Additional Designers

If you're adding a Designer for the first time or adding additional Designers, this process is the same since the Designer Launcher does all the work for you.

1. With the Designer Launcher open, click the **Add Designer** button.



2. The Designer Launcher will browse and list all the available Gateways on your local network as well as someone else's on the same network. Select from any of the available networks and press the **Add Selected Gateway** button. You can only select and add one Gateway at a time.

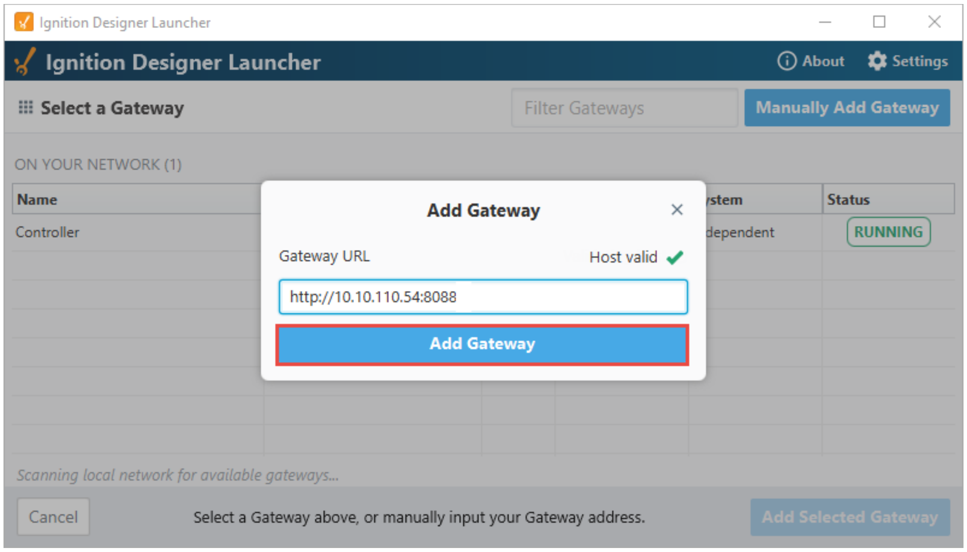


Name	URL	Version	OS	System	Status
Controller	http://localhost:8088	8.0.0-be...	⊞	Independent	RUNNING
Inductive	http://10.10.110.56:8088	Checkin...	Checking...	Checking...	Checking...
Ignition	http://10.10.110.54:8088	Checkin...	Checking...	Checking...	Checking...

Scanning local network for available gateways...

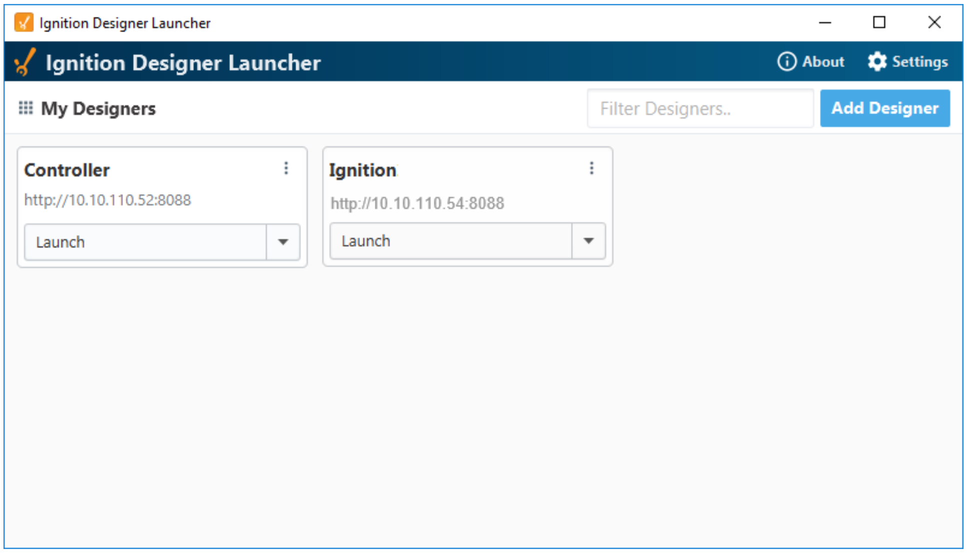
Select a Gateway above, or manually input your Gateway address.

3. If you don't see the Gateway you're looking for, but know the IP address, you can enter it manually. Click the **Manually Add Gateway** button in the upper right corner and enter the **Gateway URL**. It will inform you if you have a valid host name. If so, click **Add Gateway**.



Gateway URL: http://10.10.110.54:8088

4. Once it's added, it will appear your **My Designers** list.



Controller — http://10.10.110.52:8088

Ignition — http://10.10.110.54:8088

Designer Application Settings Reference

Setting Name	Description
Application Name	The name of the Designer application, as it appears in the launcher.
Timeout	The maximum number of seconds to allow for any gateway communication. Any communication that exceeds this amount will cause the Vision client launcher to abort the communication and try again if configured.
Retries	How many times to attempt to contact a gateway again if an error occurred during communication. Available values are: • -1 : Retry indefinitely, or until the launcher is manually closed. • 0 : Zero retries, or abort after the first failure. • 1 (or more): Determines the number of retries: i.e., a value of "5" means five retries. If the number of retries is exceeded, then the launcher will attempt to launch the Fallback Application.
Initial Heap	The initial heap size (memory) for the client.
Maximu	The maximum heap size (memory) for the client.

m Heap	
JVM Arguments	Additional JVM arguments to add to that specific application if desired. The defaults are created from the global jvm-args that exist at the time of the creation of the application. These can then be modified and the modifications will persist and never be updated again from the global setting.

Security Certificates

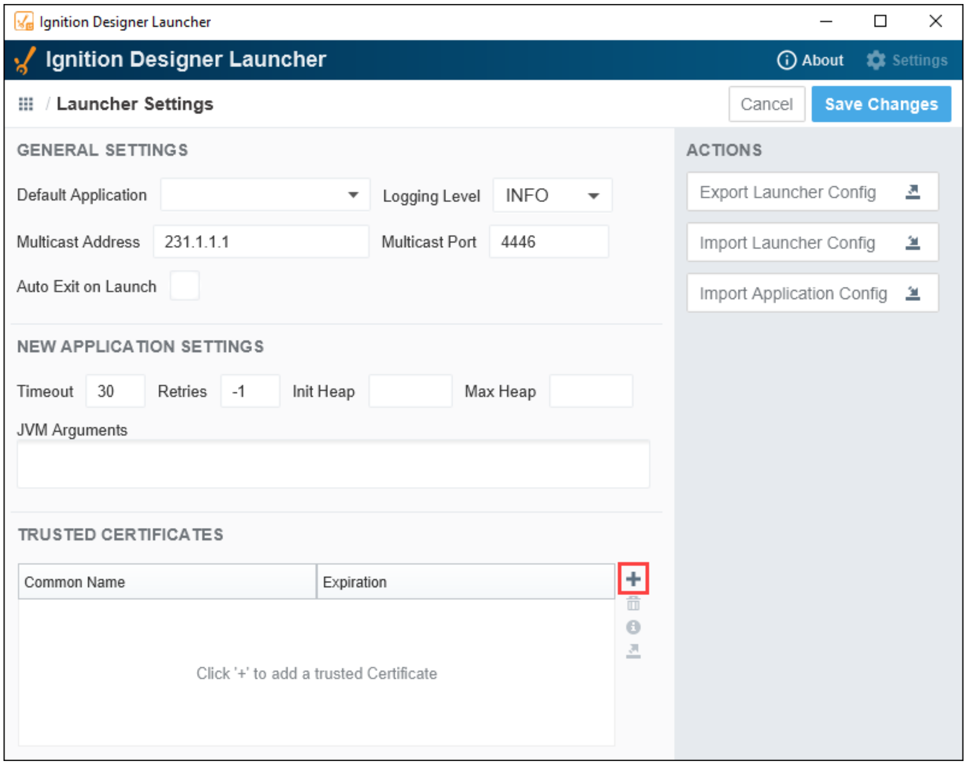
Trusted Certificates

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

As of release 8.0.13 you can add security certificates to the launcher. To utilize this feature, you must have installed a launcher from a version 8.0.13+ gateway.

While attempting to launch an application, the launcher will check against the application's host gateway, and attempt to automatically download any required certificates. In addition, certificates can be manually added using the example below.

1. Access Designer Launcher Settings by clicking **Settings** on the top right of any Designer Launcher window.
2. In the Trusted Certificates section, click the **Add** ✚ icon.



Ignition Designer Launcher

Launcher Settings

Cancel | Save Changes

GENERAL SETTINGS

Default Application:
Logging Level: INFO
Multicast Address: 231.1.1.1
Multicast Port: 4446
Auto Exit on Launch: □

NEW APPLICATION SETTINGS

Timeout: 30
Retries: -1
Init Heap:
Max Heap:
JVM Arguments:

TRUSTED CERTIFICATES

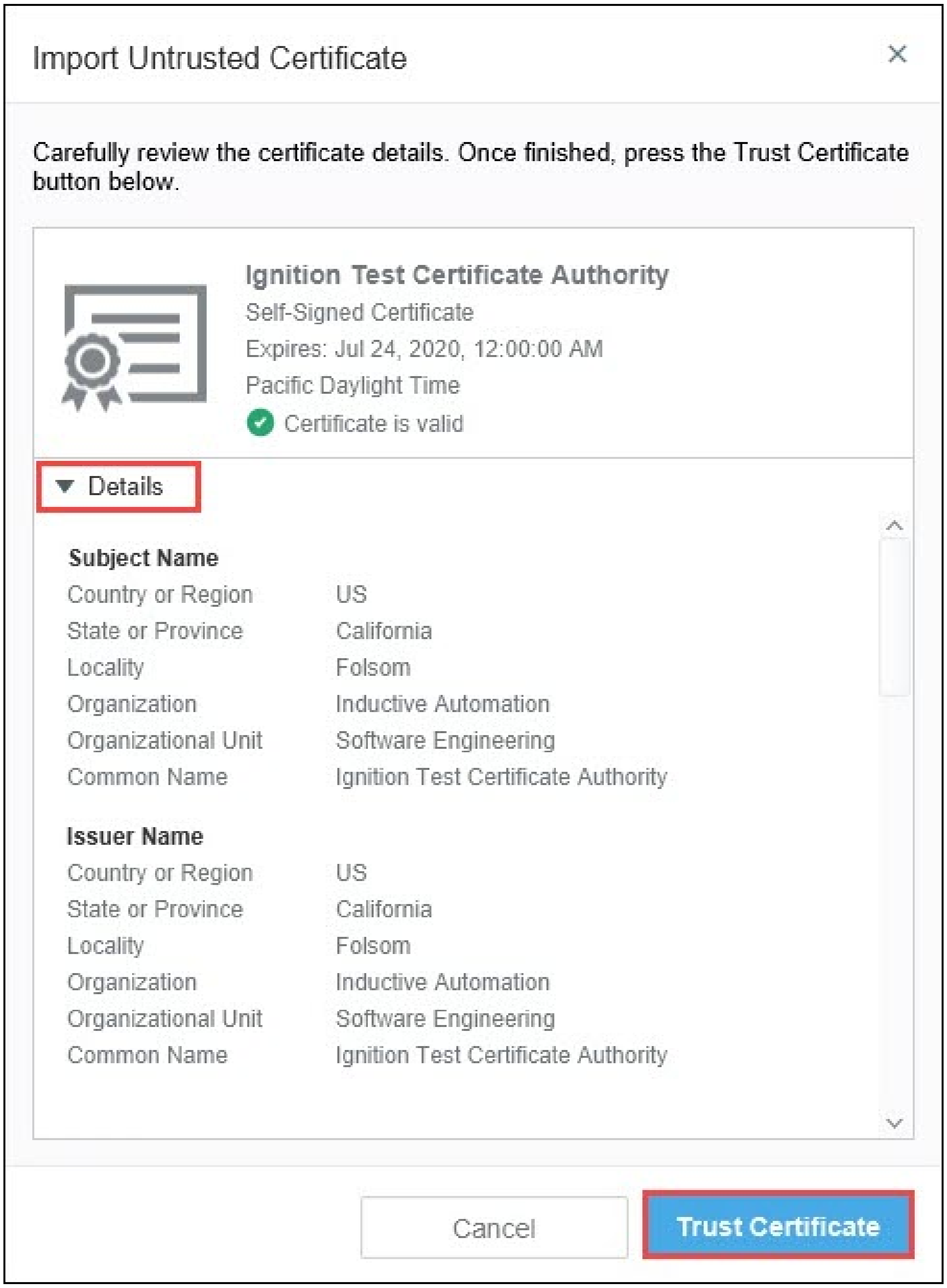
Common Name	Expiration

Click '+' to add a trusted Certificate

ACTIONS

Export Launcher Config
Import Launcher Config
Import Application Config

3. Navigate to the location of the certificate on your system and choose **Open**.
4. Information about the certificate is displayed. To view complete details, click the Expand ▶ icon next to Details. If everything looks OK, click **Trust Certificate**.



Import Untrusted Certificate

Carefully review the certificate details. Once finished, press the Trust Certificate button below.

Ignition Test Certificate Authority
Self-Signed Certificate
Expires: Jul 24, 2020, 12:00:00 AM
Pacific Daylight Time
✓ Certificate is valid

▼ Details

Subject Name

Country or Region	US
State or Province	California
Locality	Folsom
Organization	Inductive Automation
Organizational Unit	Software Engineering
Common Name	Ignition Test Certificate Authority

Issuer Name

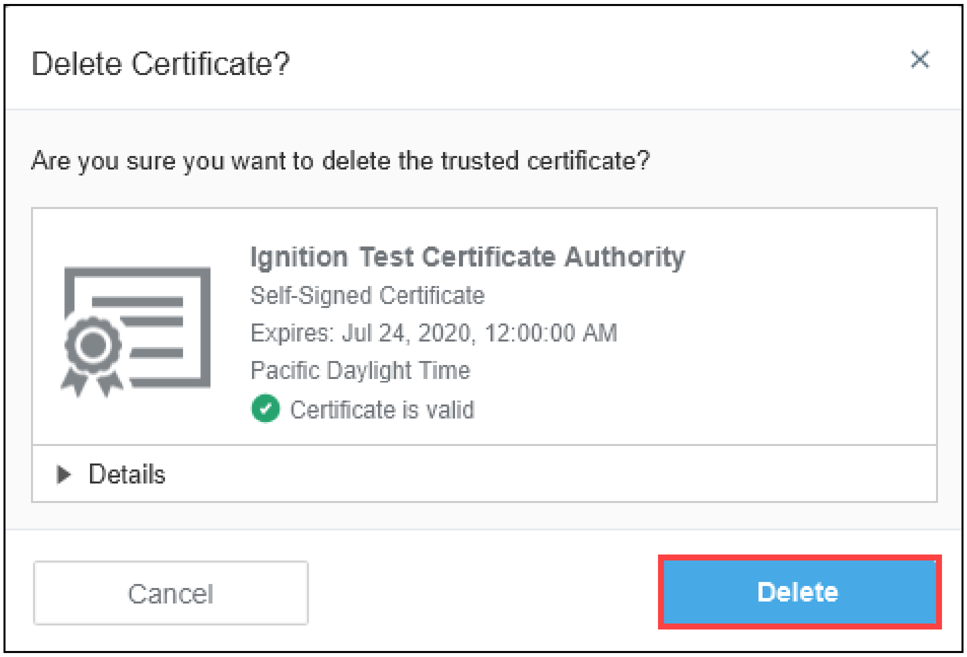
Country or Region	US
State or Province	California
Locality	Folsom
Organization	Inductive Automation
Organizational Unit	Software Engineering
Common Name	Ignition Test Certificate Authority

5. Once a certificate is imported, you can view its details by clicking the Information ℹ icon.

Delete a Certificate

1. To delete an imported certificate, access Designer Launcher Settings by clicking **Settings** on the top right of any Designer Launcher window.
2. In the Trusted Certificates section, select the certificate you want to delete.
3. Click the **Delete** 🗑 icon.

4. You'll get a confirmation window. If the certificate is the correct one you want to delete, click the Delete button.



Delete Certificate?

Are you sure you want to delete the trusted certificate?

Ignition Test Certificate Authority
Self-Signed Certificate
Expires: Jul 24, 2020, 12:00:00 AM
Pacific Daylight Time
✓ Certificate is valid

Export a Certificate

1. To export an imported certificate, access Designer Launcher Settings by clicking **Settings** on the top right of any Designer Launcher window.
2. In the Trusted Certificates section, select the certificate you want to export.
3. Click the **Export** icon.
4. Navigate to the location you want to save the certificate and click **Save**.

Self-Signed Certificates

When using a self-signed SSL certificate, the certificates for the gateway must be locally accessible to the client launcher in the following directory:

```
{user folder}\.ignition\clientlauncher-data\certificates
```

When a certificate is placed in the directory above, the launcher will attempt to automatically add the certificate(s) to the local KeyStore upon application launch.

Troubleshooting

If the Designer Launcher fails to launch a designer for some reason, the log file can be found at:

```
{user folder}\.ignition\clientlauncher-data\designer.log
```

This log should contain any errors that occurred.

Related Topics ...

- Designer
- Quick Start Guide

General Designer Interface

Designer Spaces

The Designer has a lot of panels and menus that allow you to build out a project tailored to your needs. However, while some of these like the File Menu are shared throughout the Designer, some of the panels and menu options are specific to certain objects and will typically only be displayed when an object of that type is selected. For example, when editing a Vision Window, the Designer has the Property Editor and Component Palette panels, but when editing an Alarm Pipeline, the Pipeline Blocks and Pipeline Block Editor panels are displayed instead. This creates different Designer Spaces that are used for different areas of a project.

There are many dockable and draggable panels that surround the workspace, as well as the familiar menu bars and toolbars. The dockable panels can be rearranged as you wish and will snap into place as you move them around the screen. Each workspace remembers its layout, which is the docking arrangement of the panels around it.

On this page

...

Docking System

The Designer's docking system provides a very flexible user interface allowing you to customize the layout as you wish. To rearrange the dockable panels, simply drag on their title bars. As you drag the panel, it will try to snap into place and show you a highlighted border. Use the highlighted border that appears to gauge where the panel will be moved to. Hold the CTRL key as you drag these panels to keep them from snapping into place. You can also drag these panels outside of the Designer or onto a second monitor.

Dockable panels can be in one of four modes:

1. **Docked** - The panel is visible, and located somewhere around the perimeter of the workspace. If two panels are docked in the same location, a tab strip will appear to switch between the two panels.
2. **Floating** - A panel can be dragged outside of the workspace perimeter to be floated. The panel can now be positioned anywhere on your desktop.
3. **Pinned** - Pinning a panel makes it minimize to one of the four sides of the Designer, represented by a small tab. Hover over the tab to use the panel.
4. **Hidden** - A hidden panel is not shown. You can open it again by selecting it in the **View > Panels** menu.

Toolbars can also be rearranged and floated to your liking. Simply drag on the "textured" left edge of the toolbar. If you re-arranged your panels into a layout that you don't like, you can quickly revert back to the default by selecting the **View > Reset Panels** option from the menu bar.

Expert Tip

Your docking preferences are stored under `%USER_HOME%/.ignition/*.layout`. If you really want to reset your preferences, remove these files and restart the Designer.

Project Browser

The **Project Browser** panel allows you to view the different Designer Spaces, and their component hierarchies at a glance. By clicking **Project Properties** ? Unknown Attachment icon you can view or change many of the properties settings in your project. You can expand the folders and navigate down through each folder to see elements of your project such as windows, views, containers, and components. The Project Browser shows the entire tree structure from the project level folder down to the component level. You can navigate within your project in two ways, by selecting nodes in the Project Browser tree or directly clicking on an element like a component in the project workspace. Regardless of how you select an element, properties for that element are displayed in the Property Editor.

? Unknown Attachment

Project Resources

All projects have resources, such as Vision Windows, Vision Templates, Perspective Views, project scripts, reports, and named queries. Project resources can be inherited from one project to another. For more information, see Project Inheritance.

Naming Conventions

Project resource names cannot be blank. They must start with a letter, a numeral, or an underscore. The following characters are reserved and cannot be used in names for project resources.

```
< (less than)
> (greater than)
: (colon)
" (double quote)
/ (forward slash)
\ (backslash)
| (vertical bar or pipe)
? (question mark)
* (asterisk)
```

In addition,the following names cannot be used as project resource names:

```
CON, PRN, AUX, and NUL
COM1, COM2, COM3, COM4, COM5, COM6, COM7, COM8, and COM9
LPT1, LPT2, LPT3, LPT4, LPT5, LPT6, LPT7, LPT8, and LPT9
```

Right-Click Menu

The **Project Browser** has basic functionality much the same as many applications that run on your PC. Functions can change depending on the type of project resource you are currently editing. When you right click on a project resource, several options are available:

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

Added a **"Close & Revert"** context menu item for project resource in the Project Browser. “Reverting” a modified resource will revert it back to its last saved state, assuming it's been saved before. Reverting new resources deletes them.

Function	Description
Close & Commit	Saves the updates made to the project resource and closes it.
Close & Revert	Reverts the project resource to its last saved state. Reverting an unsaved resource is the same as deleting it.
Configure View Permissions	View permissions determine whether or not a user has access to load the view.
Configure Events	Configures an event or action on a component.
Rename	Change the name of the current selection to a new name.
Cut	Cuts the current selection into the clipboard.
Copy	Copies the current selection into the clipboard.
Copy Path	Copies the path of the current selection into the clipboard.
Paste	Pastes the contents in the clipboard to the selected content.
Delete	Deletes the current selection.
Export	Exports resources from the project.
Protect	Protects the Project Resource. For more information, see Protecting Project Resources.

Tag Browser

The **Tag Browser** panel, located on the left side of the Designer workspace under the Project Browser, allows you to browse Tags in the Designer and OPC servers. Tags an be created, edited, displayed, exported, and imported directly from the Tag Browser. See Understanding Tags for more detailed information.

? Unknown Attachment

Menubar

This section addresses only the functionality that the Designer menubar shares between both the Perspective and Vision modules.

For information on the menubar tabs unique to the Vision module, see Vision Designer Interface.

For information on the menubar tabs unique to the Perspective module, see Perspective Designer Interface.

File Menu

The **File Menu** contains most of your basic options related to saving, similar to many other software applications on your PC.

? Unknown Attachment

Function	Description
New	The New menu option allows you to create a new Object. That Object could be a new window, a new template, or even a new project. It can vary depending on what modules you have installed, also providing the ability to make new transaction groups, reports, Sequential Function Charts, and more.
Open	Allows you to Open a previously created project.
Save All	Saves all project changes, but are not pushed out to the client.
Save	Save brings up a popup menu where you can choose what to save.
Save As	Saves the current project as a new project.
Update Project	Merges any changes from the Gateway to the project, if for example there were multiple people working on the same project.
Ignition Exchange	Provides access to resources, templates, and tools that can be shared with various other industries and used in your own Ignition projects.
Import	Allows you to import or export specific resources into or out of the project such as Tags, scripts, and templates, to name a few.
Export	Exports globally scoped resources, such as Alarm Pipelines.
Exit	Exits the project and allows you to save project changes.

Project Menu

Communication between the Designer and the Gateway is controlled from the Project Menu. The Designer offers three data communication modes for your projects: **Comm Off**, **Comm Read-Only**, or **Comm Read/Write**. Comm Read-Only is the default mode which does not allow writing from the Designer to Tag or Database sources.

? Unknown Attachment

About Communication Modes

Watch the Video

Function	Description
Comm Off	In this mode all database query traffic and Tag subscriptions and writes are blocked. Experts often use the **Comm Off** mode while designing a window to temporarily shut off data flow so that they can manipulate component's bound properties without the values being overwritten by the data bindings. This is useful to set the values that they want to serialize into the window. This can be important for windows with large datasets, clearing the datasets before saving the window can significantly reduce the size of the window, thus improving performance. ⚠ This setting does not affect the execution of a project's Transaction Groups because Transaction Groups execute on the Gateway, not in the Designer. See also: Communication Modes
Comm Read-Only (default)	Tag subscriptions and SELECT queries work, but Tag writes and UPDATE/INSERT/DELETE queries are blocked. The Designer starts up in **Comm Read-Only** mode as a safety mechanism so that you don't inadvertently write to a Tag as you are designing. You can customize the Designer's startup mode, see the Designer/General Properties section.
Comm Read-Only (default)	All data is passed through to the Gateway. The mode can be switched at any time via the tri-state toggle selection in the main toolbar, or the radio buttons in the Project menu. A common beginner mistake is to forget to switch the mode to **Comm Read/Write** when attempting to test a window's functionality in **Preview** mode.
Properties	Opens up the Project Properties window, allowing project settings to be changed. See also: Project Properties
Event Scripts	Opens up the appropriate event script window, either client, session, or Gateway. These can also be accessed from the Project Browser. See also: Client Event Scripts and Gateway Event Scripts.
Preview Mode	Puts the Designer into Preview Mode, allowing you to interact with it like a client. See also: 26018032
Preview Language	Determines the language that the Designer will revert to when in Preview Mode. See also Localization and Languages and Localization in Vision.

To Set or Change the Communication Mode

There are three ways to set or change the comm mode in the Designer:

- Using the **Project** menu.
- The buttons located on the **Main Toolbar.**
- Using the **Project Properties** window.

Setting the Comm Mode from the Project Menu

On the Designer toolbar, go to the **Project** menu and choose: **Comm Off**, **Comm Read-Only**, or **Comm Read/Write**.

? Unknown Attachment

Setting the Comm Mode from the Main Toolbar

If the Main Toolbar is enabled, go to **Project**, and you'll see a corresponding button for each Comm Mode. The currently selected Comm Mode will have its button highlighted in gray.

? Unknown Attachment

Setting the Comm Mode from the Project Propertied Window

This will set the default Comm Mode that the Designer starts up in for the current project.

1. In the Designer, either double-click on the **Project > Properties** node in **Project Browser**, or click on the **Project > Properties** command on the top menu.
   The **Project Properties** window is displayed.
2. Go to **Project > Designer.**
3. Under **Startup Options**, for the **Initial Gateway Comm Mode**, choose from the dropdown **Comm Off**, **Comm Ready-only**, or **Comm Read/Write.**

   ⚠ These property settings are saved on a per-project basis.

   ? Unknown Attachment

Communication Error Message

If a user is attempting to write to a project in the Designer or from a Client, and the Comm Mode is not enabled for Read/Write, a dialog box will popup stating the Designer is not in Read/Write Mode.

? Unknown Attachment

If your Tag is not being written too, there are a couple places to check:

- From Main Toolbar go to **Project,** and make sure the Comm Mode is set to Read/Write.
- If Comm Read/Write is checked, you may have a Client Event Script enabled that could be preventing you from writing to a project or a Tag, as shown in the error message example below. You may need to edit your Client Event Script.

  ? Unknown Attachment

Previewing the Project

IU INDUCTIVE UNIVERSITY

Previewing the Project

Watch the Video

Many times, it is useful to test the components on the screen to ensure that certain bindings or scripts are working the way that was intended. The Designer can go into **Preview Mode** that will allow you to interact with the currently opened window as though you were working in a client. This means that instead of clicking between components and seeing their properties, you will be able to interact directly with the components such as clicking on a button to execute its script, or entering a value into a text field to update a Tag value.

? Unknown Attachment

Preview Mode

The window workspace operates in two distinct modes: Design Mode and Preview Mode. There are three different ways you can switch between Design and Preview Mode:

- One of the easiest ways is from the Main Toolbar using the **Preview** icon or the **Design** icon to switch between modes.

- From the Main Toolbar using the **Project > Preview Mode** menu item.
- Using the **F5** key to toggle between the two modes.

In Design Mode, your mouse is used to manipulate components in a window. You can select, drag, and resize them. You can alter data bindings and event script configuration. Data bindings are active in Design mode, but event handlers are not.

In Preview Mode, you are interacting with a "live" version of the window. Property bindings and event handlers will run, just like in the Client.

⚠ Some of Ignition's functionality will not work in Preview Mode, for example, the retarget and openWindowInstance scripting functions must be tested in a Client.

Preview Mode is useful for a quick check of the operation of a window, but it becomes cumbersome when trying to test a whole project. For that, we recommend having a launched Client up as well, and doing testing in the true Client. You can quickly launch a client in one of the following two launch modes via the **Tools > Launch Project** menu.

Tools Menu

The **Tools Menu** provides some tools to help you when creating projects.

Each of the tools are described here.

Function	Description
Console	The Output Console is a dockable panel prints system messages that are coming from the designer. This can vary from simple info messages that show when things are loaded, to error messages when something goes wrong. The Console is also used frequently to test and debug Python scripts, as print statements on components that are run in the designer are printed here.
Image Management	The **Image Manager** is available from the **Tools > Image Management** menu. This tool is a drag-and-drop browser that helps manage the images that are stored on the Gateway. It is important to realize that these images are shared across all projects: they are not stored inside a project itself. Use the Image Management tool to do common tasks like uploading new images and creating folders. You can drag images and folders from your computer's desktop or hard drive into this window to easily upload new images to the Gateway. You can also get to this tool by putting an **Component Palette > Display > Image** component on a window, and clicking the browse button on the image's **Image Path** property. See Images and SVGs in Vision and Images, SVGs, and Icons in Perspective.
Script Console	Opens up the Script Console. window. This is where you can test scripts out.
Database Query Browser	Opens up the Database Query Browser panel, which allows you to run SQL queries against your database connections.
Translation Manager	Opens up the Translation Manager panel, which allows you to configure translations. See Localization and Languages.
Symbol Factory	If you have the Symbol Factory module installed, you'll be able to open the Symbol Factory browser under the Tools menu in the Designer. You can browse through the symbols or use the convenient search function to find the symbol you need.

	Once you find a symbol, you can drag-and-drop it into a window. Each symbol is dropped as a shape group. You will be able to un-group it or double-click into the group just as if you had drawn the symbol yourself using fundamental shapes. This means that you can alter the shape if you need to, or bind any colors inside the shape to a Tag to make the shape dynamic.
Launch Project	Allows you to launch the project directly from the Designer.

Help Menu

The **Help Menu** provides online assistance when looking for information or troubleshooting an issue.

? Unknown Attachment

Functions	Description
Help	Opens up your web browser and takes you to this User Manual for quick reference.
Diagnostics	The **Help** menu in the Designer and the Vision Client has a **Diagnostics** window that contains a number of tabs each providing a useful troubleshooting feature. You can right-click on any of the tabs to show or hide the other tabs. For more information on these tabs and the troubleshooting features, go to Designer Diagnostics.
About Ignition Designer	Provides information about the Designer such as the versions of the modules, Java version, and Gateway IP Address that the Designer is using.

Related Topics ...

- Creating Tags
- Vision Designer Interface
- Perspective Designer Interface

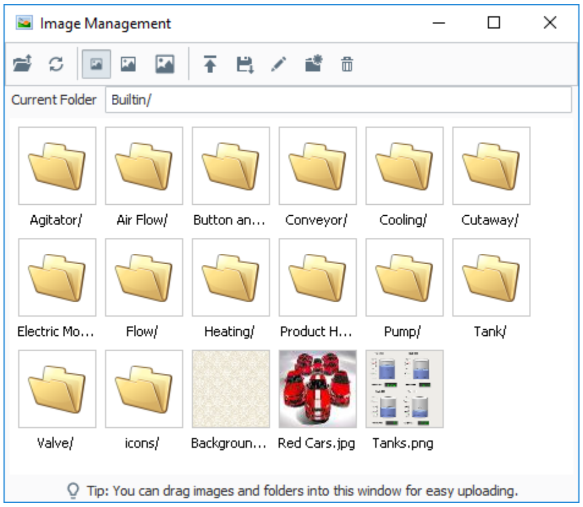
Designer Tools

The Designer comes with many tools that allow you to manage and test various resources within a project. Each of the tools have their own interface and are accessed within the Tools menu on the menu bar of the Designer.

On this page ...

Image Management Tool

The Image Management Tool allows you to manage the images that are stored within the Ignition Gateway. The path to the images can be copied out and pasted into a component's property that is expecting an image path.

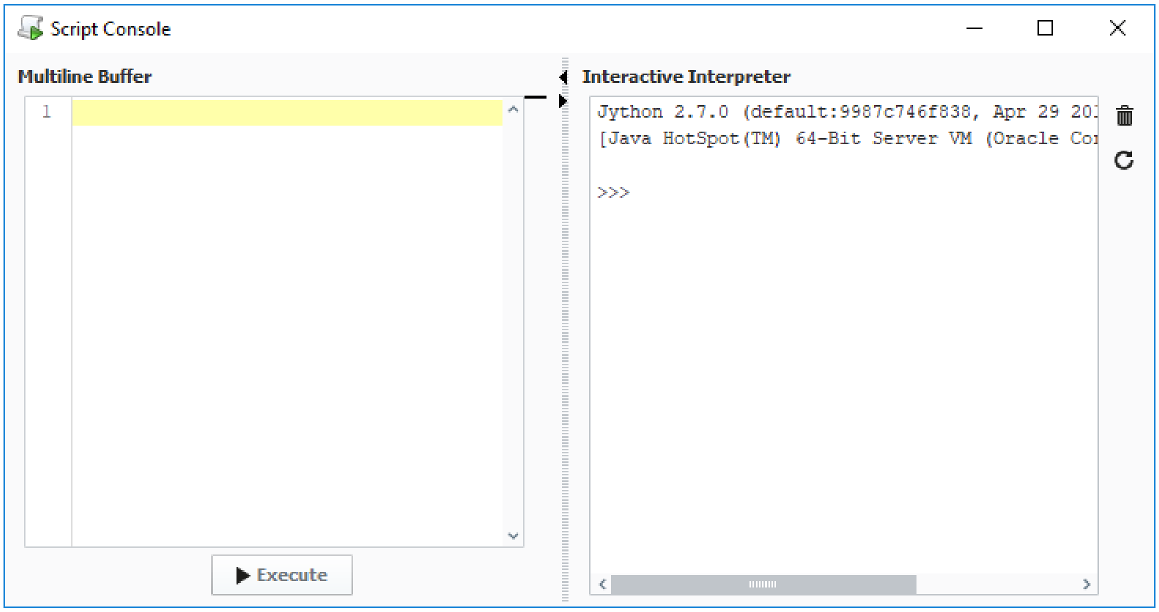


Current Folder: Builtin/

Tip: You can drag images and folders into this window for easy uploading.

Script Console

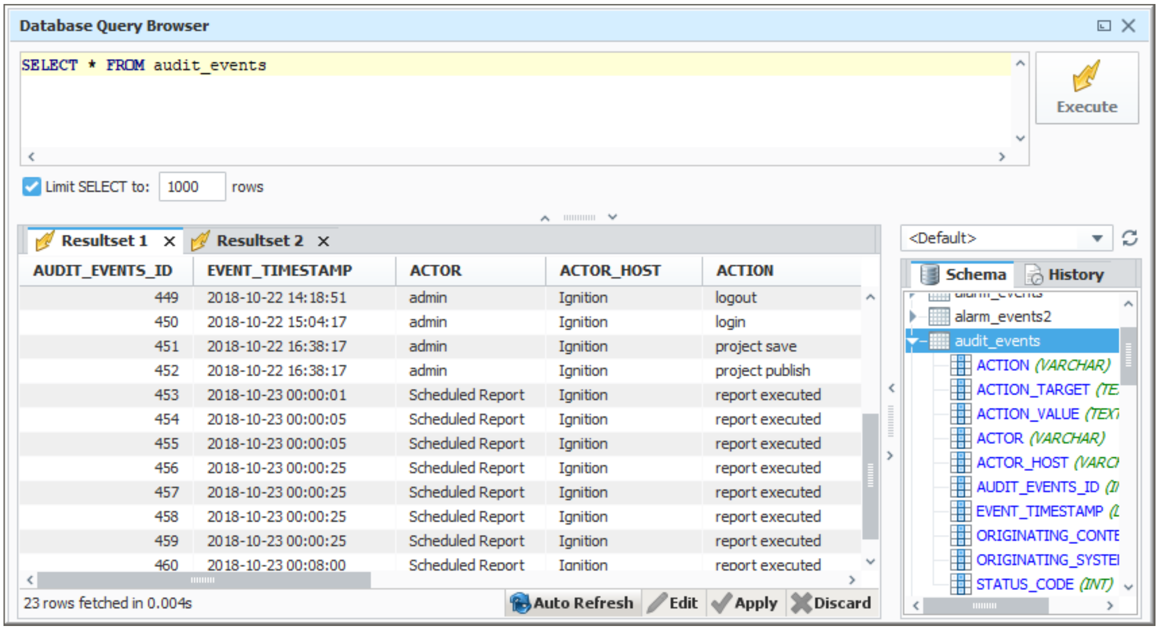
The Script Console allows you to test out code snippets, printing the results out in the panel on the right.



```
Jython 2.7.0 (default:9987c746f838, Apr 29 20
[Java HotSpot(TM) 64-Bit Server VM (Oracle Co

>>>
```

Database Query Browser

The Database Query Browser is like the Script Console, but for SQL and databases. Here, you can test out queries to ensure you are returning the correct data from your database.



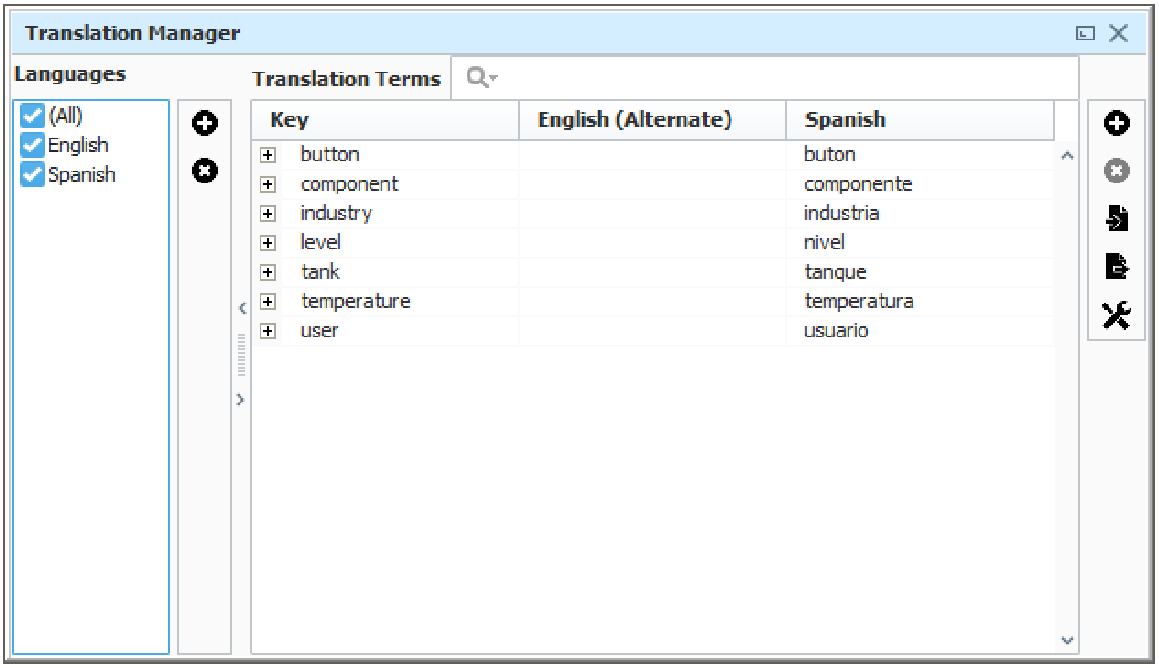
```
SELECT * FROM audit_events
```

AUDIT_EVENTS_ID	EVENT_TIMESTAMP	ACTOR	ACTOR_HOST	ACTION
449	2018-10-22 14:18:51	admin	Ignition	logout
450	2018-10-22 15:04:17	admin	Ignition	login
451	2018-10-22 16:38:17	admin	Ignition	project save
452	2018-10-22 16:38:17	admin	Ignition	project publish
453	2018-10-23 00:00:01	Scheduled Report	Ignition	report executed
454	2018-10-23 00:00:05	Scheduled Report	Ignition	report executed
455	2018-10-23 00:00:05	Scheduled Report	Ignition	report executed
456	2018-10-23 00:00:25	Scheduled Report	Ignition	report executed
457	2018-10-23 00:00:25	Scheduled Report	Ignition	report executed
458	2018-10-23 00:00:25	Scheduled Report	Ignition	report executed
459	2018-10-23 00:00:25	Scheduled Report	Ignition	report executed
460	2018-10-23 00:08:00	Scheduled Report	Ignition	report executed

23 rows fetched in 0.004s

Translation Manager

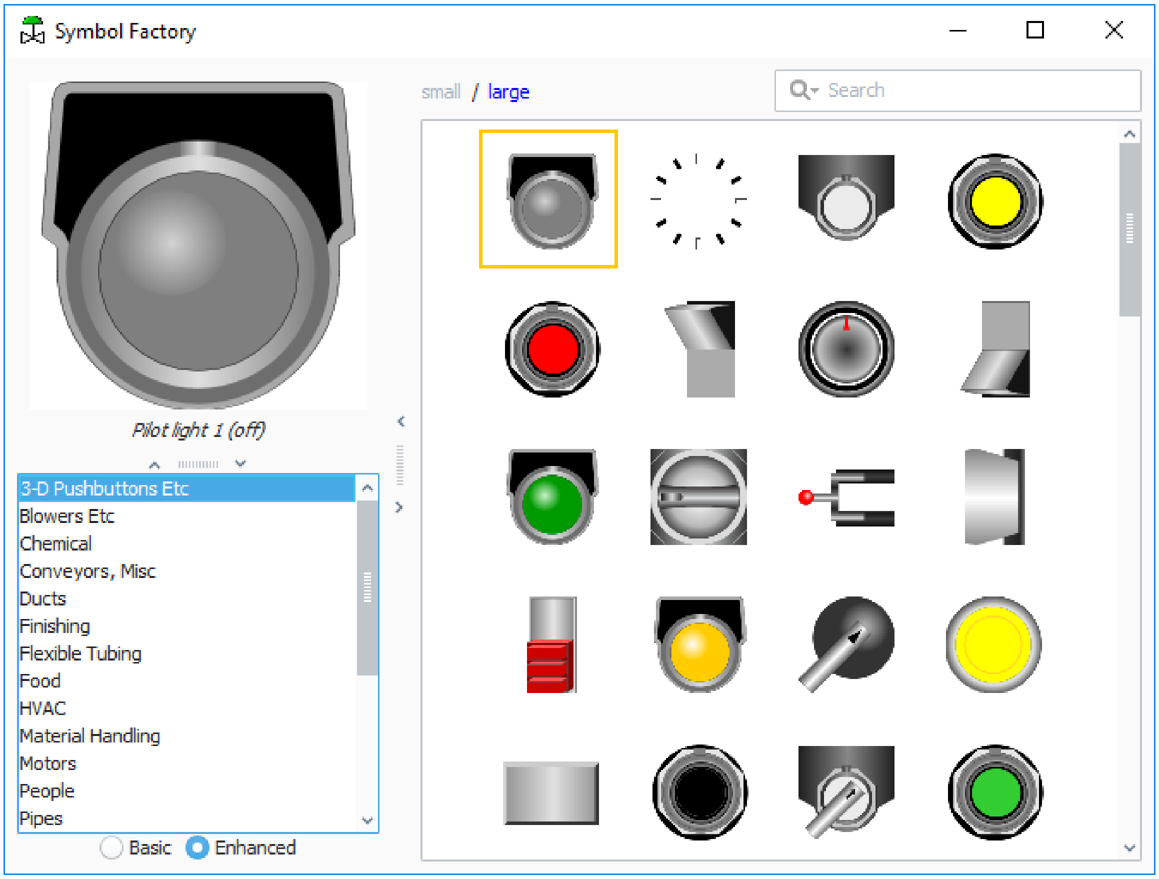
The Translation Manager allows you to add, edit, and remove translation mappings to your system. Works in conjunction with the Localization system.



Key	English (Alternate)	Spanish
button		buton
component		componente
industry		industria
level		nivel
tank		tanque
temperature		temperatura
user		usuario

Symbol Factory

Symbol Factory module is a unique designer tool included with Vision or Perspective.



Related Topics ...

- Images and SVGs in Vision
- Localization and Languages

In This Section ...

Image Management Tool

Images such as PNGs, JPGs, GIFs, and SVGs can be uploaded to the Image Management Tool and used inside of windows in Ignition. Once uploaded, these images may be used on windows and in templates. The Image Manager tool, available from the **Tools** > **Image Management**, provides an interface to upload, download, or select images.

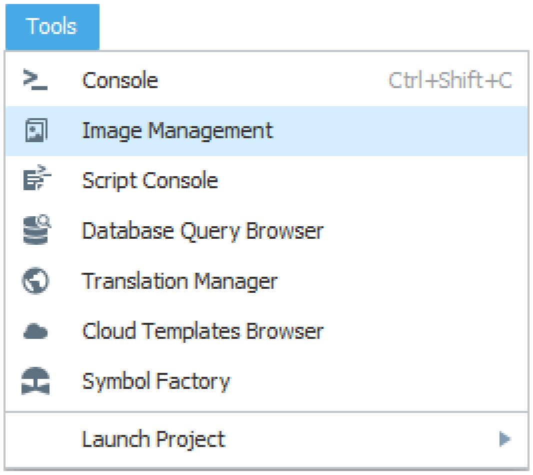
On this page ...

Bitmap file types are not supported

The Image Manager process does not support bitmap files.

The following feature is new in Ignition version **8.0.4**
Click here to check out the other new features

SVGs are now supported in the Image Management Tool.



INDUCTIVE UNIVERSITY

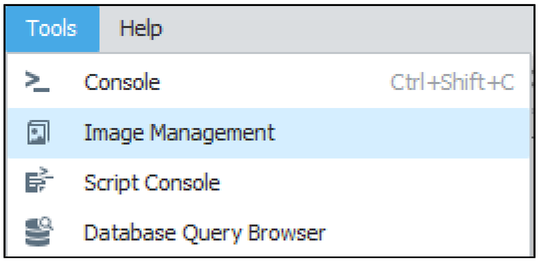
Images (png, jpg, gif)

Watch the Video

Using Images in the Designer

Uploading an Image to the Image Management Tool

There are two ways to upload an image into the Image Management tool. Both ways involve having the Image Management tool open. At the top of the Designer in the Menu Bar, select **Tools > Image Management.**

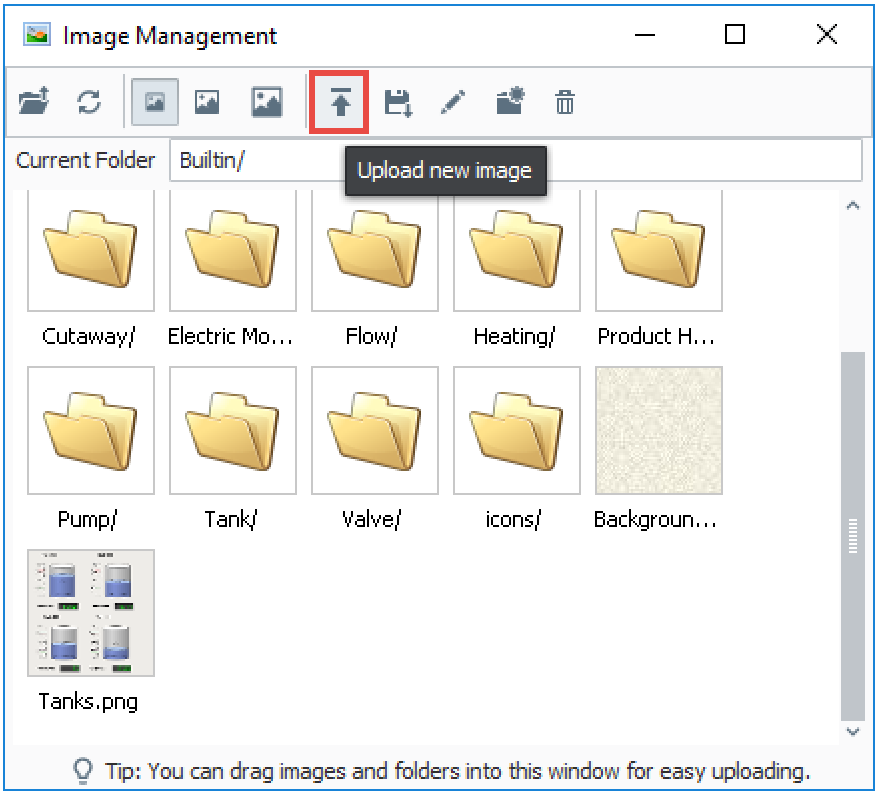


Upload on Drag and Drop

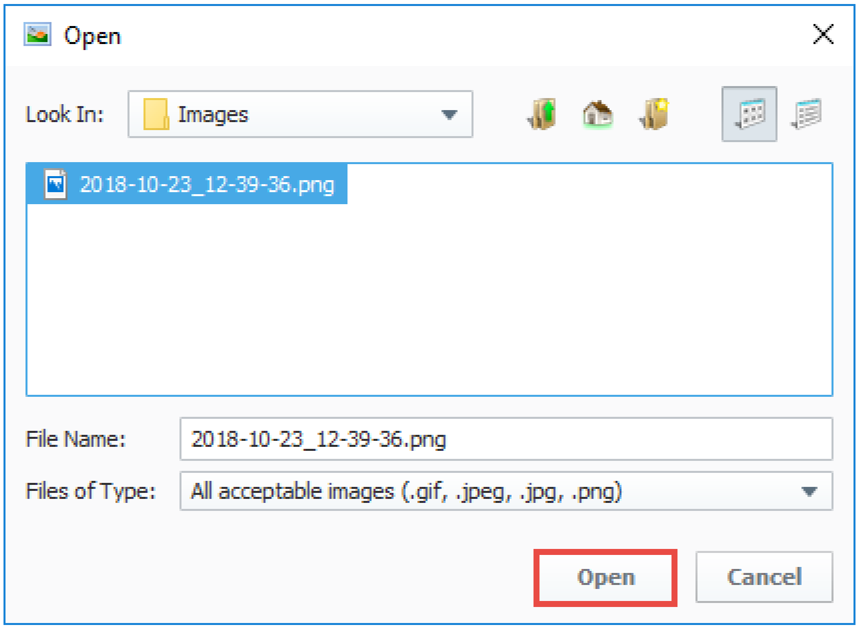
Images can simply be dragged and dropped from the local file system into the **Image Management** window.

Manual Upload

Alternatively, the Image Management window has an Upload button to pass images in. Locate the directory you wish to upload the image. You may use the root folder, or create a new folder to keep your images organized. When ready, click on the **Upload Image** button.



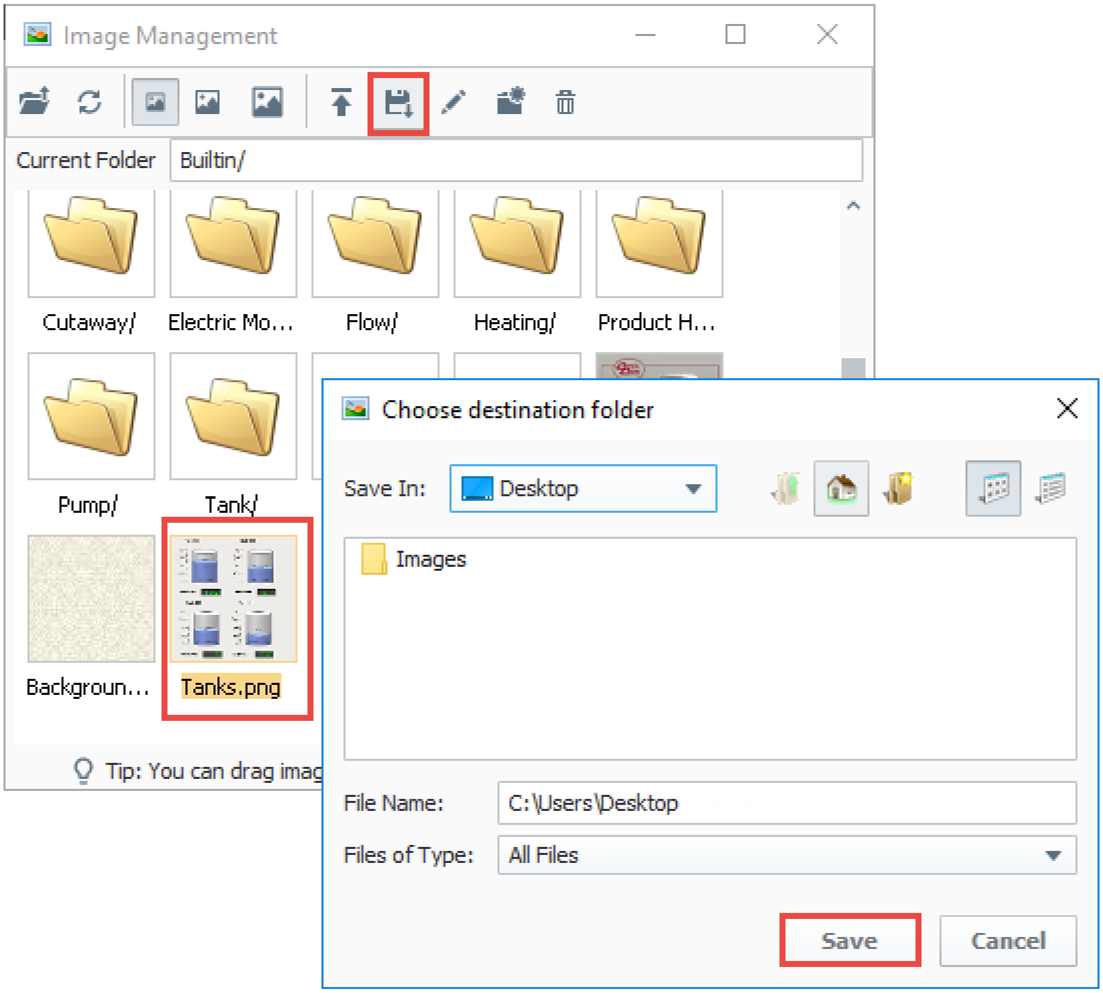
An **Open** dialog window will appear. Simply find your image on the local system, and click **Open** to upload the selected image.



Downloading Images from the Image Management Tool

Single images, as well as entire directories, may be downloaded from the Image Management tool. This is useful when migrating a project to another Gateway.

Image downloads can be taken from either the **Image Management** or **Image Browser** windows. If at least one folder or image is selected, the **Save** button will become enabled. Click the **Save** button, pick a local directory to save the images to, and click the **Save** button again. Note that all images and subfolders in the selected folder will be copied to the selected directory.



Current Folder: Builtin/

Save In: Desktop

File Name: C:\Users\Desktop

Files of Type: All Files

Exporting and Importing Images in Projects

Uploading an image involves storing the file in the Gateway's internal database. This means that project exports do not contain any referenced images.

When exporting a project for use in another Gateway, it is recommended to also export any images that the project uses, and upload them into the new Gateway at the same directory. Examples of uploading and downloading images can be found on this page.

Related Topics ...

- Images, SVGs, and Icons in Perspective
- Images and SVGs in Vision
- Using Images
- Symbol Factory

Script Console

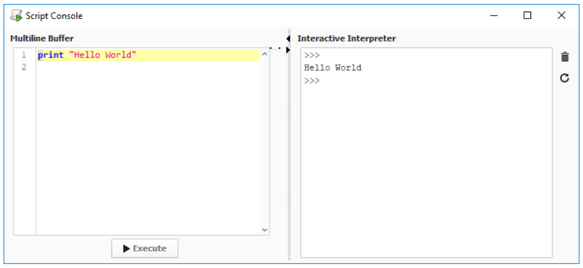
The Script Console is a live Python terminal that is only accessible in the Designer. It is a great way to very quickly test a script as it does not rest on a scripting event or specific component. The Script Console can be opened via the **Tools > Script Console** menu. It consists of two parts: a **Multiline Buffer**, and an **Interactive Interpreter**. Code can be typed into both sides.

Due to how "scope" works, the Script Console can not interact with components on a window, but it can call Project and Shared scripts. If a Project or Shared script was recently added, then the console will need to be reset before it can be called. This can be accomplished by clicking on the Reset ↻ icon in the upper right.

> ⚠ Gateway scoped information will not appear in either the Script Console or Output Console. Instead, the output will be sent to the wrapper.log file. Alternatively, system.util.getLogger() will send messages to the Gateway Console, and is a preferred method of troubleshooting Gateway scoped scripts.

Scripting Console

Watch the Video

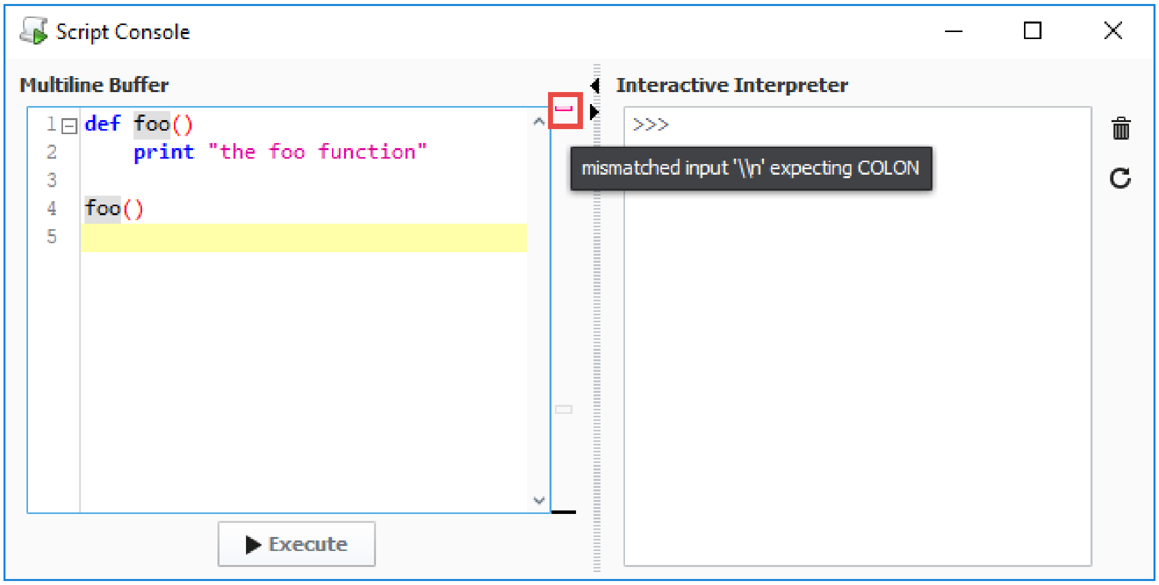


```
print "Hello World"
```

```
>>>
Hello World
>>>
```

Features

There are several icons and UI elements located on the Script Console window. The image below highlights a syntax error.



```
def foo()
    print "the foo function"

foo()

```

```
>>>
```

mismatched input '\\n' expecting COLON

A reference of the icons and descriptions are found in the table below:

Icon	Name	Description
	Clear	Clears the text from the Interactive Interpreter.

🗑		
↻	Reset	Clears and resets the text, and deletes all user defined objects (variables and functions) from the Interactive Interpreter.
◀ ▶	Expand/Collapse	Expands / Collapses Multiline Buffer and Interactive Interpreter.
⎵	Syntax Error Highlight	Shows up next to a line in the Multiline Buffer identifying an error. Hover over the Error Symbol to see information on the exception.

Multiline Buffer

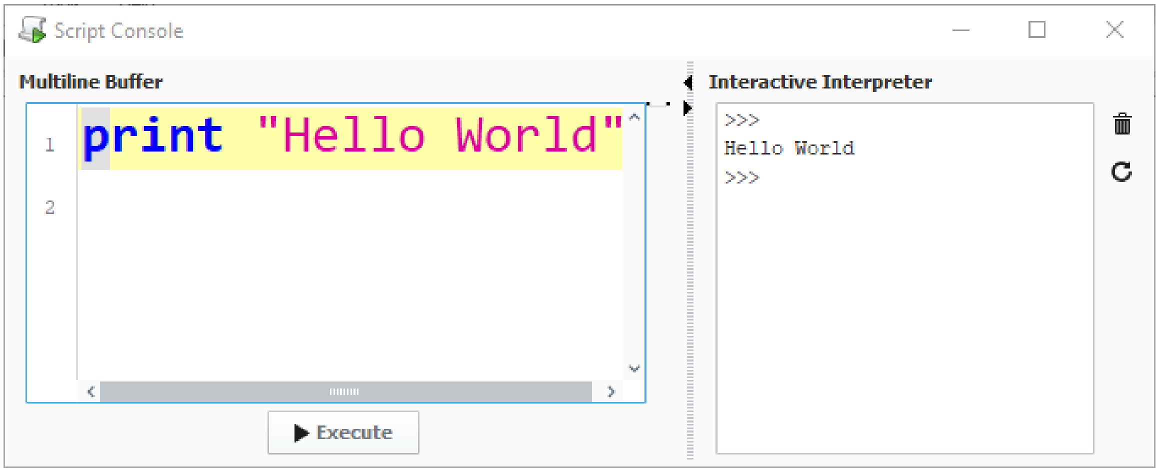
The Multiline Buffer, located on the left side of the Script Console, allows for multiple lines of code to be entered and then executed by clicking on the ▶ Execute button. All **print** statements will output to the Interactive Interpreter. It also supports code folding for function definitions and comments.

> The following feature is new in Ignition version **8.0.4**
> Click here to check out the other new features

When executing a script in the console, the ▶ Execute button will change to an ❚ Interrupt button. Developers can press the **Interrupt** button to interrupt / stop a script from executing when testing code with a lot of data, or when the script inadvertently gets stuck in an infinite loop.

Font Size Adjustment

Font size in the Multiline Buffer can be adjusted by holding **Ctrl** and scrolling the mouse wheel.

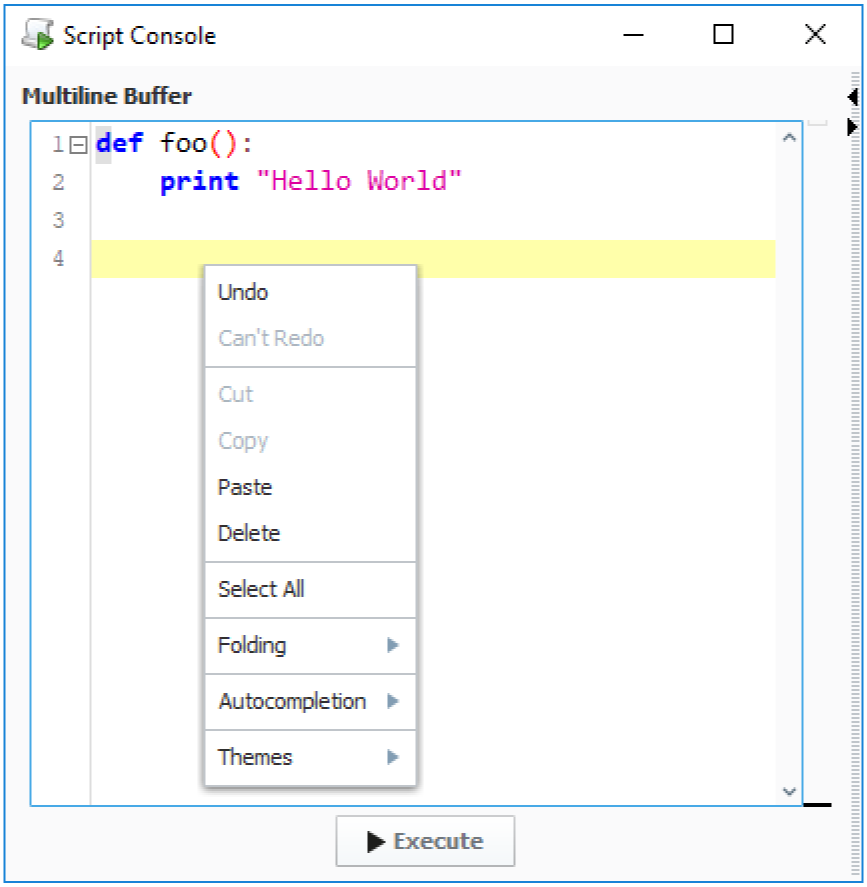


```
print "Hello World"
```

```
>>>
Hello World
>>>
```

Right Click Menu

Right clicking on the Multiline Buffer opens a menu. The menu options are described in the table below.



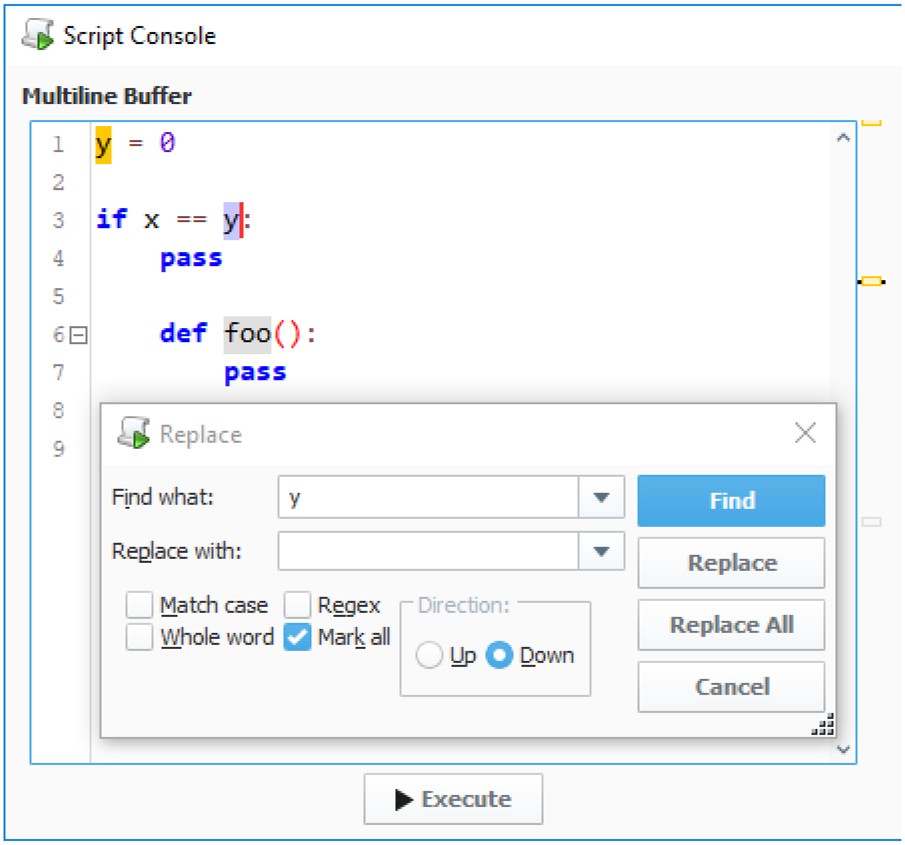
```
def foo():
    print "Hello World"
```

Main Menu Options	
Name	**Description**
Undo	Undoes the last action.
Redo	Gets rid of the last undo action.
Cut	Cuts the selected text.
Copy	Copies the selected text.
Paste	Pastes the selected text.
Delete	Deletes the selected text.
Select All	Selects all text in the window.
Folding	
Name	**Description**
Toggle Current Fold	Expand or collapse the fold where the text cursor is located.
Collapse All Comments	Collapse all instances of multi-line comments. Only contiguous comments are collapsible.
Collapse All Folds	Collapse all expanded folds.
Expand	Expand all collapsed folds.

All Folds	
Autocompletion	
Name	**Description**
Automatic Activation	Determines if the Autocompletion window to automatically appear. When set to true, the window will appear after a second of inactivity once at least "system." has been typed. When set to false, the window can still be brought up manually by pressing **Ctrl+Space**.
Show Description Pane	Determines if the Description pane should appear in the Autocompletion window.
Themes	
Name	**Description**
Default	Uses the default theme with light background, dark characters.
Dark	Uses a dark theme with dark background, light characters.
Visual Studio	Uses a theme reminiscent of Visual Studio. Light background, dark characters.

Find/Replace

Pressing **Ctrl+R** while the text cursor is in the Multiline Buffer opens a Find and Replace window. This will search for instances of text throughout the Multiline Buffer, and allows the user to replace all or some instances with new text.



```
y = 0

if x == y:
    pass

    def foo():
        pass
```

Find what: y

Replace with:

□ Match case □ Regex

□ Whole word ☑ Mark all

Direction: ○ Up ● Down

Keyboard Shortcuts

The following shortcuts apply only to the Multiline Buffer.

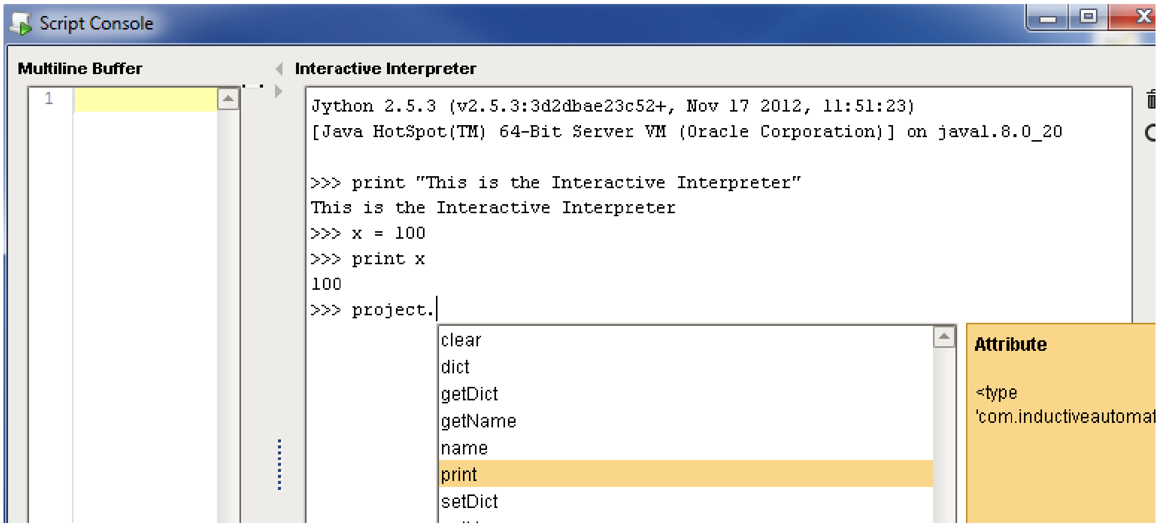
Key(s)	Description

Ctrl + </> (on the number pad)	Collapse all folds.
Ctrl + <*> (on the number pad)	Expand all folds.
Ctrl + <-> (on the number pad)	Collapse the fold on the same line as the text cursor.
Ctrl + <+> (on the number pad)	Expand the fold on the same line as the text cursor.
Ctrl + <Space>	Open Autocompletion window. By default, the window will automatically open once "**system.**" has been typed.
Ctrl + <R>	Open Find/Replace window.
Ctrl + <Mouse Wheel Scroll>	Increase and decrease the font size.
Ctrl + <Enter>	Executes the script in the script editor.

Interactive Interpreter

The Interactive Interpreter is located on the right side of the Script Console, and allows you to run a single line of code at a time. Code is executed from the Interactive Interpreter by pressing the **Ente**r key. **Print** statements from both sides of the Script Console will appear in the Interactive Interpreter.

The Autocompletion window, available in the Interactive Interpreter, has access to the current working environment so items such as Project and Shared scripts will automatically appear. They can also be typed in manually.



```
Jython 2.5.3 (v2.5.3:3d2dbae23c52+, Nov 17 2012, 11:51:23)
[Java HotSpot(TM) 64-Bit Server VM (Oracle Corporation)] on java1.8.0_20

>>> print "This is the Interactive Interpreter"
This is the Interactive Interpreter
>>> x = 100
>>> print x
100
>>> project.
```

Keyboard Shortcuts

The following shortcuts apply only to the Interactive Interpreter

Key(s)	Description
Ctrl + <L>	Clear the Interactive Interpreter. Functionally the same as clicking the Clear button.
Ctrl + <R>	Reset the Interactive Interpreter. Functionally the same as clicking the Reset button.
Up arrow	Cycle backward through command history.
Down arrow	Cycle forward through command history.

Ctrl + <C>	Keyboard interrupt.
Ctrl + <Space>	Open Autocompletion window. By default, the window will automatically open once an "**object.**" has been typed such as "system" or "project," and a Project script has already been defined.
Ctrl + <A>	Move the text cursor to the start of the line. Similar to pressing the Home key.

Related Topics ...

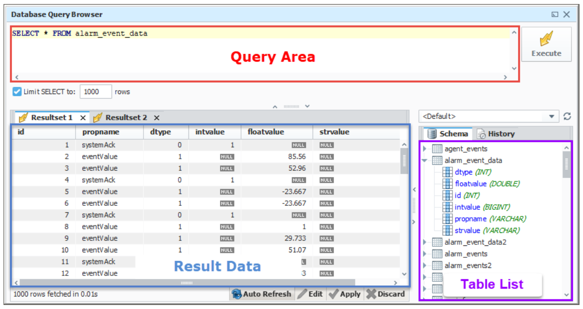
- Output Console
- Python Scripting
- Scripting in Ignition

Database Query Browser

The Database Query Browser Panel

The **Database Query Browser** is a very convenient tool that lets you query any database connected to Ignition, and interact with tables. Because Ignition is so heavily integrated with databases, it is very common in the course of project design the need to inspect the database directly, or to experiment with a SQL query to get it just right.

The Database Query Browser is found in the tools menu and has a three basic parts to it. The **Query Area** is where a query can be written and then executed against the selected database connection. This can accept any type of query, but if the query would update the database, the Designer needs to have its Gateway Communication Mode set to read/write first. The database connection that the query is executed against can be chosen from the dropdown, below the **Execute** button. The **Result Data** space is where the results of the executed query will appear. If a SELECT query was run, then the table data will be shown. If an UPDATE, INSERT, or DELETE query was run, then the number of rows affected will be displayed instead. Finally, the **Table List** will display all of the tables in the specified database connection. The tables can be expanded to show the columns and their data types for that table. This can help when writing queries in the Query Area. Additionally, when a table is double clicked, the Query Area will automatically be populated with a SELECT * FROM the table.



```
SELECT * FROM alarm_event_data
```

☑ Limit SELECT to: 1000 rows

id	propname	dtype	intvalue	floatvalue	strvalue
1	systemAck	0	1	NULL	NULL
2	eventValue	1	NULL	85.56	NULL
3	eventValue	1	NULL	52.96	NULL
4	systemAck	0	1	NULL	NULL
5	eventValue	1	NULL	-23.667	NULL
6	eventValue	1	NULL	-23.667	NULL
7	systemAck	0	1	NULL	NULL
8	eventValue	1	NULL	1	NULL
9	eventValue	1	NULL	29.733	NULL
10	eventValue	1	NULL	51.07	NULL
11	systemAck	[illegible]	[illegible]	[illegible]	NULL
12	eventValue	[illegible]	[illegible]	3	NULL

1000 rows fetched in 0.01s

- agent_events
- alarm_event_data
  - dtype (INT)
  - floatvalue (DOUBLE)
  - id (INT)
  - intvalue (BIGINT)
  - propname (VARCHAR)
  - strvalue (VARCHAR)
- alarm_event_data2
- alarm_events
- alarm_events2

INDUCTIVE UNIVERSITY

Using the Query Browser

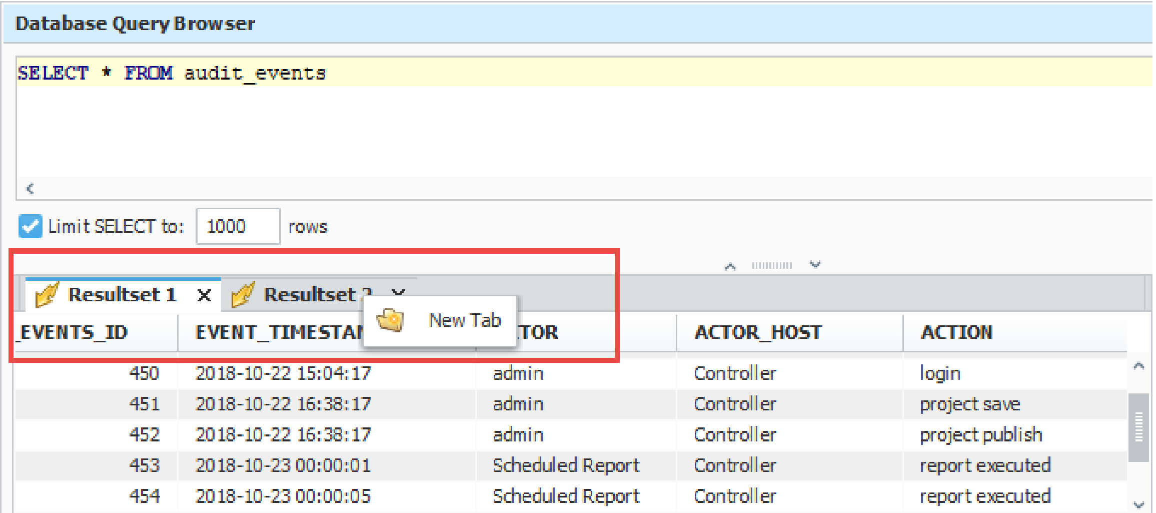
Watch the Video

Features of the Query Browser

The Database Query Browser has a few features that can help manage and build any SQL query.

Multiple Resultsets

The Query Browser allows you to make multiple tabs of results, so that multiple queries can be run and their results compared to each other. Simply right click on the **Resultset** area and select **New Tab**. Your queries will be saved for each tab.



Database Query Browser

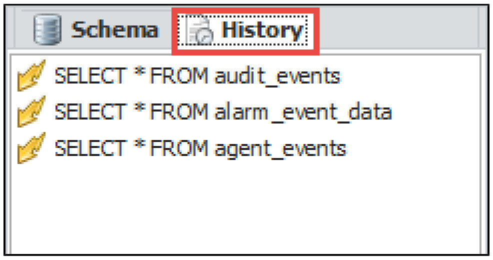
```
SELECT * FROM audit_events
```

☑ Limit SELECT to: 1000 rows

_EVENTS_ID	EVENT_TIMESTAI	TOR	ACTOR_HOST	ACTION
450	2018-10-22 15:04:17	admin	Controller	login
451	2018-10-22 16:38:17	admin	Controller	project save
452	2018-10-22 16:38:17	admin	Controller	project publish
453	2018-10-23 00:00:01	Scheduled Report	Controller	report executed
454	2018-10-23 00:00:05	Scheduled Report	Controller	report executed

Query History

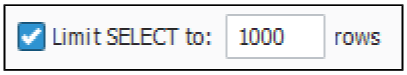
At the top of the Table List, there is a second tab labeled **History**. This will switch the Table List to instead show the history of queries run in the query browser. Double clicking one of the entries will push the query into the Query Area (for the selected tab). This allows you see what queries have been run previously to help you tweak your current query.



```
SELECT * FROM audit_events
SELECT * FROM alarm_event_data
SELECT * FROM agent_events
```

Select Limit

By default, any SELECT statement is limited to 1000 rows. This is to help the queries return quickly, however, it may not always be wanted. This can be turned off or adjusted by either clicking the checkbox or manipulating the row number located underneath the Query Area on the left. It is recommended to leave this on unless you know the result set size. It's better to use a count query than to return all results to see the result count. IE: SELECT COUNT(*) FROM table...



☑ Limit SELECT to: 1000 rows

Auto Refresh

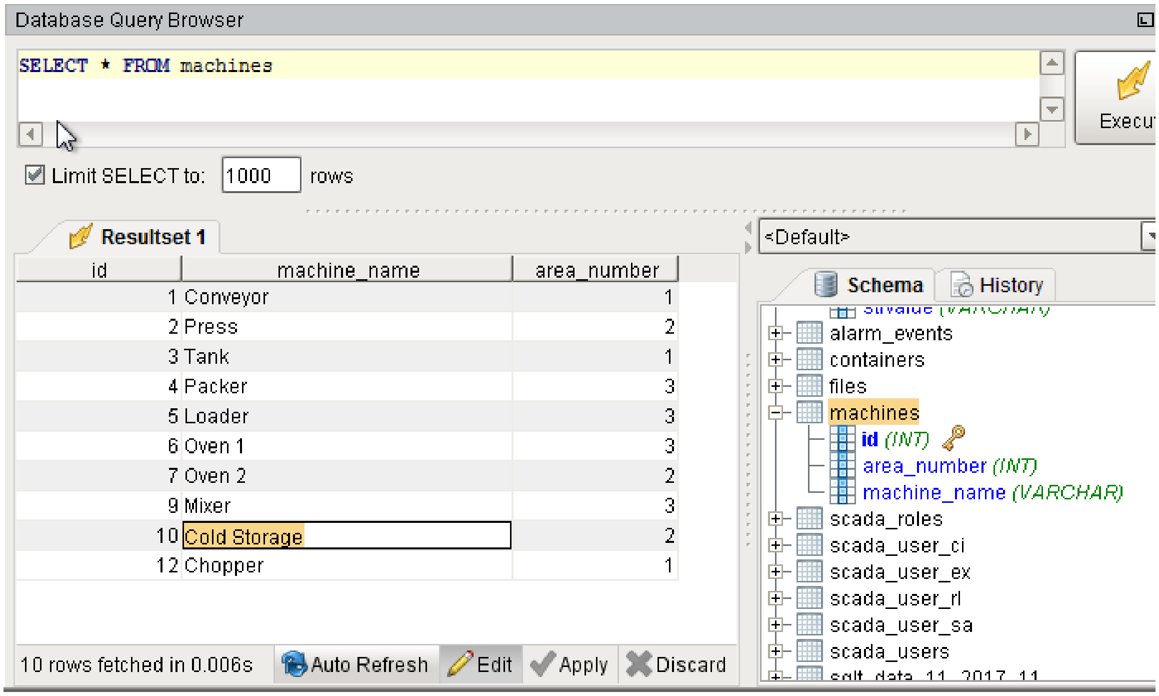
The Query Browser also lets you monitor a database table for changes by using the **Auto Refresh** button. This is often convenient when designing Transaction Groups. As the group runs, you can view the table that it is targeting with Auto Refresh turned ON to watch how the group is altering the table. Simply click the Auto Refresh button at the bottom of the Query Browser to periodically rerun the query in the Query Area. Make sure to include an ORDER BY clause so your results show up in the order you want.

Editing the Table in the GUI

In addition to editing the table data using INSERT, UPDATE, and DELETE statements, the data can also be edited within the Result Data. Simply click the Edit button at the bottom of the Query Browser window.

Editing in this way requires that the data be obtained from a single table with a simple query. The table must also not be tied to other tables.

Once the Edit button is clicked, the values in the table can be edited by double clicking on the value and entering a new one.



```
SELECT * FROM machines
```

☑ Limit SELECT to: 1000 rows

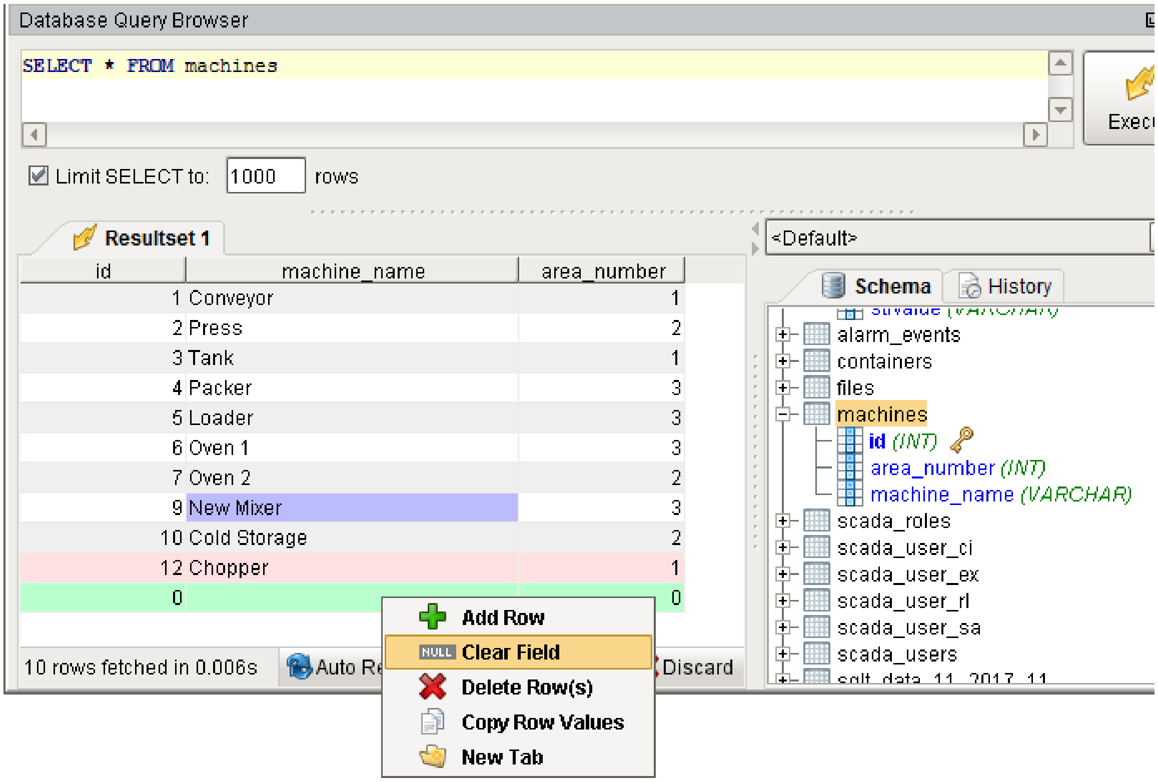
id	machine_name	area_number
1	Conveyor	1
2	Press	2
3	Tank	1
4	Packer	3
5	Loader	3
6	Oven 1	3
7	Oven 2	2
9	Mixer	3
10	Cold Storage	2
12	Chopper	1

10 rows fetched in 0.006s

Right clicking on a row also provides a few options:

- **Add Row**: Will add a new row to the table for data to be entered into.
- **Clear Field:** Will clear out the value in the selected cell so that it will be NULL. This is different than simply deleting the value out and leaving it empty.
- **Delete Row(s):** Will delete the selected row or rows from the table.
- **Copy Row Values:** Will copy the row values in a comma separated form to the clipboard.
- **New Tab:** Will Create a new Resultset tab for a new query to be run in.

When editing values, cells will highlight depending on what is being done to them. Green cells are new, and typically indicate a new row was added. Red cells are marked for deletion, and will be deleted when the changes are confirmed. Blue cells are cells that have had values changed during editing.



Database Query Browser

```
SELECT * FROM machines
```

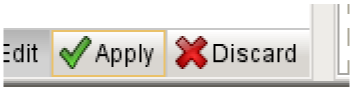
☑ Limit SELECT to: 1000 rows

Resultset 1

id	machine_name	area_number
1	Conveyor	1
2	Press	2
3	Tank	1
4	Packer	3
5	Loader	3
6	Oven 1	3
7	Oven 2	2
9	New Mixer	3
10	Cold Storage	2
12	Chopper	1
0		0

10 rows fetched in 0.006s

After making edits to the table data, the changes then either need to be applied or discarded. This gives you the opportunity to revert the table to the way it was before the current edit session, or apply the changes and rewrite the table appropriately. Simply click the corresponding button at the bottom of the Query Browser next to the Edit button. Make sure your Gateway Communication Mode is set to Read/Write before Applying your changes.

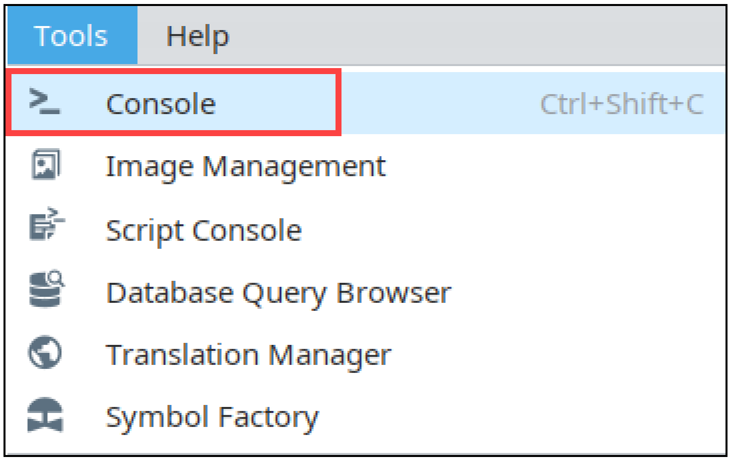


Related Topics ...

- Keyboard Shortcuts

Output Console

The Output Console is the script-writer's best friend. It is a dockable panel, and can be opened via the **Tools > Console** menu or the **Ctrl-Shift-C** keyboard shortcut.



The **Console** is a dockable panel that you can open via **Tools > Script Console** menu or the **Ctrl-Shift-C** keyboard shortcut. The Console prints system messages that are coming from the Designer. This can vary from simple info messages that show when things are loaded, to error messages when something goes wrong. The Console is also frequently used to test and debug Python scripts, as print statements on components that are run in the designer are printed here.

For example, by using the **print** function in your script, you can observe the inner workings of your script as it executes. If you executed the following script:

```
# A python script that demonstrates the print statement

print "Print me first"
x=10
z=2
print x, y, x/y

#Result looks like this:
#Print me first
#10 2 5
```

Gateway scoped information will not appear in either the Output Console or Script Console. Instead, the output will be sent to the wrapper.log file. Alternatively, system.util.getLogger() will send messages to the Gateway Console, and is a preferred method of troubleshooting Gateway scoped scripts.

The Output Console is most frequently used to test and debug Python scripts on components in Ignition. By using the **print** keyword in your script, you can observe the inner workings of your script as it executes. For example, if you executed the following script:

Python - Using the Output Console to Test and Debug Python Scripts

```
# A function that intercepts tag writes, printing out the previous value first
def writeToTag(path, value):
   prevValue = system.tag.getTagValue(path)
   print "Writing value '%s' to %s, was previously '%s'" % (value, path, prevValue)
   system.tag.writeToTag(path, value)

writeToTag("Compressor/HOA", 2)
```

It would print the following to the console:

```
Writing value '2' to Compressor/HOA, was previously '0'
```

⚠ The Output Console is also available in the Vision Client from **Help > Diagnostics** and selecting the **Console** tab.

Related Topics ...

- Python Scripting
- Scripting in Ignition

Project Properties

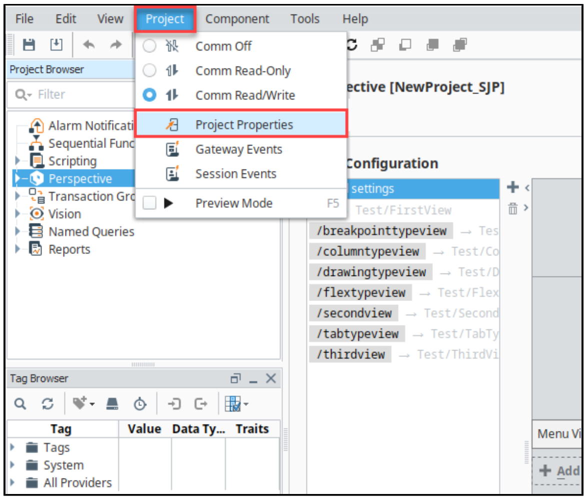
There are a number of properties you can set for your projects within the Designer, Vision Clients, and Perspective Sessions. For example, there are properties for setting the touchscreen mode, customizing a client's auto-login, and configuring how the clients receive updates.

The property settings on the **Project Property** window apply to the whole project. This page identifies and describes all the available project properties.

On this page ...

Accessing Project Properties

In the Designer, click on Project tab on the menu bar at the top of the Designer, then select **Project Properties**.



The Project Properties screen is displayed. Project Properties span several functional areas each containing settings applicable to that area.

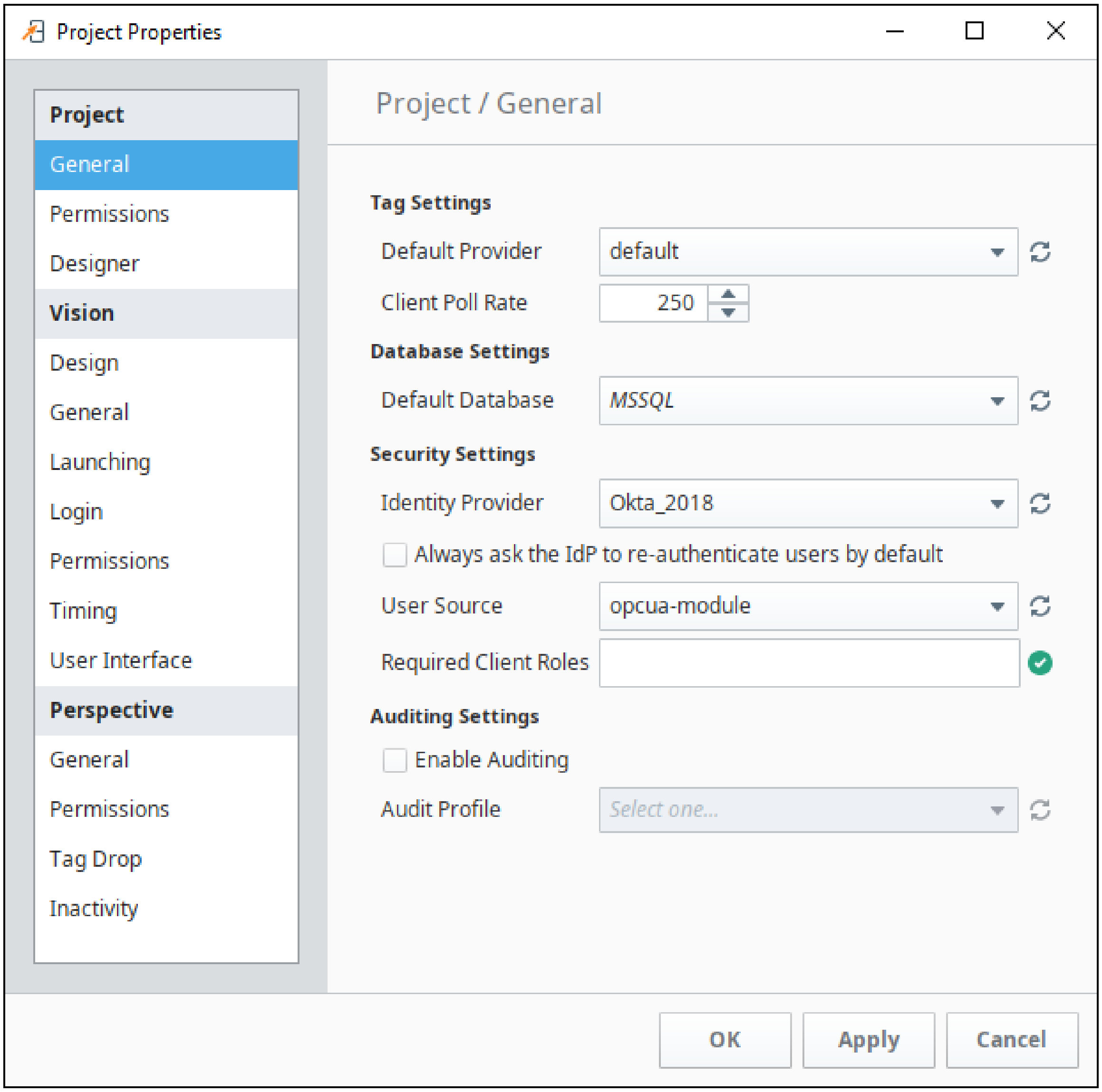
About Project Properties

Watch the Video

Project Properties

A project's General properties apply to the project as a whole, across all module functionality. For a new project, there are many default settings that you can use. For example, there are default settings for the Tag provider, database, initial comm mode, window editing, and Client launching.

When properties in a section have been updated but not saved yet, the section heading will change to italicized text.



Project Properties

Project / General

Tag Settings

Default Provider: default

Client Poll Rate: 250

Database Settings

Default Database: *MSSQL*

Security Settings

Identity Provider: Okta_2018

□ Always ask the IdP to re-authenticate users by default

User Source: opcua-module

Required Client Roles:

Auditing Settings

□ Enable Auditing

Audit Profile: *Select one...*

There are a few properties of a project, such as its name, description, and title that are not available from the Designer and you need to set them in the Gateway. To do so, go to the Gateway, under the **Config > Projects** section, click on the **edit** link next to the project. You cannot edit these settings while the project is open in the Designer.

Important Concept: Defaults

Under **Project > General** is where you set the project's **Default Database** and its **Default Tag Provider**. It is important to understand how to use defaults effectively for proper project design.

Wherever you use a database connection or a Tag in a project, you are always given the option to use the project's default, or an explicitly named connection or provider. If your project is like most typical projects, it primarily uses a single database and a single Tag provider. By consistently using the **default** option, you make your project more resilient to change.

For example, suppose you have a project, and it has a database connection named **`Production_DB`**. Now you want to adapt the project to a new, similar plant, while leaving the existing project intact. You copy the project and create a new database connection, named **`New_DB`**. If your project consistently used its default database connection, the switchover will be as simple as changing the copied project's default database. However, if you used the explicit **`Production_DB`** connection in your groups and screens, you will need to laboriously switch the bindings over to **`New_DB`**.

General Properties

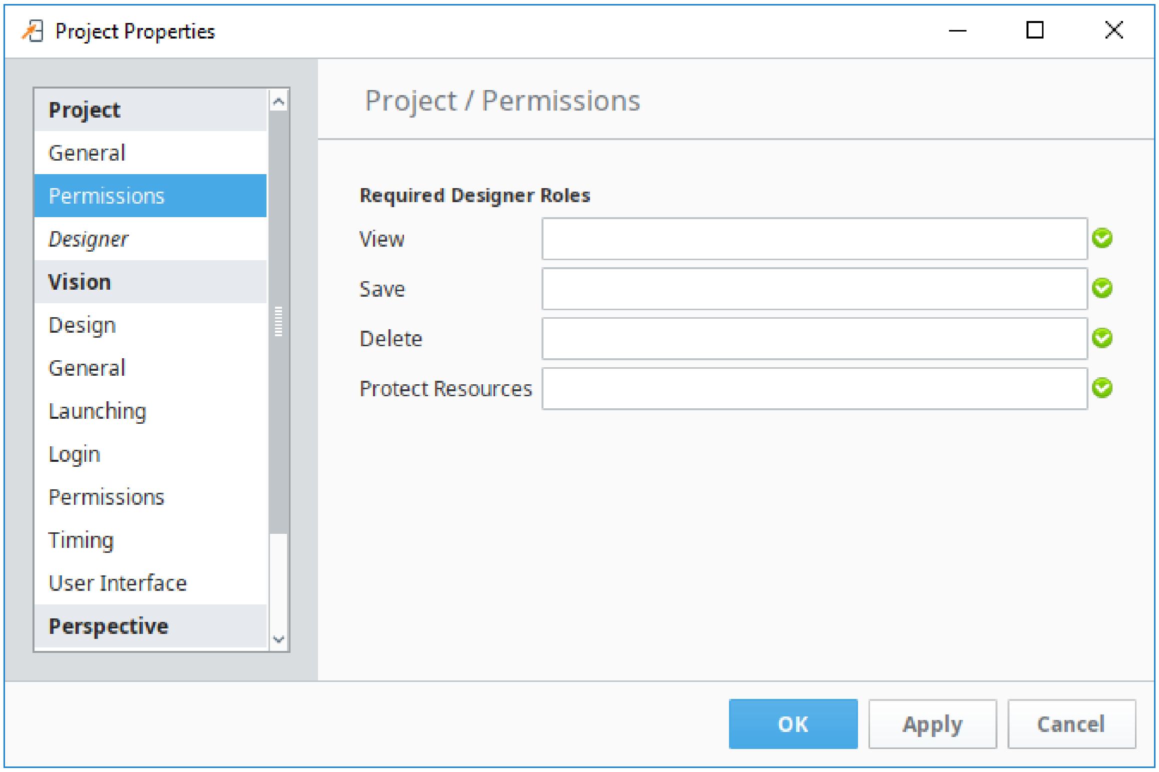
Tag Settings	
Property	**Description**
Default Provider	The Tag provider chosen here will act as the project's default Tag provider. See also: Understanding Tags.
Client Poll Rate	The Client poll rate (in milliseconds) is the rate at which a Vision Client or Ignition Designer polls the Gateway for updates to its subscribed Tags.
Database Settings	
Default Database	This is the default database connection to use for this project. To use the default database connection, use the special `<default>` connection, or in scripting, the empty-string connection `""`.
Security Settings	
Identity Provider	The following feature is new in Ignition version **8.0.6** Click here to check out the other new features The default Identity Provider (IdP) for this project. Choose from a dropdown list of IdPs. You can also choose <None> if the project already had an IdP but you want to remove it.
Always ask the IdP to re-authenticate users by default	The following feature is new in Ignition version **8.0.14** Click here to check out the other new features When enabled, authenticated uses will always need to re-enter their credentials on login. When false, if a user is already authenticated, then they will not be required to re-type their credentials when an action in the session triggers a login (such as a call to system.perspective.login). ⚠ Allowing re-authentication is entirely determined by the Identity Provider. This setting simply makes the Gateway send the re-authentication request. Consult your Identity Provider's documentation for information on re-authentication support.
User Source	Choose the User Source that governs this project's security: specifically, which group of users (User Sources) is allowed to log into the client. The User Sources are all defined in Gateway's **Config** section under the **Security > User, Roles** page. Note that this is for Vision projects only.
Required Client Roles	This property is for Client logins, and determines what role(s) a user must have before they can log into the Client. You can optionally specify a list of roles that are required for a user to log into this project. Use commas to separate the roles. Users must have **at least one of** the roles in order to log in. If no roles are specified, the user only needs to correctly authenticate with the User Source in order to log in. To define the roles, go to the Gateway's **Configure** section under the **Security > User, Roles** page, click the **manage users** link, and then go to the **Roles** tab. See also Security.
Audit Settings	
Enable Auditing	If auditing is enabled, audit events that relate to this project will be stored in the chosen audit profile.
Audit Profile	The audit profile stores the audit events when auditing is enabled.

Permissions Properties

When opening the project in the Designer, these properties determine which roles are required when making certain changes to the project. More details can be found on the Project Security in Designer and Gateway page.

Required Designer Roles	
View	Users must have at least one of these roles to view the project in the Designer.
Save	Users must have at least one of these roles to save the project.
Delete	Users must have at least one of these roles to delete the project.

Protected Resources	Users must have at least one of these roles to access protected resources.



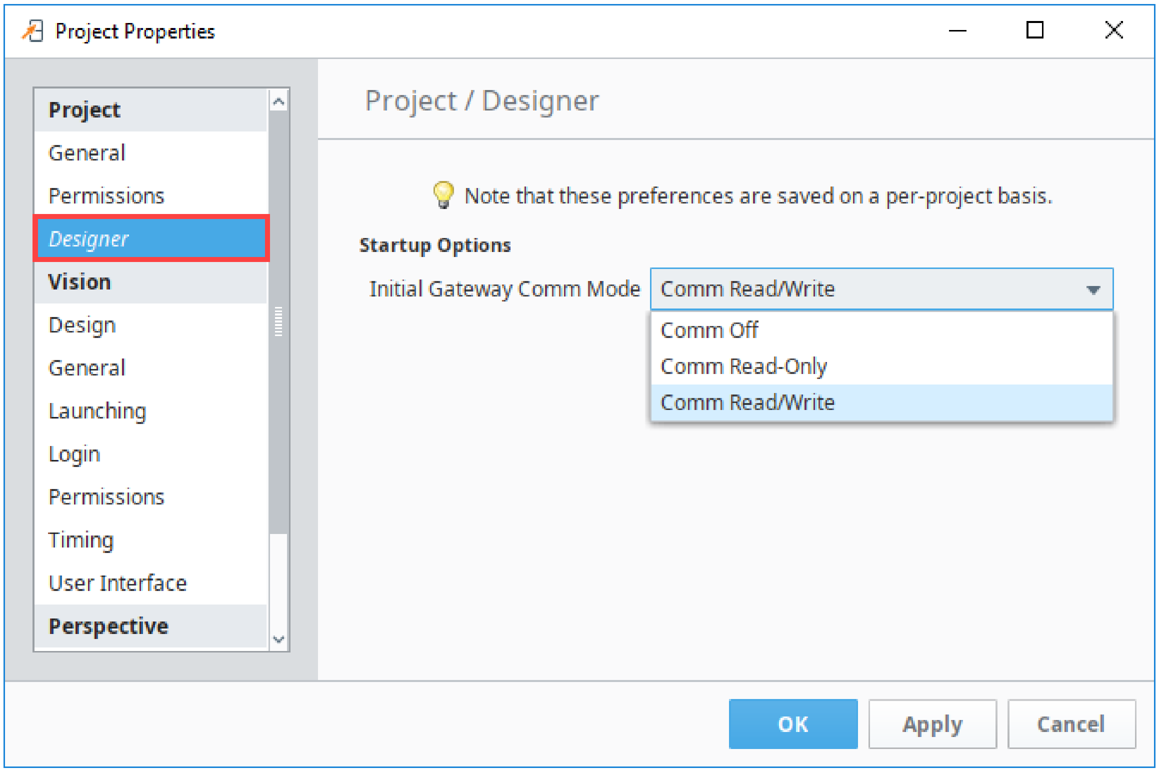
Required Designer Roles

View	✓
Save	✓
Delete	✓
Protect Resources	✓

Designer Properties

These preferences are saved on a per-project basis.

Startup Options	
Initial Gateway Comm Mode	The Designer starts up in the default Comm Read-Only mode. The property allows you to change the mode the Designer starts in when viewing the project. The options are: **Comm Off**, **Comm Ready-Only**, **Comm Read/Write**. • **Comm Off** - In this mode, all database query traffic and Tag subscriptions and writes are blocked. • **Comm Read-Only** - Tag subscriptions and SELECT queries work, but Tag writes and UPDATE/INSERT/DELETE queries are blocked. • **Comm Read/Write** - The Designer may freely request Tag and database values from the Gateway, as well as write or change these values. For more information, see the Communication Modes page.



Project / Designer

Note that these preferences are saved on a per-project basis.

Startup Options

Initial Gateway Comm Mode: Comm Read/Write

Vision Project Properties

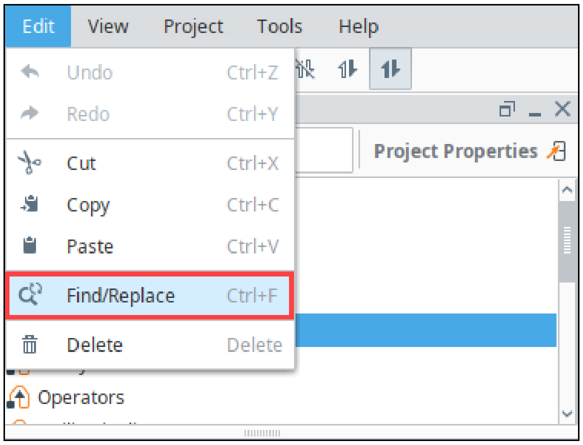
There are many project properties that apply specifically to Vision Clients. You can find more information at Vision Project Properties.

Perspective Properties

There are many project properties that apply specifically to Perspective Sessions. You can find more information at Perspective Project Properties.

Find and Replace

The Find/Replace tool in the Edit menu of the Designer allows you to search your entire project for specific components, properties, scripts, Transaction Groups, and Tags. You can then use the replace feature to make mass changes to a project with the click of a button.



On this page ...

INDUCTIVE UNIVERSITY

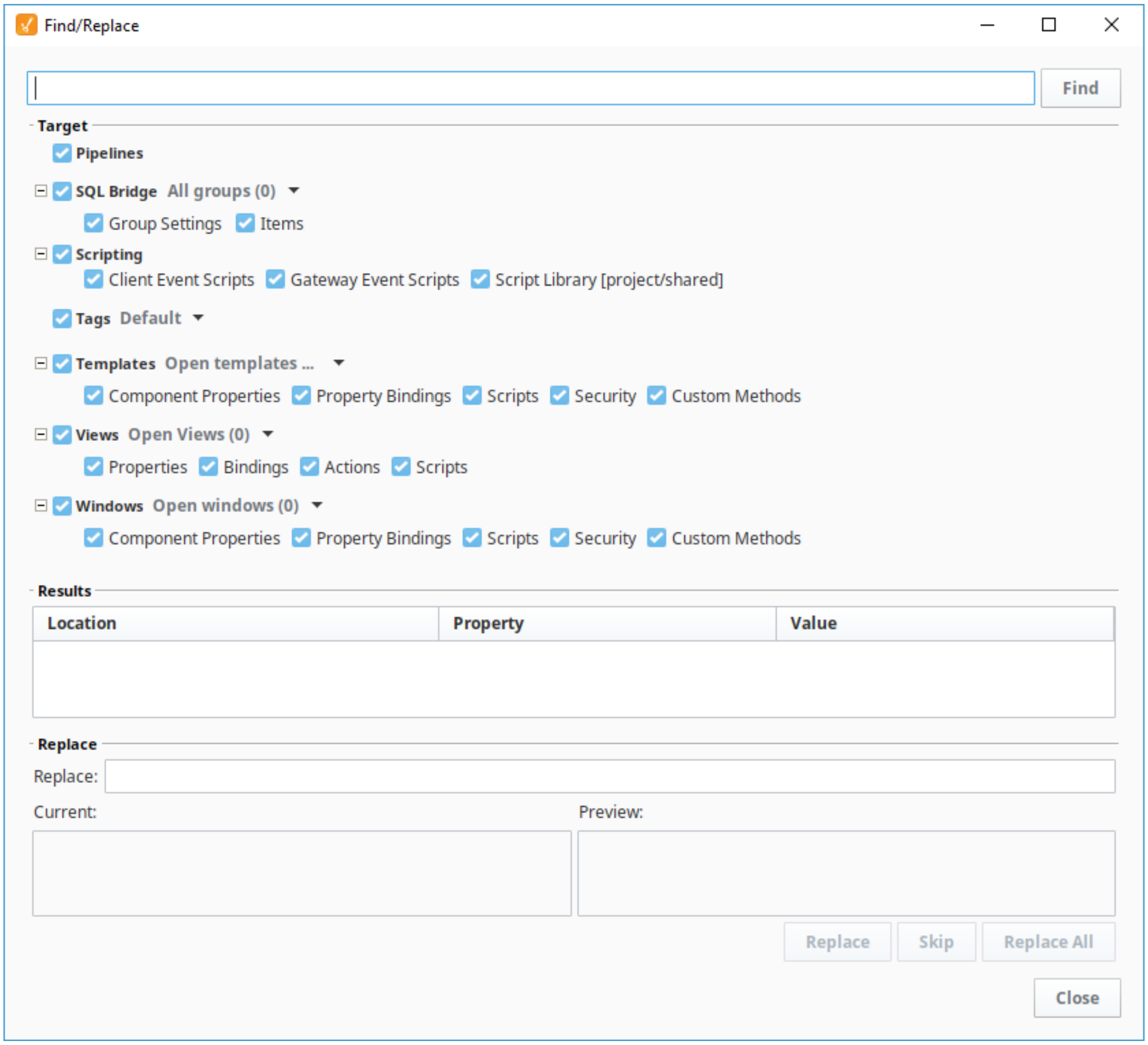
Using Find and Replace

Watch the Video

Search Options

In the Target section, the Find/Replace tool has options for searching through many different parts of a project.

- Pipelines
- SQL Bridge Transaction Groups
- Scripting
- Tags
- Templates (Vision)
- Views (Perspective)
- Windows (Vision)


Find/Replace

Find

Target

☑ Pipelines

⊟ ☑ SQL Bridge All groups (0)

☑ Group Settings ☑ Items

⊟ ☑ Scripting

☑ Client Event Scripts ☑ Gateway Event Scripts ☑ Script Library [project/shared]

☑ Tags Default

⊟ ☑ Templates Open templates ...

☑ Component Properties ☑ Property Bindings ☑ Scripts ☑ Security ☑ Custom Methods

⊟ ☑ Views Open Views (0)

☑ Properties ☑ Bindings ☑ Actions ☑ Scripts

⊟ ☑ Windows Open windows (0)

☑ Component Properties ☑ Property Bindings ☑ Scripts ☑ Security ☑ Custom Methods

Results

Location	Property	Value

Replace

Replace:

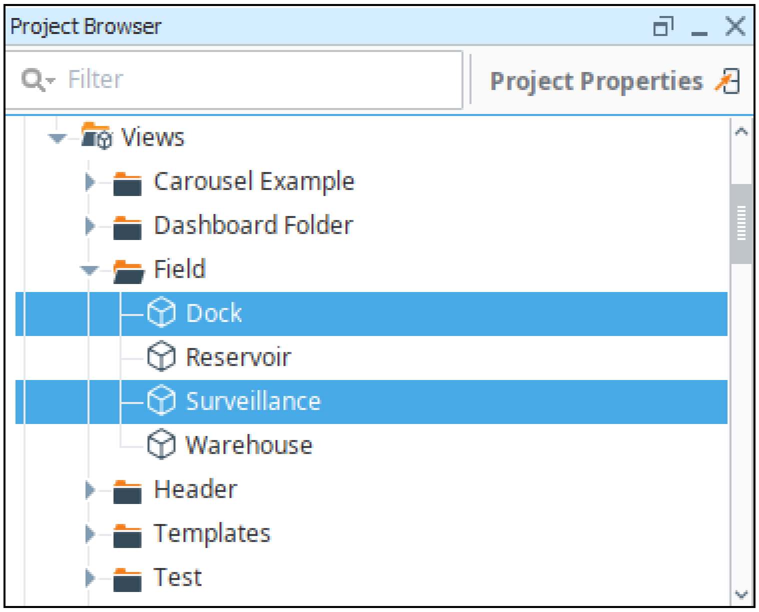
Current:

Preview:

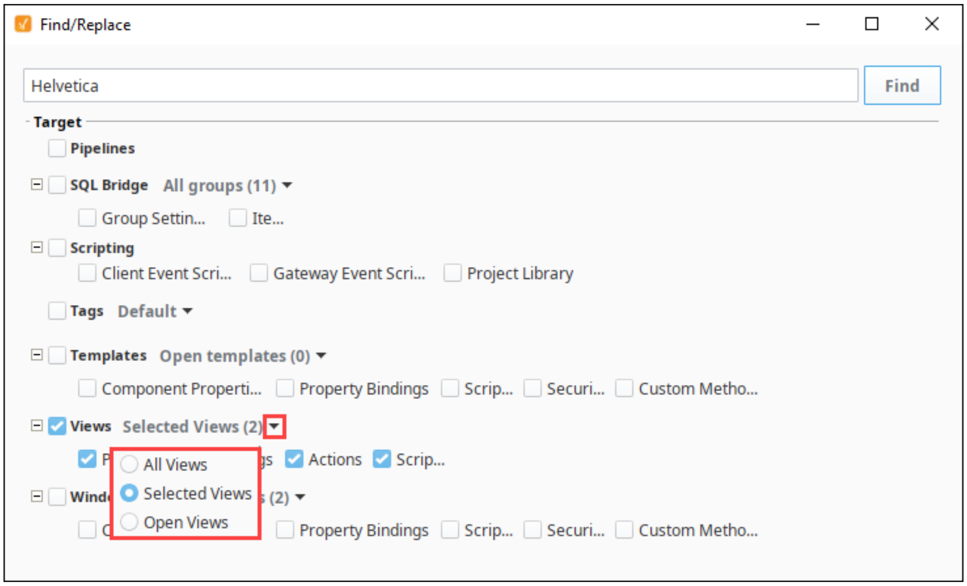
You can narrow down your search by selecting and deselecting categories you include in the search. The SQL Bridge, Templates, Views, and Windows options also have dropdown options to customize your search.

For example, if you want to search only a couple Views, do the following:

1. In the Project Browser, use **Ctrl-Click** to select the views you want to search.



2. In the Designer toolbar, go to **Edit > Find/Replace** or use the shortcut **Ctrl-F**. The Find/Replace window is displayed.
3. De-select all of the options except for **Views**.
4. In Views, click on the expand arrow and choose the **Selected Views** option.



Find/Replace

Helvetica

Find

Target

□ Pipelines

⊟ □ SQL Bridge All groups (11) ▾

□ Group Settin... □ Ite...

⊟ □ Scripting

□ Client Event Scri... □ Gateway Event Scri... □ Project Library

□ Tags Default ▾

⊟ □ Templates Open templates (0) ▾

□ Component Properti... □ Property Bindings □ Scrip... □ Securi... □ Custom Metho...

⊟ ☑ Views Selected Views (2) ▾

○ All Views
● Selected Views
○ Open Views

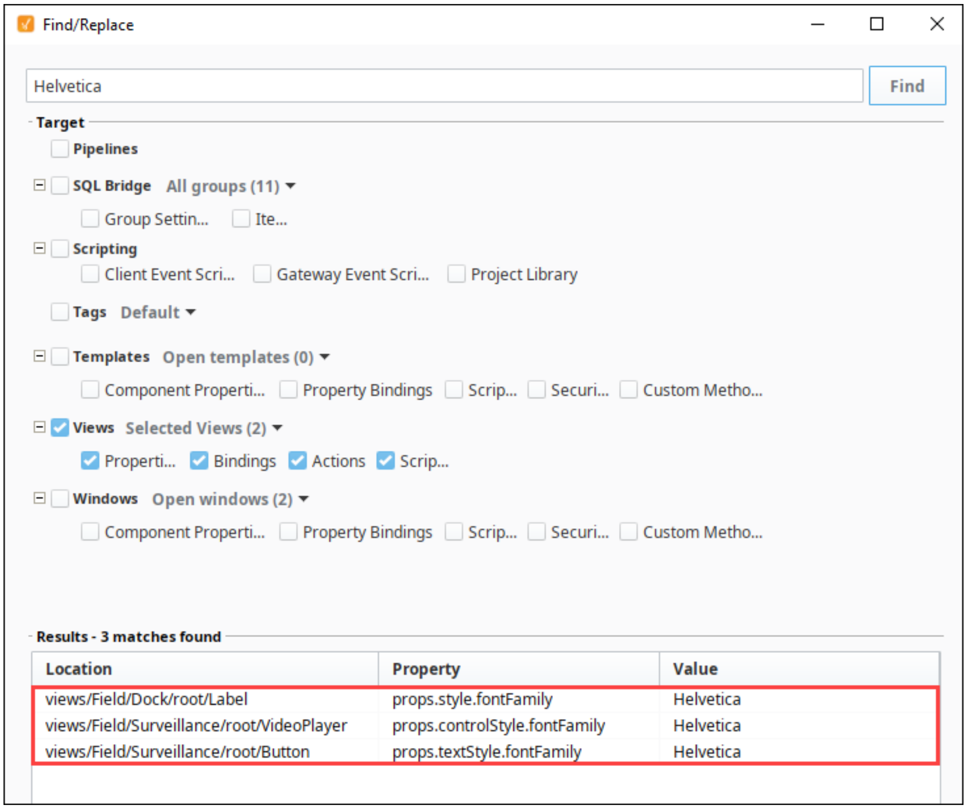
☑ P gs ☑ Actions ☑ Scrip...

⊟ □ Wind s (2) ▾

□ C □ Property Bindings □ Scrip... □ Securi... □ Custom Metho...

5. Enter the property, action, script, or such that you want to search for. In this case we searched for property value Helvetica.
6. Click **Find**.

7. The search results are displayed in the Results box.



Helvetica

Target

□ Pipelines

⊟ □ SQL Bridge All groups (11)
- □ Group Settin... □ Ite...

⊟ □ Scripting
- □ Client Event Scri... □ Gateway Event Scri... □ Project Library

□ Tags Default

⊟ □ Templates Open templates (0)
- □ Component Properti... □ Property Bindings □ Scrip... □ Securi... □ Custom Metho...

⊟ ☑ Views Selected Views (2)
- ☑ Properti... ☑ Bindings ☑ Actions ☑ Scrip...

⊟ □ Windows Open windows (2)
- □ Component Properti... □ Property Bindings □ Scrip... □ Securi... □ Custom Metho...

Results - 3 matches found

Location	Property	Value
views/Field/Dock/root/Label	props.style.fontFamily	Helvetica
views/Field/Surveillance/root/VideoPlayer	props.controlStyle.fontFamily	Helvetica
views/Field/Surveillance/root/Button	props.textStyle.fontFamily	Helvetica

Wildcards

To customize your search further, you can use the wildcard characters * and ?.

Use an asterisk (*****) to indicate that any character(s) could be where the asterisk is. For example, to search for Tags that start with the word "Motor" enter "Motor*". This would return Motor 1, Motor2, Motor_East3, and so forth.

Use a question mark (**?**) to indicate any single character could be where the question mark is. For example, to search for Valve1Status through Valve9Status, you could enter Valve?Status. Note that this would not return something like Valve44Status because that is more than one character where the wildcard is.

Using Find and Replace

1. In the Designer toolbar, go to **Edit > Find/Replace** or use the shortcut **Ctrl-F**. The **Find/Replace** window is displayed.
2. Enter what you're searching for in the textbox at the top to search through your project.
3. Click the **Find** button at the top right to execute the search. All matching items appear in the **Results** section. Double-click on an item in the Results table to bring that item into editing focus in the Designer.
4. To replace a value, select an entry in the Results section and you'll see the Value of that entry in the **Current** box of the **Replace** section.
5. Enter the text you want to use as a replacement in the **Replace** textbox and a preview of the new value is shown in the **Preview** box.
6. Click the **Replace** button to execute the replace. This will move your selection down in the Results table so that you can rapidly execute multiple replacements. If you're satisfied and you'd like to make the identical replacement to many items, select them all in the Results table and click the **Replace All** button.

Related Topics ...

- Database Query Browser

Keyboard Shortcuts

Using Keyboard Shortcuts in the Designer

You can interact with screens and more by using some of the popular keyboard shortcuts in the Designer.

Action	Keystrokes
Nudge Moves selected component(s) in the direction of the arrow key by the default nudge distance.	, , ,
Alt-Nudge Same as Nudge, but uses the “alt-nudge” distance.	Atl + , Alt + , Alt + , Alt +
Resize Right Moves the right edge of the component left or right. Add Alt to use the alt-nudge distance.	Shift + , Shift +
Resize Bottom Moves the bottom edge of the component to top or bottom.	Shift + , Shift +
Resize Left Moves the left edge of the component left or right.	Ctrl + Shift + , Ctrl + Shift +
Resize Top Moves the top edge of the component to top or bottom.	Ctrl + Shift + , Ctrll + Shift +
Move Forward Moves the selected components(s) forward in the Z-order	PgUp
Move Backward Moves the selected component(s) backwards in the Z-order	PgDn
Move to Front Moves the selected component(s) to the front of the Z-order	Home
Move to Back Moves the selected component(s) to the back of the Z-order	End
Copy-Move Copies the component when holding Ctrl while doing a mousemove.	Ctrl + drag a component
Orthogonal-Move Restricts to only moving straight up, down, left, or right when holding Shift while doing a mousemove.	Shift + drag a component
Selection-Move Drags the components that are currently selected without having to press the mouse button down on the component first.	Hold Alt + drag a component
Copy-Axis-Move Combines copy move with axis move.	Hold Ctrl + Alt + drag a component.
Proportional Resize Resizes a component while maintaining its aspect ratio.	Hold Ctrl + resize a component
On-Center Resize Resizes the selection using the center as the anchor point.	Hold Shift+ resize a component
	Ctrl + A

On this page ...

- Using Keyboard Shortcuts in the Designer

Using the Popular Keyboard Shortcuts

Watch the Video

Select All (Vision only) Selects all components that are siblings of the selected component.	
Select Same Type (Vision only) Selects all components that are siblings of the selected component and the same component type.	Ctrl + Shift + A
Select Same Type in Window (Vision only) Selects all components in a window that are the same type as the selected component.	Ctrl + Alt + Shift + A
Layout (Vision only) Opens the Layout Constraints window to let you specify layout for the component (s).	Ctrl + L
Size & Position (Vision only) Opens Size & Position to let you specify exact size and position for selected component or window.	Ctrl + P
Customizer (Vision only) Opens the Custom Properties window to let you configure complex component properties.	Ctrl + U
Customizer 2 (Vision only) Some components have a secondary customizer. This command will open that customizer, if available.	Ctrl + 2
Jython Opens Event Configuration page (Perspective) or the Component Scripting page (Vision) to let you configure actions for component or window.	Ctrl + J
Script Configuration Opens the Script Configuration for the View or Component selected. (Perspective Only)	Ctrl + K
Security (Vision only) Opens Security Settings to let you set security for Component(s) or window.	Ctrl + E
Save Saves the project.	Ctrl + S
Open Opens the Open/Create Project to let you select a different project.	Ctrl + O
Undo Undoes the last action.	Ctrl + Z
Redo Gets rid of the last undo action.	Ctrl +Y
Copy Copies selected component(s) or window.	Ctrl + C
Duplicate Duplicates selected component(s) or window.	Ctrl + D
Cut Cuts selected component(s) or window.	Ctrl + X
Paste Pastes selected component(s) or window in clipboard. Pasted component(s) wait for position before pasting.	Ctrl + V
Immediately Paste (Vision only) Places pasted component(s) at the same location where they were copied/cut.	Ctrl + I
Comment/Uncomment Lines of Code Quickly comment or uncomment lines of a script or query in Designer.	Ctrl + /
Cancel Cancels a pending paste operation, deselects the current row of a table, cancels dragging components onto window.	Esc
	Ctrl + F

Find/Replace Opens Find/Replace to let you Search and replace the project based on string, pattern, or regex.	
Delete Deletes the current selection.	Delete
Snap to Grid (Vision only) Toggles whether or not moving and resizing components snaps to the grid.	Ctrl + G
Snap to Guides (Vision only) Toggles whether or not moving and resizing components snaps to guides.	Ctrl + Shift + G
Console Opens the Output Console.	Ctrl + Shift + C
Help Launches the Ignition User Manual in a web browser.	F1
Rename Renames the selected item (Tag, window, transaction group, component, and so on).	F2
Preview Mode Toggles preview/design mode.	F5
Launch Client/Session (Perspective and Vision only) When in the Vision workspace, launches a Vision Client in windowed mode. While in the Perspective workspace, launches a Perspective Session in windowed mode.	F10
Launch Full Screen Client (Perspective and Vision only) While in the Vision workspace, launches a Vision Client in full screen mode. While in the Perspective workspace, launchers a Perspective Session in full screen mode.	F11
Update Project Updates project to server. Receives concurrent edits from other Designers.	Ctrl + Shift + U
Zoom Zooms in/out in the Designer.	Ctrl + Mousewheel
Touch Selection (Vision only) Draws a line while dragging. Every component in the path of the line will be selected when letting go of the mouse button. To activate, a container must first be selected	Alt + Left Click and Drag
Select Through (Vision only) Click on a component that is beneath another component. You can do this multiple times if there are several layers of components.	Alt + Click
Code Folding/Unfolding Select a line of code, and this command will collapse the selection. Press again while the folded code is selected, and the code will unfold. The Script Console has a separate command for code folding.	Ctrl + .

Related Topics ...

- Saving Projects

Saving Projects

Once you created the Windows, Templates, Tags, Views, and anything else that goes into your project, you need to save your work. Saving your progress means pushing the changes from your Designer into the Gateway. If you are saving viewable resources like Vision Windows or Perspective Views, saving the changes means telling all of the clients or sessions that they can use the new updates to your project.

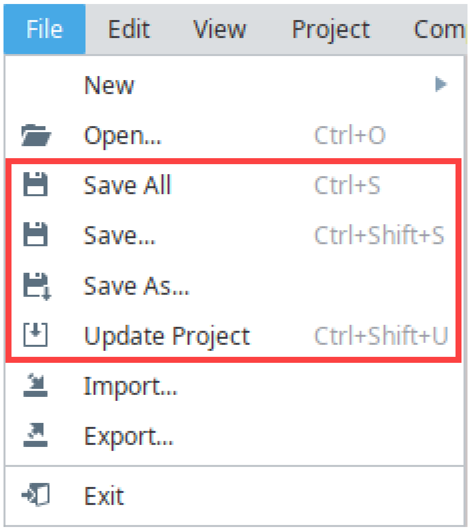
When you save, it's possible to save individual resources on a project as they are finished instead of saving the entire project. This is particularly helpful if you want to get resource updates to users quickly, or parts of a project into production sooner rather than waiting for the entire project to be completed

You also have options on how to perform Client and Session updates. You can choose between notifying the operator of an available update, or automatically pushing updates as soon as it is published making it transparent to the operator. This page assumes you are using the default Notify mode, see Client Update Modes or Session Project Updates for more information.

On this page ...

Saving a Project

The project save options are all located in the Designer under the **File** menu.



Project Creation and Publishing

Watch the Video

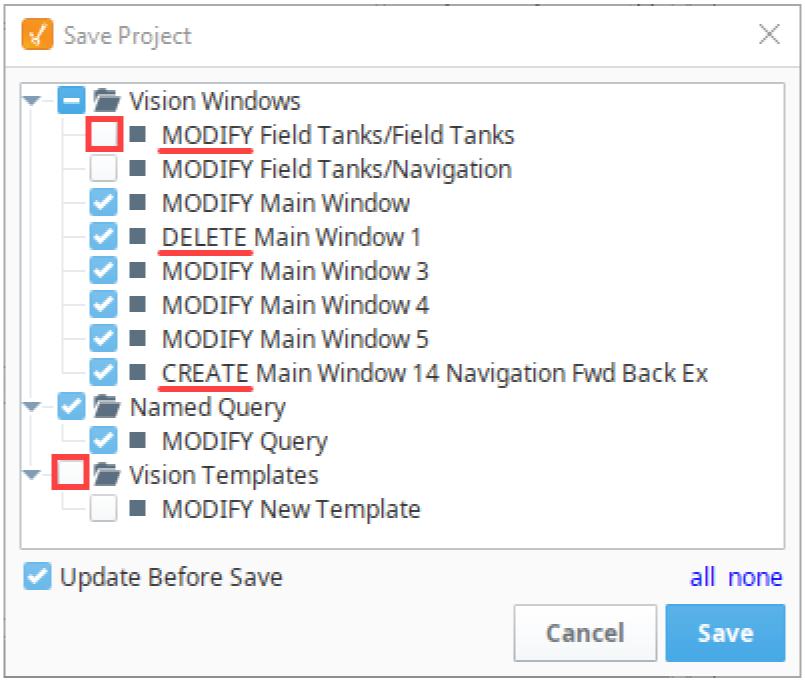
Save All

Save All option saves the entire project. The project will be saved immediately, unless there are any conflicts due to concurrent editing. If there are conflicts, the Resolve Conflicts screen will open automatically. Once you've resolved the conflicts, the Designer saves the project.

Save...

The **Save...** option option displays a list of all items that have been created, modified, or deleted since the last save. On this list you can save (or not save) individual resources on a project rather than of saving the entire project.

- To deselect all items, click "none" in the lower right corner.
- To select all items, click "all" in the lower right corner.
- To select or deselect individual items, click on the Checkbox ☑ icon next to the item.



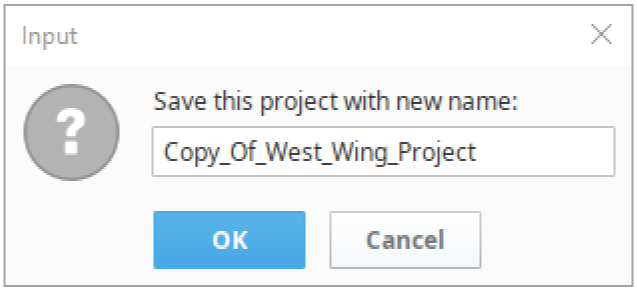
Save Project

- ⊟ Vision Windows
  - □ MODIFY Field Tanks/Field Tanks
  - □ MODIFY Field Tanks/Navigation
  - ☑ MODIFY Main Window
  - ☑ DELETE Main Window 1
  - ☑ MODIFY Main Window 3
  - ☑ MODIFY Main Window 4
  - ☑ MODIFY Main Window 5
  - ☑ CREATE Main Window 14 Navigation Fwd Back Ex
- ☑ Named Query
  - ☑ MODIFY Query
- □ Vision Templates
  - □ MODIFY New Template

☑ Update Before Save

Save As

The **Save As** option enables you to save your open project with a different name. When you choose **File** > **Save as**, the Designer will display an Input window. It will append "Copy_Of_" to the beginning of the current project name. However, you can enter a different name in this Input window. Click **OK** to save the project with a new name.



Input

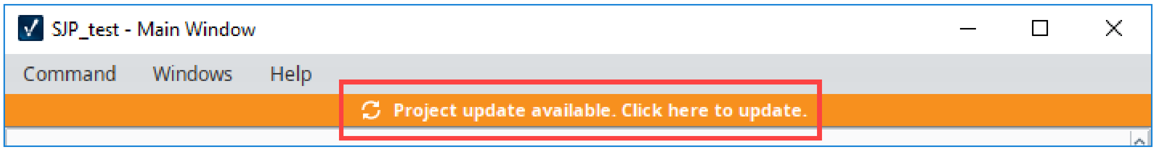
Save this project with new name: Copy_Of_West_Wing_Project

Update Project

The **Update Project** option pulls in any changes (saves by other people) that have happened since you opened the project or last updated it. If there are any conflicts due to concurrent editing, the Designer Resolve Conflicts screen will open automatically. Once you resolve the conflicts, the Designer updates the project.

Project Updates in a Vision Client

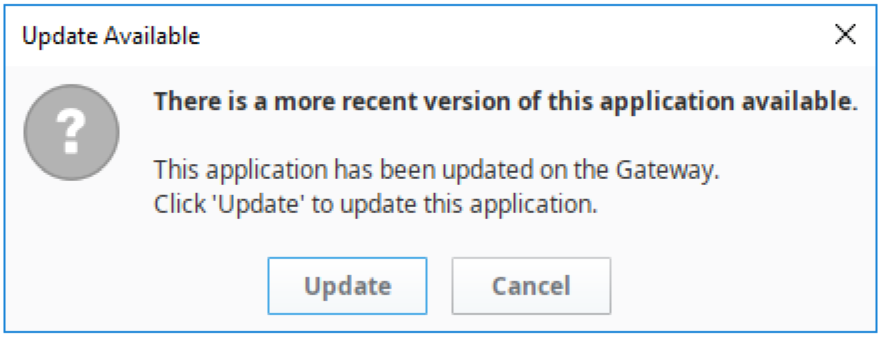
Depending on how Clients are setup to receive project updates, operators may be notified with a banner stating that a project update is available. For more information, see Client Update Modes.



SJP_test - Main Window

Project update available. Click here to update.

When the operator clicks on the banner, a confirmation window appears. The operator can choose to update or cancel and keep working in the previous version of the client.



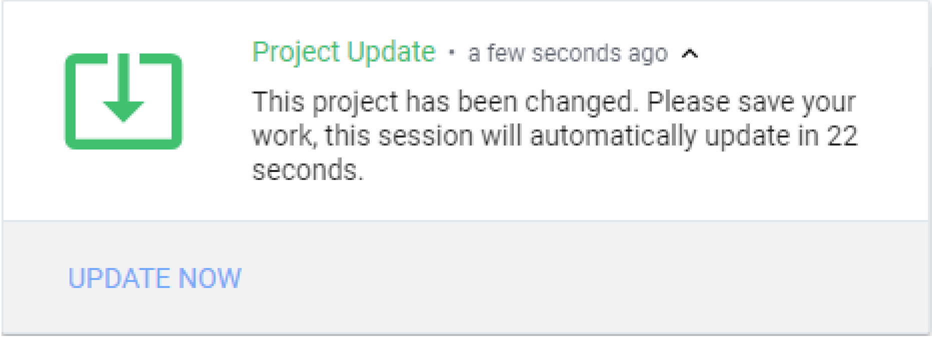
Update Available

There is a more recent version of this application available.

This application has been updated on the Gateway.
Click 'Update' to update this application.

Project Updates in a Perspective Session

If you have a Perspective Session open and a change was made in the Designer that was saved and published, one of two things may happen. Either the project will silently update, or an Update Notification window will appear in the session. Your session will automatically update in 30 seconds or you can click **Update Now**. For more information, see Session Project Updates.



Project Update • a few seconds ago

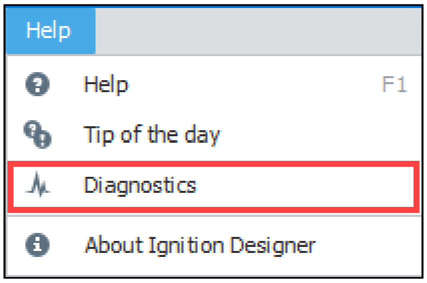
This project has been changed. Please save your work, this session will automatically update in 22 seconds.

Related Topics ...

- Vision Client Launcher
- Client Update Modes
- Session Project Updates

Designer Diagnostics

The Help menu in the Designer has a Diagnostics option, which displays the Diagnostics window with tabs for six different troubleshooting features.



On this page

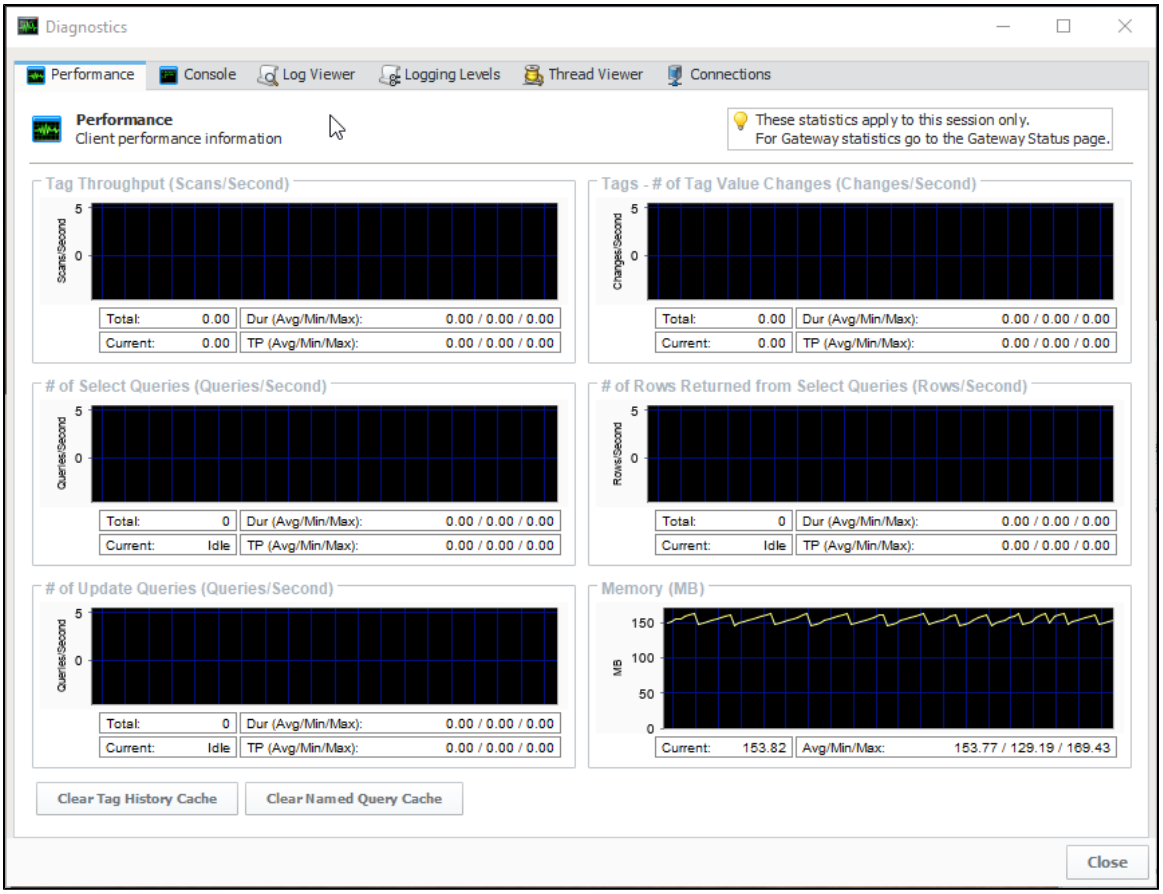
...

⚠ The information accessed through Designer Diagnostics is specific to the client runtime or the session. For Gateway statistics, see Diagnostics - Logs.

Performance Tab

The Performance tab displays six realtime charts, each showing an aspect of the current client's performance. These charts can be very useful to help troubleshoot performance issues, especially slow queries.

- **Tag Throughput (Scans/Second)** - Displays the Tag throughput in scans per second.
- **Tags - # of Tag Value Changes (Changes/Second)** - Displays the number of Tag changes in changes per second.
- **Select Queries (Queries/Second)** - One of the most common causes of query slowdown is simply running too many queries too frequently, and the # of Select Queries (Queries/Second) chart can help identify when this is occurring.
- **Rows Returned from Select Queries (Rows/Second)** - Displays the number of rows returned from selected queries in rows per second.
- **Update Queries (Queries/Second)** - Displays the number of update queries in queries per second.
- **Memory (MB)** - Displays the client's memory usage in megabytes. This will almost always be a saw tooth pattern since memory is used, discarded, and re-acquired on a regular basis.



Performance
Client performance information

These statistics apply to this session only.
For Gateway statistics go to the Gateway Status page.

Tag Throughput (Scans/Second)

Total: 0.00 | Dur (Avg/Min/Max): 0.00 / 0.00 / 0.00
Current: 0.00 | TP (Avg/Min/Max): 0.00 / 0.00 / 0.00

Tags - # of Tag Value Changes (Changes/Second)

Total: 0.00 | Dur (Avg/Min/Max): 0.00 / 0.00 / 0.00
Current: 0.00 | TP (Avg/Min/Max): 0.00 / 0.00 / 0.00

# of Select Queries (Queries/Second)

Total: 0 | Dur (Avg/Min/Max): 0.00 / 0.00 / 0.00
Current: Idle | TP (Avg/Min/Max): 0.00 / 0.00 / 0.00

# of Rows Returned from Select Queries (Rows/Second)

Total: 0 | Dur (Avg/Min/Max): 0.00 / 0.00 / 0.00
Current: Idle | TP (Avg/Min/Max): 0.00 / 0.00 / 0.00

# of Update Queries (Queries/Second)

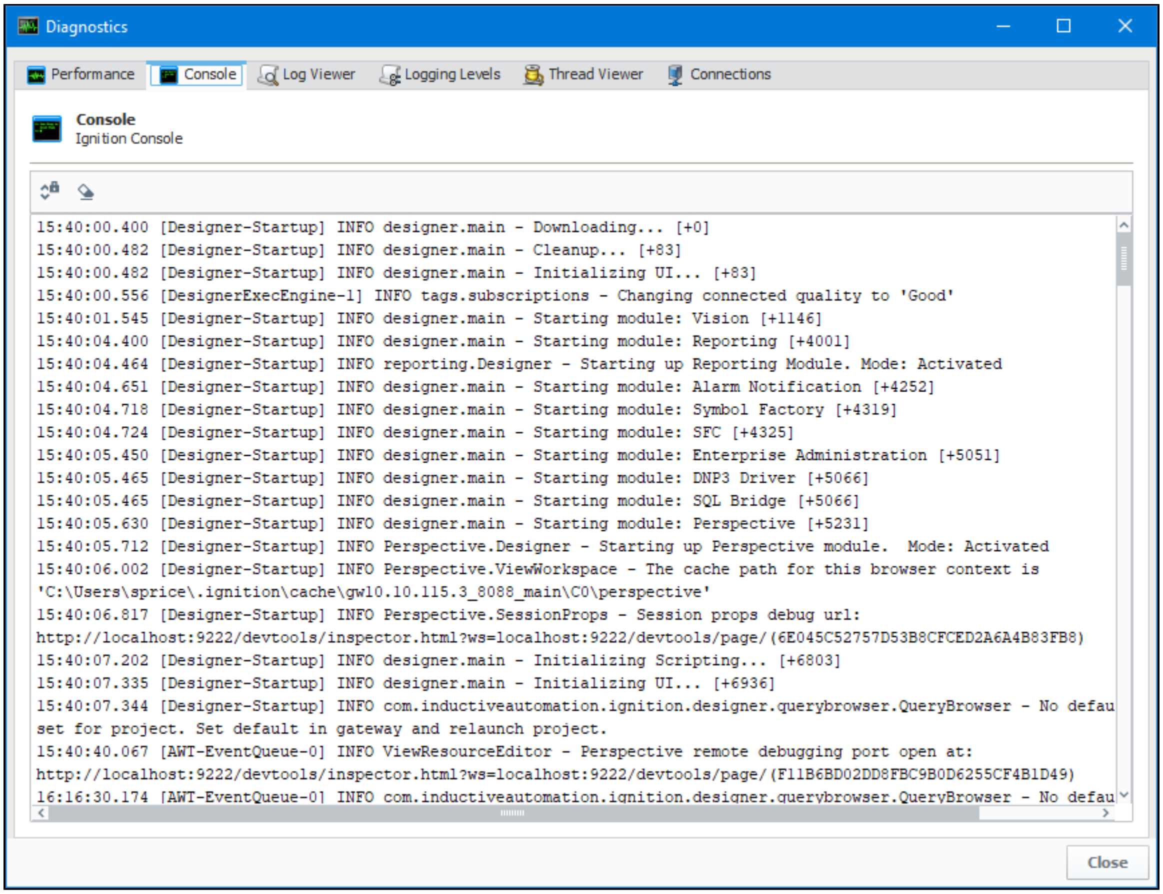
Total: 0 | Dur (Avg/Min/Max): 0.00 / 0.00 / 0.00
Current: Idle | TP (Avg/Min/Max): 0.00 / 0.00 / 0.00

Memory (MB)

Current: 153.82 | Avg/Min/Max: 153.77 / 129.19 / 169.43

Console Tab

The Console tab displays the Ignition Console. This console displays messages that are generated by the entity that the console was launched from, such as the Designer. Any print statements or errors in your scripts will show up here (except Gateway scope scripts).

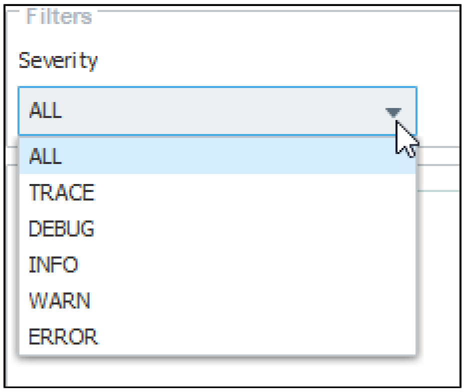


```
15:40:00.400 [Designer-Startup] INFO designer.main - Downloading... [+0]
15:40:00.482 [Designer-Startup] INFO designer.main - Cleanup... [+83]
15:40:00.482 [Designer-Startup] INFO designer.main - Initializing UI... [+83]
15:40:00.556 [DesignerExecEngine-1] INFO tags.subscriptions - Changing connected quality to 'Good'
15:40:01.545 [Designer-Startup] INFO designer.main - Starting module: Vision [+1146]
15:40:04.400 [Designer-Startup] INFO designer.main - Starting module: Reporting [+4001]
15:40:04.464 [Designer-Startup] INFO reporting.Designer - Starting up Reporting Module. Mode: Activated
15:40:04.651 [Designer-Startup] INFO designer.main - Starting module: Alarm Notification [+4252]
15:40:04.718 [Designer-Startup] INFO designer.main - Starting module: Symbol Factory [+4319]
15:40:04.724 [Designer-Startup] INFO designer.main - Starting module: SFC [+4325]
15:40:05.450 [Designer-Startup] INFO designer.main - Starting module: Enterprise Administration [+5051]
15:40:05.465 [Designer-Startup] INFO designer.main - Starting module: DNP3 Driver [+5066]
15:40:05.465 [Designer-Startup] INFO designer.main - Starting module: SQL Bridge [+5066]
15:40:05.630 [Designer-Startup] INFO designer.main - Starting module: Perspective [+5231]
15:40:05.712 [Designer-Startup] INFO Perspective.Designer - Starting up Perspective module.  Mode: Activated
15:40:06.002 [Designer-Startup] INFO Perspective.ViewWorkspace - The cache path for this browser context is
'C:\Users\sprice\.ignition\cache\gw10.10.115.3_8088_main\C0\perspective'
15:40:06.817 [Designer-Startup] INFO Perspective.SessionProps - Session props debug url:
http://localhost:9222/devtools/inspector.html?ws=localhost:9222/devtools/page/(6E045C52757D53B8CFCED2A6A4B83FB8)
15:40:07.202 [Designer-Startup] INFO designer.main - Initializing Scripting... [+6803]
15:40:07.335 [Designer-Startup] INFO designer.main - Initializing UI... [+6936]
15:40:07.344 [Designer-Startup] INFO com.inductiveautomation.ignition.designer.querybrowser.QueryBrowser - No defau
set for project. Set default in gateway and relaunch project.
15:40:40.067 [AWT-EventQueue-0] INFO ViewResourceEditor - Perspective remote debugging port open at:
http://localhost:9222/devtools/inspector.html?ws=localhost:9222/devtools/page/(F11B6BD02DD8FBC9B0D6255CF4B1D49)
16:16:30.174 [AWT-EventQueue-0] INFO com.inductiveautomation.ignition.designer.querybrowser.QueryBrowser - No defau
```

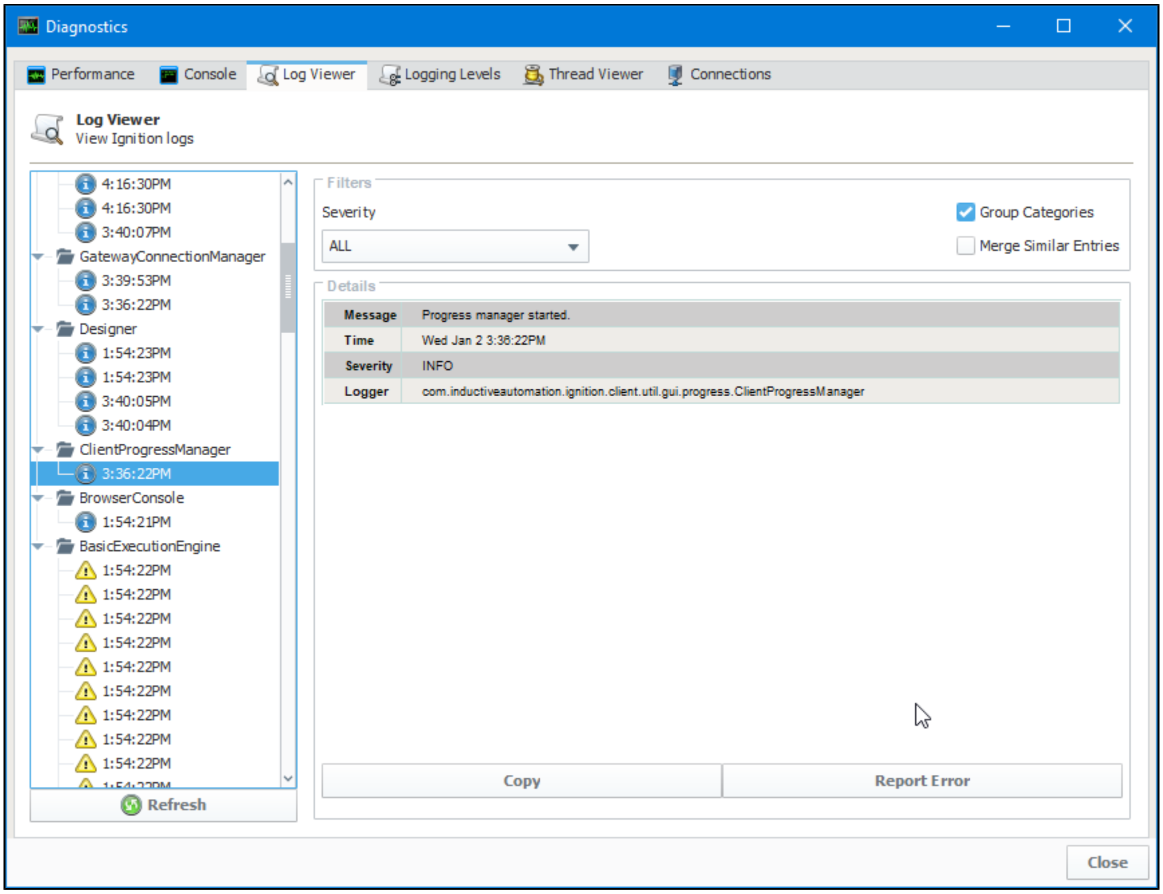
Log Viewer

The Log Viewer tab displays the logged events for the current entity, such as the Designer. Whenever messages occur in the console, they are logged and displayed in this tab. Each message has a logging level associated with it. This is a good place to go when troubleshooting an issue, as any errors shown here may illuminate the cause of the problem.

Logs can be filtered by severity by selecting an option in the Filters dropdown.



To view entries across all categories chronologically, uncheck the **Group Categories** checkbox.



Severity: ALL

☑ Group Categories

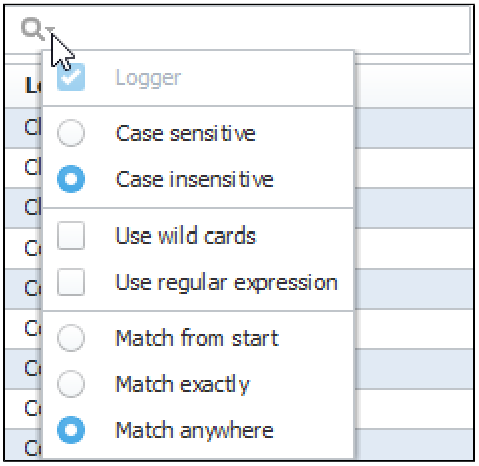
□ Merge Similar Entries

Message	Progress manager started.
Time	Wed Jan 2 3:36:22PM
Severity	INFO
Logger	com.inductiveautomation.ignition.client.util.gui.progress.ClientProgressManager

Logging Levels

The Logging Levels tab displays the list of internal loggers. Most users will not use this tab unless prompted by a technical support representative. Make sure to change logging levels back to info when you are done, otherwise they will flood your console and make it difficult to find any new problems.

A Search box in the upper left enables you to search the loggers. You can also set options to make the search case sensitive, use wildcards, use regular expressions, match from start, match exactly, or match anywhere.



☑ Logger

○ Case sensitive

● Case insensitive

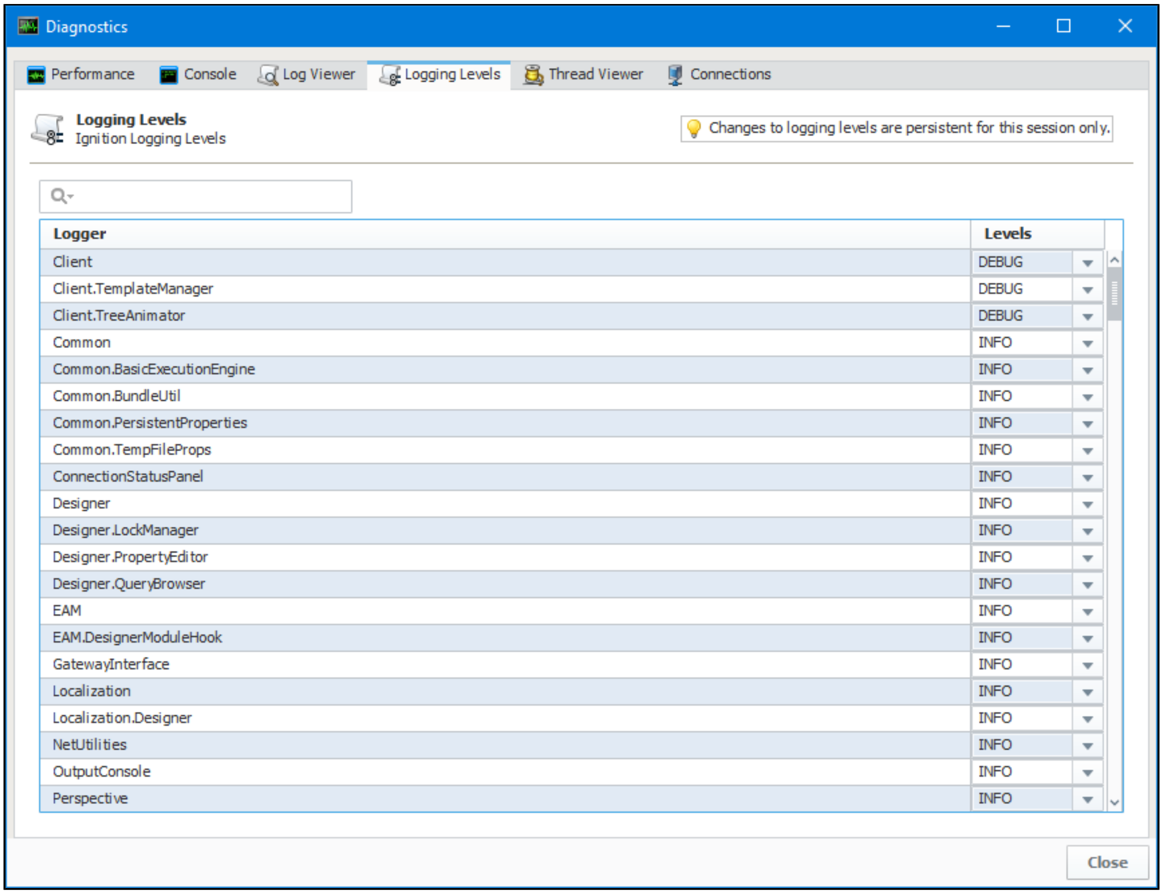
□ Use wild cards

□ Use regular expression

○ Match from start

○ Match exactly

● Match anywhere



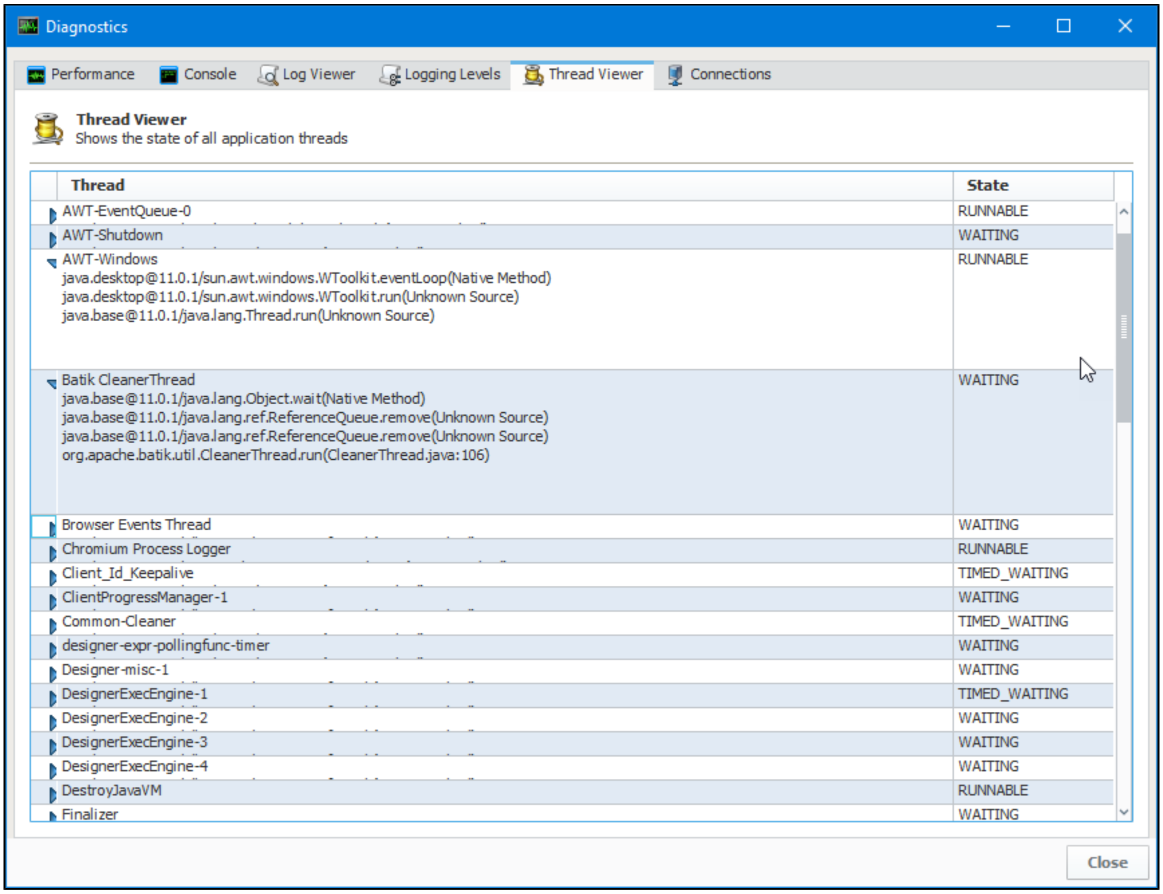
Logging Levels
Ignition Logging Levels

Changes to logging levels are persistent for this session only.

Logger	Levels
Client	DEBUG
Client.TemplateManager	DEBUG
Client.TreeAnimator	DEBUG
Common	INFO
Common.BasicExecutionEngine	INFO
Common.BundleUtil	INFO
Common.PersistentProperties	INFO
Common.TempFileProps	INFO
ConnectionStatusPanel	INFO
Designer	INFO
Designer.LockManager	INFO
Designer.PropertyEditor	INFO
Designer.QueryBrowser	INFO
EAM	INFO
EAM.DesignerModuleHook	INFO
GatewayInterface	INFO
Localization	INFO
Localization.Designer	INFO
NetUtilities	INFO
OutputConsole	INFO
Perspective	INFO

Thread Viewer

The Thread Viewer tab displays information about the currently running threads. Each thread can be expanded by clicking the **Expand** ▶icon or collapsed by clicking the **Collapse** ▲ icon. Most users will not use this tab unless prompted by a technical support representative.

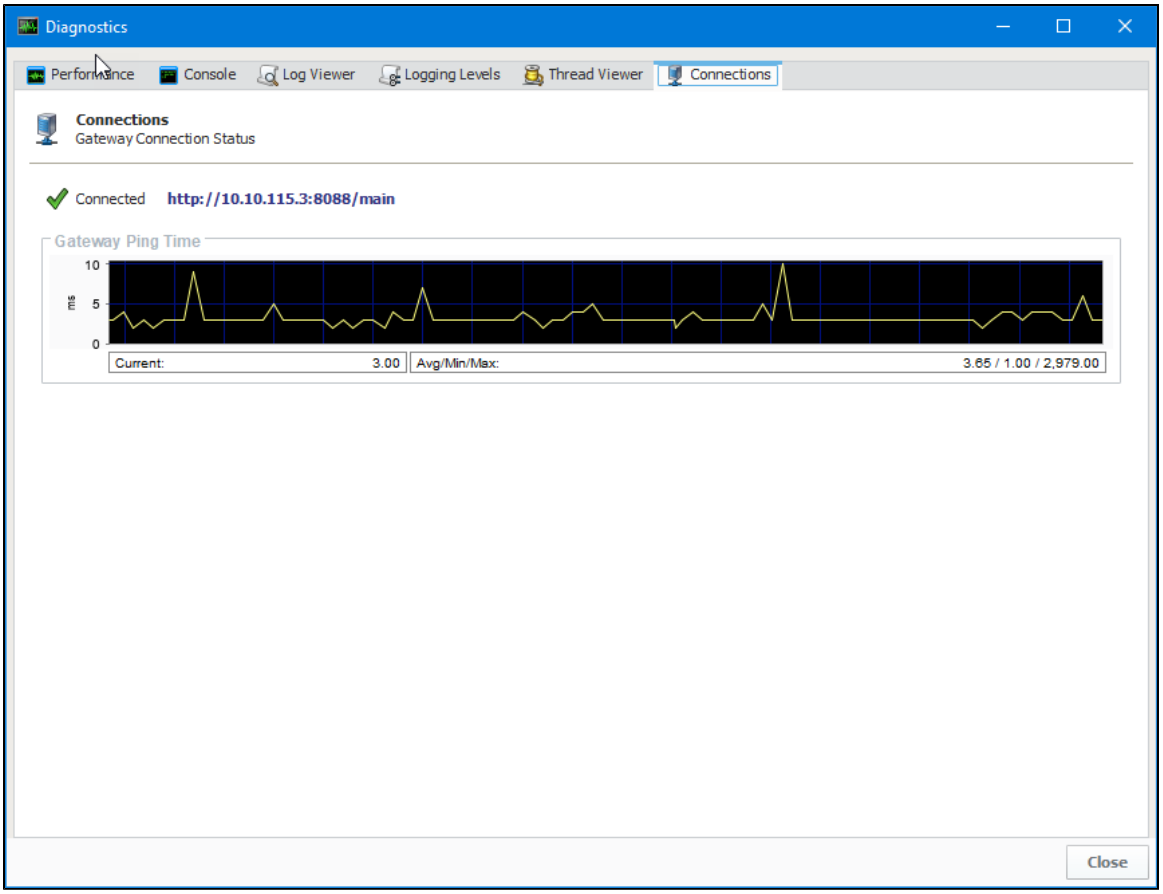


Thread Viewer
Shows the state of all application threads

Thread	State
AWT-EventQueue-0	RUNNABLE
AWT-Shutdown	WAITING
AWT-Windows java.desktop@11.0.1/sun.awt.windows.WToolkit.eventLoop(Native Method) java.desktop@11.0.1/sun.awt.windows.WToolkit.run(Unknown Source) java.base@11.0.1/java.lang.Thread.run(Unknown Source)	RUNNABLE
Batik CleanerThread java.base@11.0.1/java.lang.Object.wait(Native Method) java.base@11.0.1/java.lang.ref.ReferenceQueue.remove(Unknown Source) java.base@11.0.1/java.lang.ref.ReferenceQueue.remove(Unknown Source) org.apache.batik.util.CleanerThread.run(CleanerThread.java:106)	WAITING
Browser Events Thread	WAITING
Chromium Process Logger	RUNNABLE
Client_Id_Keepalive	TIMED_WAITING
ClientProgressManager-1	WAITING
Common-Cleaner	TIMED_WAITING
designer-expr-pollingfunc-timer	WAITING
Designer-misc-1	WAITING
DesignerExecEngine-1	TIMED_WAITING
DesignerExecEngine-2	WAITING
DesignerExecEngine-3	WAITING
DesignerExecEngine-4	WAITING
DestroyJavaVM	RUNNABLE
Finalizer	WAITING

Connections

The Connections tab displays the Gateway connection status as well as a real-time chart of the Gateway Ping time and a summary with the current ping time and average, minimum, and maximum ping times.



Connections
Gateway Connection Status

✓ Connected http://10.10.115.3:8088/main

Current: 3.00

Avg/Min/Max: 3.65 / 1.00 / 2,979.00

Tags

What Is a Tag?

On this page ...

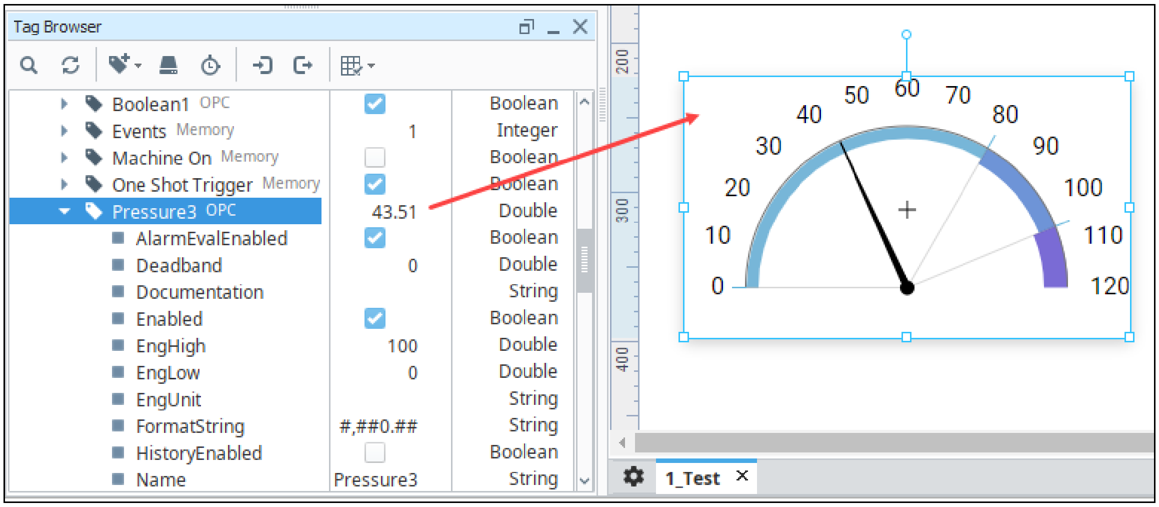
Tags are points of data and may have static values or dynamic values that come from an OPC address, an expression, or a SQL query. The values can be used on screens and in transaction groups.

Tags provide a consistent data model throughout Ignition, and offer the easiest way to get up and running creating real-time status and control systems. Despite their fast initial learning curve, however, Tags offer a great amount of power in system design and configuration. The ability to aggregate Tags from a variety of installations means that you can build widely distributed SCADA systems more easily than ever before with a high level of performance and relatively easy configuration.

While the goal of Tags in Ignition is to create an easy yet powerful model, the variety of options and terminology can sometimes make configuration confusing. Tags are created and controlled using both the Gateway and the Designer for configuration.

- In the Designer, you create or import the Tags. There are several types of Tags such as OPC Tags and Memory Tags. Each Tag has many properties and other functionality such as alarming, history, etc. Once your Tags are created, you can use them in your windows, views, and reports.
- In the Gateway, you create and modify Tag Providers. You can create these Realtime Providers to store groupings of Tags for use in your projects either locally in Ignition or share them externally with connected Gateways. There are also Historian Providers used to store historical data for the Tags, but these are automatically created for each data source you have. These Tag Provider configurations in the Gateway apply globally to all your projects.

The following example shows a pressure Tag in the Tag Browser and a Gauge component in the Designer. The value on the Designer component is bound to the Tag and updates in realtime. This is just a simple example of how Tag values can be represented in your SCADA designs.



Boolean1 OPC	☑	Boolean
Events Memory	1	Integer
Machine On Memory	□	Boolean
One Shot Trigger Memory	☑	Boolean
Pressure3 OPC	43.51	Double
AlarmEvalEnabled	☑	Boolean
Deadband	0	Double
Documentation		String
Enabled	☑	Boolean
EngHigh	100	Double
EngLow	0	Double
EngUnit		String
FormatString	#,##0.##	String
HistoryEnabled	□	Boolean
Name	Pressure3	String

Tag Providers

There are two types of Tag providers; Internal and Remote. By default, a fresh Ignition installation will have an internal Tag provider. This can be thought of as a standard internal Tag database, and stored in the Ignition Gateway. Additionally, it is possible to create Remote Tag Providers, linking one installation of Ignition to the Tags on another Ignition. This ability opens up some very flexible architectures.

Tag User Defined Types

Tag User Defined Types (UDTs) provide an object-oriented approach to Tag building, allowing you to define parameterized data types, extend and override types, and then rapidly generate instances. A change to the type definition is then inherited by all instances, drastically saving time when making routine changes. The UDTs are fully supported by Vision templates, which means you can configure templates for your custom data types and take advantage of drag-and-drop binding to rapidly build complex screens.

Ignition Tags or PLC Tags?

In this manual, we refer to Ignition Tags as simply "Tags." Any mentions of tags from a PLC or another OPC Server will be referred to as "PLC tags" or "OPC tags."

Tag Features

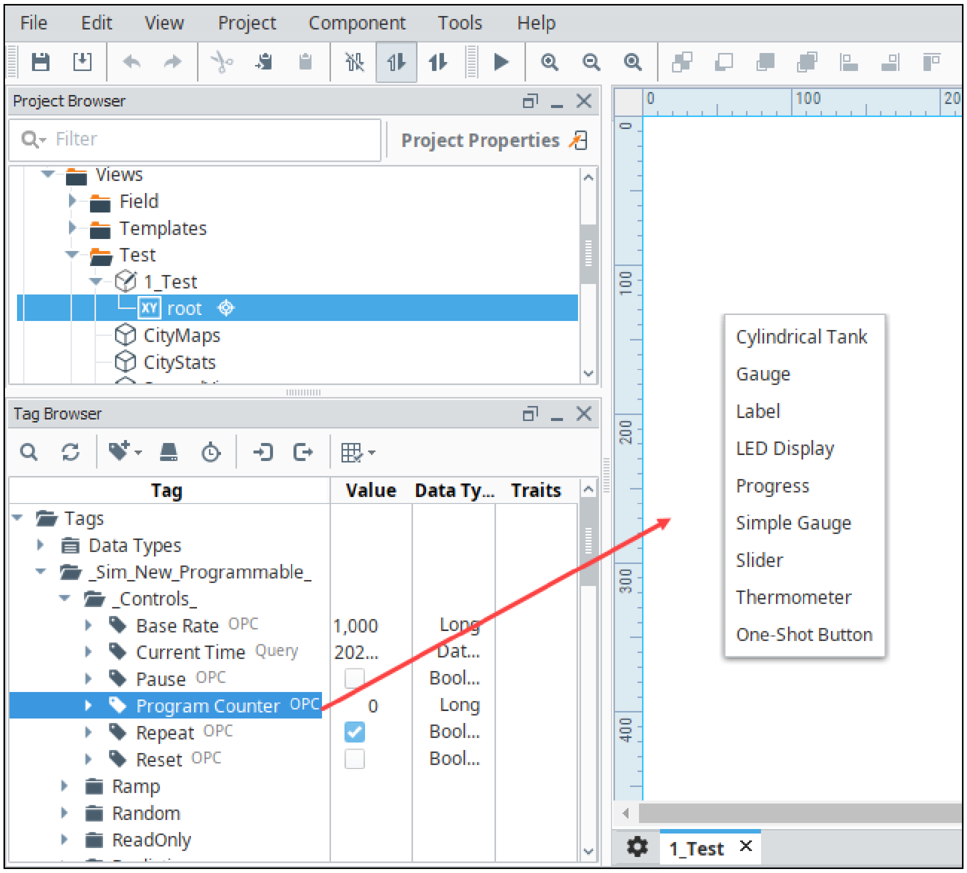
Tags work naturally and easily with Ignition to offer the following features:

- **Performance and Scalability**
  Tags offer great performance on the Gateway, in Perspective Sessions, and in the Vision Client. On the Gateway, the system can support many thousands of value changes per second and millions of Tags. In runtime, Tags improve efficiency with their lightweight subscription architecture. Adding additional Clients creates a nearly negligible effect on the database and the Gateway performance.

- **Object-Oriented Design**
  Use Tag UDTs (User Defined Types) to design re-usable, parameterized, and extendable data types. You can create and configure new instance Tags in seconds, saving a great amount of time over traditional Tag systems.

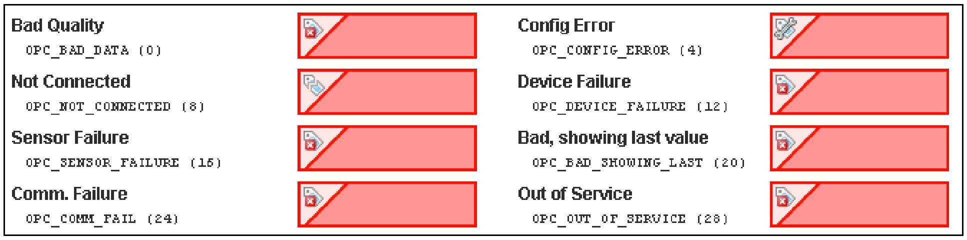
- **Powerful Alarming Model**
  Each Tag can have any number of alarms configured on it. There are many different alarm modes accommodating simple digital alarms, analog high/low value alarms, as well as more specialty alarms like bad data quality and bit-packed alarms. The settings for alarms can bound to other Tags, making the alarm configuration dynamic.

- **Drag-and-Drop Screen Design**
  You can drag and drop Tags onto a window or view to automatically create new bound components. Drag Tags onto existing components or properties to quickly bind them to the data.



- **Historical Logging**
  The Tag Historian Module makes it easier than ever to store and use historical data. When you simply select a check box on a Tag, historical data is stored in an efficient format in your SQL database. This data is then available for querying through scripting, historical bindings, and reporting. Also, you can drag-and-drop Tags directly onto an many components to create trends or display historical values. Tags Historian's robust querying provides you great flexibility in how you retrieve the data.

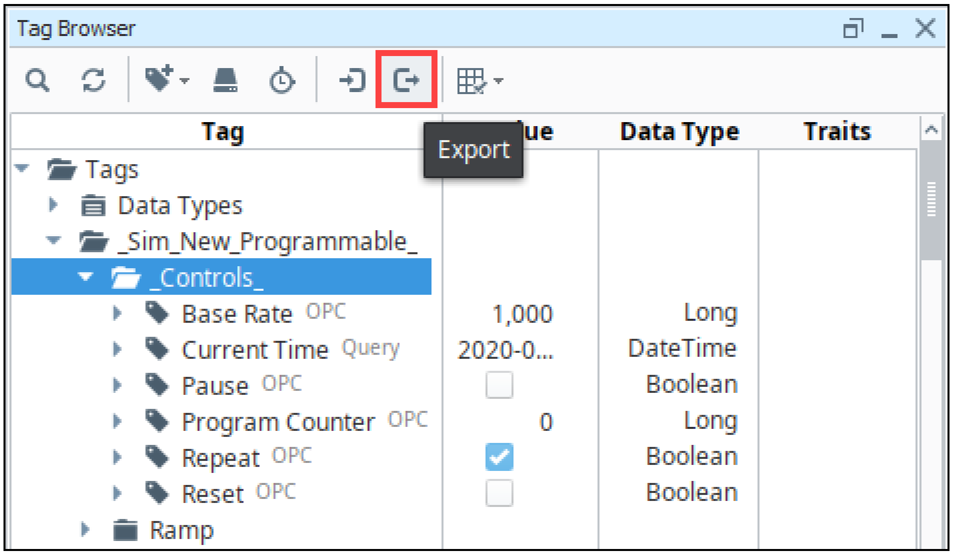
- **Integrated Component Feedback**
  Tags offer a quality and overlay system for components. If a Tag's data quality is anything but good, a component that depends on it gets a visual overlay. Input components display an animated overlay while write pending requests are being written. These features effectively communicate the status of the system at a glance.



Bad Quality OPC_BAD_DATA (0)	🟥	Config Error OPC_CONFIG_ERROR (4)	🟥
Not Connected OPC_NOT_CONNECTED (8)	🟥	Device Failure OPC_DEVICE_FAILURE (12)	🟥
Sensor Failure OPC_SENSOR_FAILURE (16)	🟥	Bad, showing last value OPC_BAD_SHOWING_LAST (20)	🟥
Comm. Failure OPC_COMM_FAIL (24)	🟥	Out of Service OPC_OUT_OF_SERVICE (28)	🟥

Importing and Exporting Tags

Ignition Tags can easily be imported and exported from the Designer by selecting the Tags or folders that you want. See the Exporting and Importing Tags page for more information.



Tag	Value	Data Type	Traits
Tags			
Data Types			
_Sim_New_Programmable_			
Controls			
Base Rate OPC	1,000	Long	
Current Time Query	2020-0...	DateTime	
Pause OPC	☐	Boolean	
Program Counter OPC	0	Long	
Repeat OPC	☑	Boolean	
Reset OPC	☐	Boolean	
Ramp			

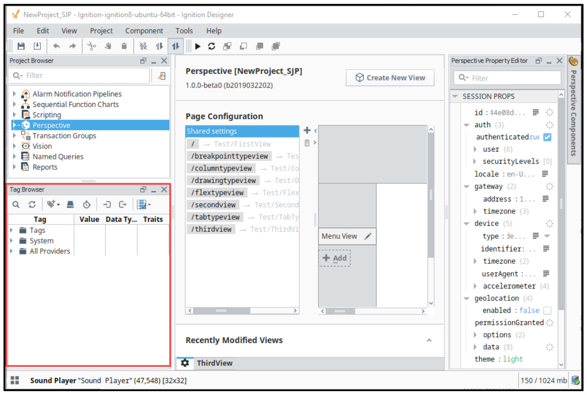
In This Section ...

Understanding Tags

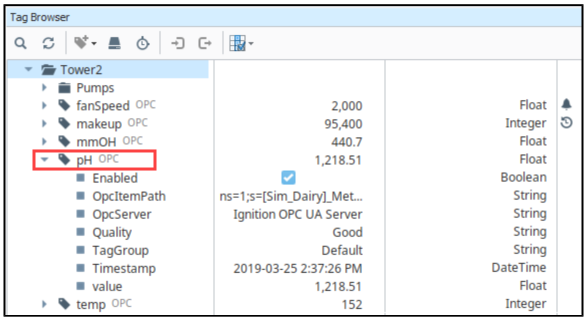
Tags offer a great amount of power in system design and configuration. The ability to aggregate Tags from a variety of installations means that you can build widely distributed SCADA systems more easily than ever before with a high level of performance and relatively easy configuration. The following are key tools and concepts for understanding Tags in Ignition.

Tag Browser

The Tag Browser is the central location for interaction with all types of Tags on your system. It gives you full view of the Tags including the current value, datatype, and any traits. When panels are in their default configuration in the Designer, the Tag Browser appears at the lower left-hand side of the screen.



The Tag Browser is set up in an interactive tree structure with folders that can be expanded or collapsed to view more Tags. Click on the **Expand** ▶ icon to expand any folder or the **Collapse** ▲ icon to collapse the folder. In the example below, the pH Tag for Tower2 has been expanded.



Tower2			
Pumps			
fanSpeed OPC	2,000	Float	🔔
makeup OPC	95,400	Integer	⟲
mmOH OPC	440.7	Float	
pH OPC	1,218.51	Float	
Enabled	☑	Boolean	
OpcItemPath	ns=1;s=[Sim_Dairy]_Met...	String	
OpcServer	Ignition OPC UA Server	String	
Quality	Good	String	
TagGroup	Default	String	
Timestamp	2019-03-25 2:37:26 PM	DateTime	
value	1,218.51	Float	
temp OPC	152	Integer	

Find

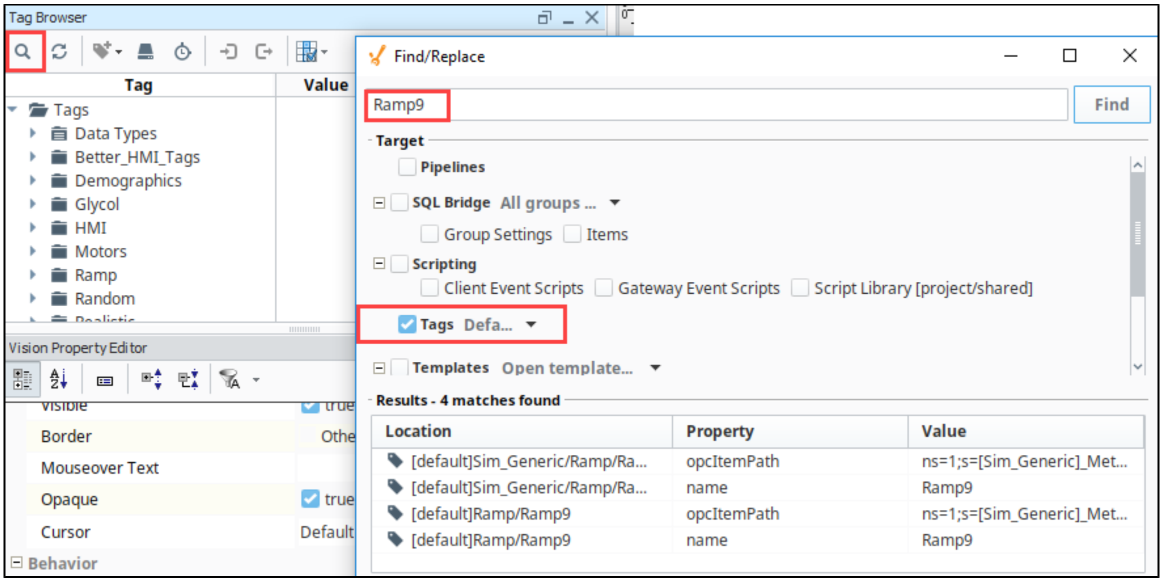
Clicking on the **Search** 🔍icon in the Tag Browser will open up the Designer's global Find/Replace screen. In the the example below, we searched for the Ramp9 Tag and limited the search to the default Tags. Results are shown at the bottom of the screen. For additional information, see Find and Replace.

On this page ...

INDUCTIVE UNIVERSITY

Tags in Ignition

Watch the Video



Ramp9

☑ Tags Defa...

Results - 4 matches found

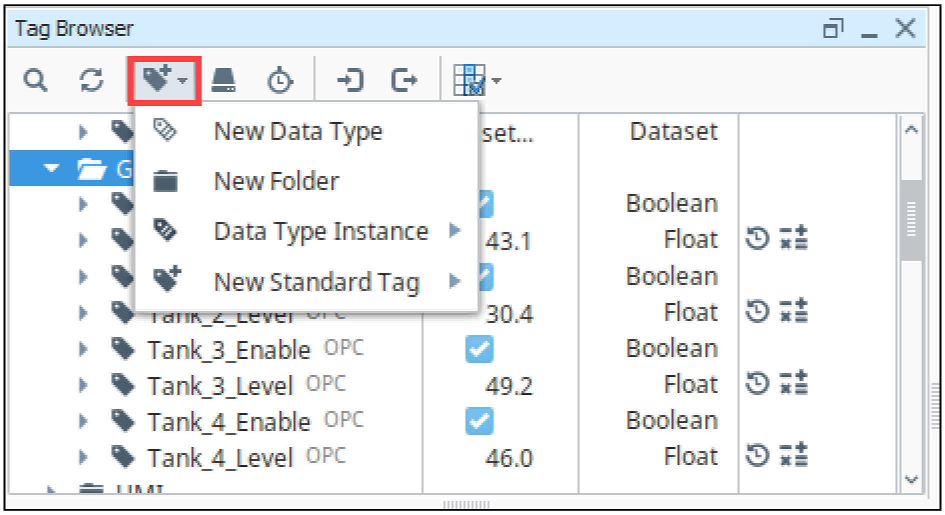
Location	Property	Value
[default]Sim_Generic/Ramp/Ra...	opcItemPath	ns=1;s=[Sim_Generic]_Met...
[default]Sim_Generic/Ramp/Ra...	name	Ramp9
[default]Ramp/Ramp9	opcItemPath	ns=1;s=[Sim_Generic]_Met...
[default]Ramp/Ramp9	name	Ramp9

Refresh Providers

The **Refresh Providers** icon refreshes all of the Providers in the Tag Browser. This is useful if you or others have modified Tags but do not see an update. In general, this button is not used very often.

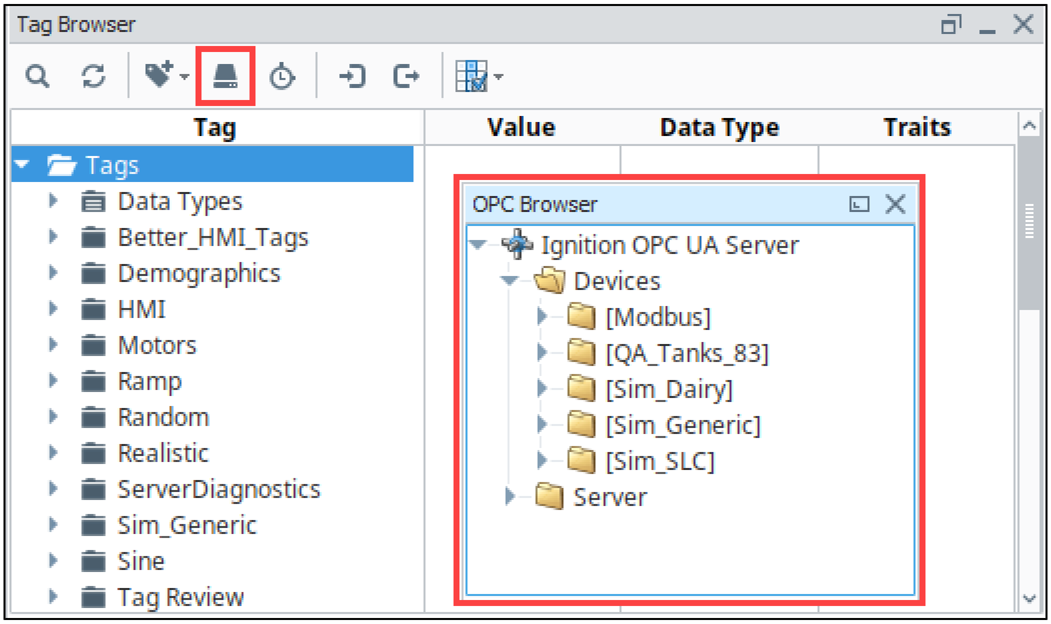
Add Tag

The **New Tag** icon opens a context menu showing all the options to add a Tag, Folder, UDT Instance or a UDT Definition. The new object is added under the Folder you have selected or as a sibling to the Tag you have selected. This button is disabled if there is no selection. See Creating Tags for more information.



OPC Browser

With the OPC Browser, you can browse to find external PLC or OPC Tags. Click the **OPC Server** icon at the top of the Tag Browser to open the OPC Browser. You can then select Tags and drag them to the Tag Browser to be used in the Ignition system. See Creating Tags for more information.



Tag Groups

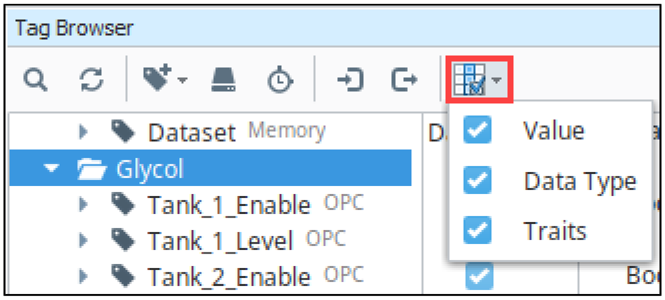
In the Tag Browser, the **Tag Group** icon opens the **Tag Group Editor** window. Tag Groups dictate the rate of execution of Tags. This is where you set up your Tag Groups and scan rates. See Tag Groups for more information.

Import/Export

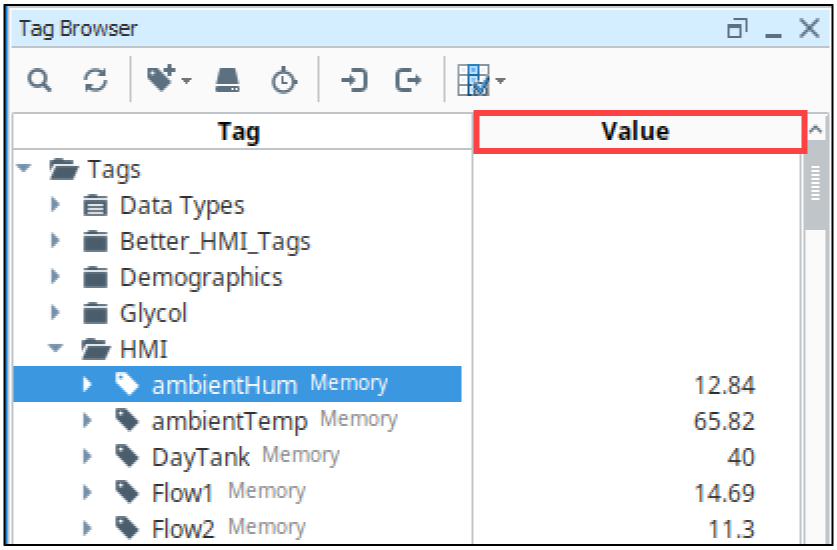
Ignition can export and import Tag configurations to and from the **JSON** (JavaScript Object Notation) file format. Use the **Import** icon or **Export** icon to import and export Tags in this Gateway. See Exporting and Importing Tags for more information.

Column Selector

The Tag Browser displays the Value, Data Type, and Traits columns by default. To toggle any of the options, click on the **Column Selector** icon, then click the checkbox.



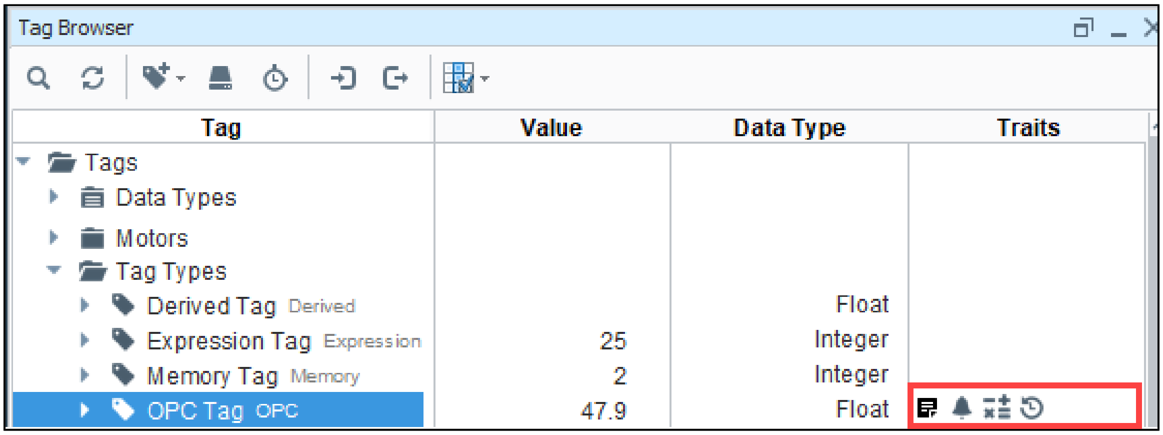
In the example below, we changed the Column Selector so that only the Tag values are shown next to the Tag names.



Tag	Value
Tags	
Data Types	
Better_HMI_Tags	
Demographics	
Glycol	
HMI	
ambientHum Memory	12.84
ambientTemp Memory	65.82
DayTank Memory	40
Flow1 Memory	14.69
Flow2 Memory	11.3

Tag Traits

Certain settings or Tag configurations are visually represented next to the Tag in the Tag Browser.



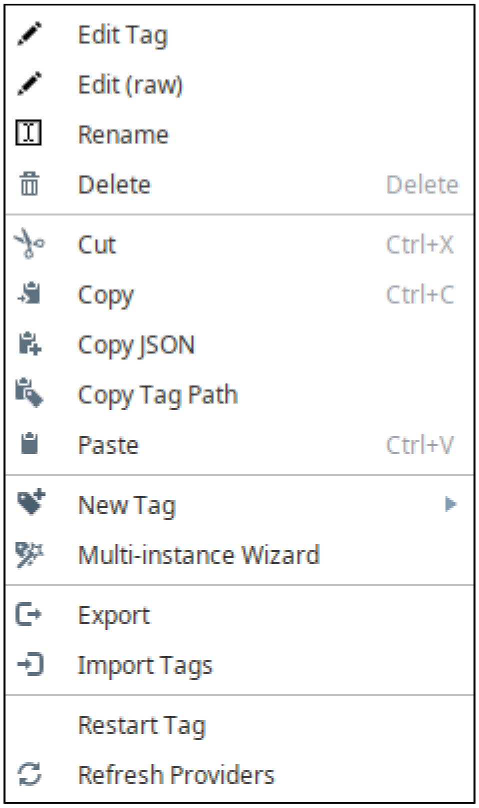
Tag	Value	Data Type	Traits
Tags			
Data Types			
Motors			
Tag Types			
Derived Tag Derived		Float	
Expression Tag Expression	25	Integer	
Memory Tag Memory	2	Integer	
OPC Tag OPC	47.9	Float	📝 🔔 ± 🕘

The following icons enable you to note some important settings on the Tag at a glance. A description of the icons are listed below.

Icon	Setting	Description
±	Scaling	The Scale Mode property under the Numeric Tag Properties section of the Tag Editor has been set to a value other than "Off." The value on the Tag will be scaled to some degree.
🔔	Alarming	At least one alarm has been configured on this Tag.
🕘	Tag History	This Tag has been configured to log data into the Tag Historian system.
📝	Tag Event Script	At least one Tag Event Script has been enabled on this Tag.

Tag Right Click Menu

Editing Tags is done mostly through the Tag Browser. The **Tag Browser** allows you to right click on a Tag or folder to perform any of the following functions. Note that different objects will have different options available. The special **Data Types** folder is slightly different than a regular folder and will have even fewer options. For examples, see Creating Tags.



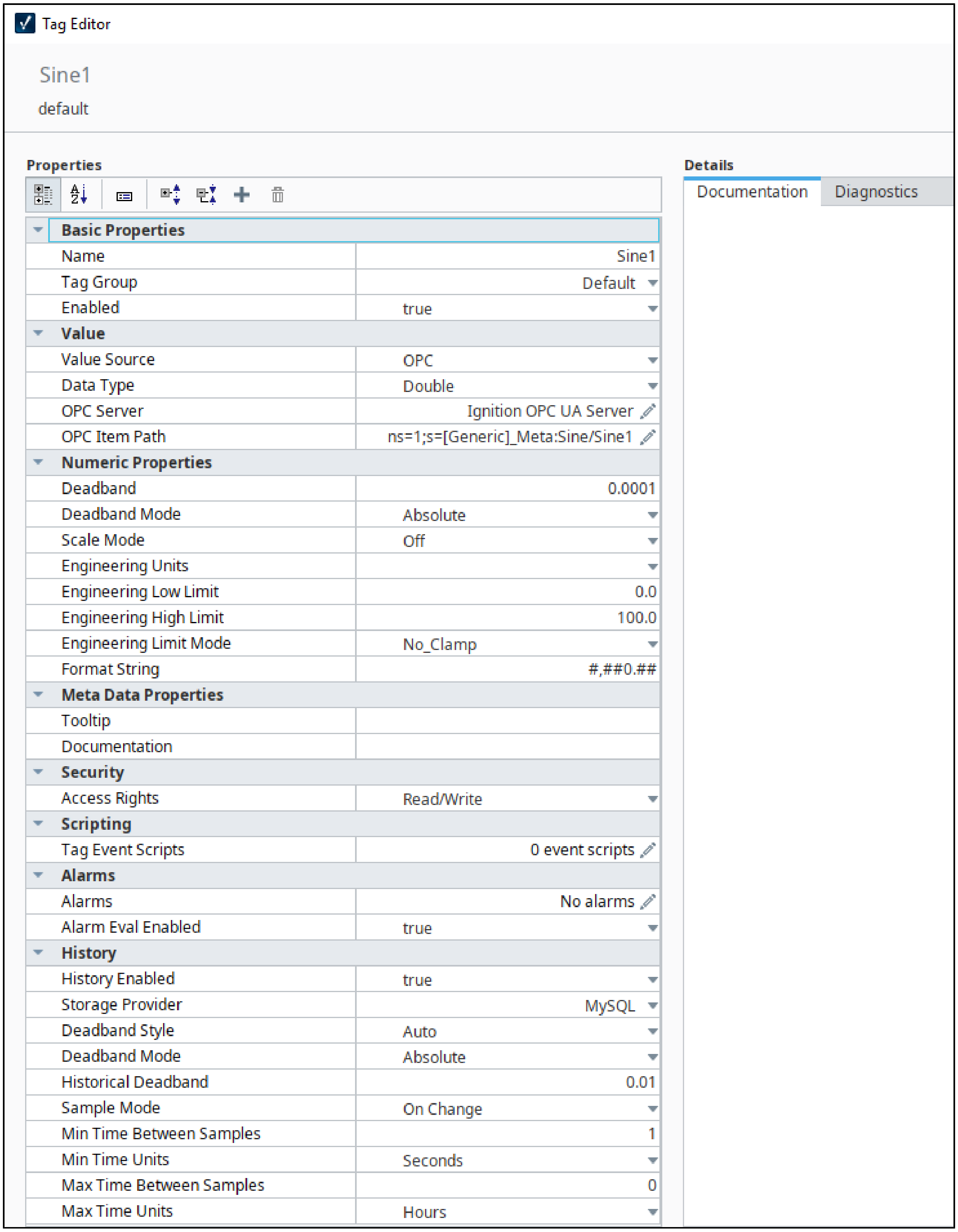
Function	Description
Edit Tag	*Disabled when a Folder is selected.* Opens the Tag Editor window so the Tag can be edited.
Edit (raw)	*Disabled when a Folder is selected.* Editing the JSON code of the Tag.
Rename	Renames the current selection.
Delete	Deletes the current selection.
Cut	Cuts the current selection into the clipboard.
Copy	Copies the current selection into the clipboard.
Copy JSON	The following feature is new in Ignition version **8.0.8** Click here to check out the other new features Available for non-client tags only. Copies the JSON for the selected Tags into the system clipboard. In addition, pasting the JSON into a different provider /designer will create or overwrite Tags.
Copy Tag Path	Copies the currently selected Tag path into the clipboard.
Paste	Pastes the content in the clipboard into the selected context.
New Tag	*Disabled when a Tag is selected.* For Folders, this option opens a sub-menu to create a Tag or Tags.

Function	Description

<table>
<tr><td></td><td>
<table>
<tr><td>New Data Type</td><td>Creates a new data type definition.</td></tr>
<tr><td>New Folder</td><td>Creates a new Tag folder.</td></tr>
<tr><td>Data Type Instance</td><td>Creates a new instance of an existing data type. The instance is linked to the parent type so when the parent changes, the instances are overwritten with the parent type changes.
Sub Menu - based on Data Types</td></tr>
<tr><td>New Standard Tag</td><td>Creates different types of Tags such as Derived, Expression, Memory, OPC, Query, and Reference Tags.
Sub Menu - Standard Tag Types</td></tr>
</table>
</td></tr>
<tr><td>Multi-instance Wizard</td><td>Creates many instances of a UDT at the same time.</td></tr>
<tr><td>Export</td><td>Exports the selected Tags.</td></tr>
<tr><td>Import Tags</td><td>Imports Tags into the project.</td></tr>
<tr><td>Go To Definition</td><td><i>Removed unless a UDT or UDT Member is selected.</i>
Finds the UDT definition used in the UDT instance.</td></tr>
<tr><td>Restart Tag</td><td>Attempts to start / restart the selected Tag. If a folder is restarted, then all tags under the folder will restart.</td></tr>
<tr><td>Refresh Providers</td><td>Refreshes the list of Tag providers.</td></tr>
</table>

Tag Editor

The Tag Editor is robust interface that contains all the properties that can be configured for Tags. In the Tag Editor, you set the Tag's name, value, numeric and meta data properties, security, alarming, history, and more. For more information, see Tag Properties.



Sine1

default

Properties

Basic Properties	
Name	Sine1
Tag Group	Default
Enabled	true
Value	
Value Source	OPC
Data Type	Double
OPC Server	Ignition OPC UA Server
OPC Item Path	ns=1;s=[Generic]_Meta:Sine/Sine1
Numeric Properties	
Deadband	0.0001
Deadband Mode	Absolute
Scale Mode	Off
Engineering Units	
Engineering Low Limit	0.0
Engineering High Limit	100.0
Engineering Limit Mode	No_Clamp
Format String	#,##0.##
Meta Data Properties	
Tooltip	
Documentation	
Security	
Access Rights	Read/Write
Scripting	
Tag Event Scripts	0 event scripts
Alarms	
Alarms	No alarms
Alarm Eval Enabled	true
History	
History Enabled	true
Storage Provider	MySQL
Deadband Style	Auto
Deadband Mode	Absolute
Historical Deadband	0.01
Sample Mode	On Change
Min Time Between Samples	1
Min Time Units	Seconds
Max Time Between Samples	0
Max Time Units	Hours

Tag Naming

Tags names are flexible and to not have to match data source names (like an OPC path) or tag codes (such as N7, F8, etc.). It is not necessary that Tag's name be related at all to its underlying data source (OPC path, for instance). This provides a level of indirection that is convenient for systems with many repeat Tag structures.

It is important to give Tags a meaningful structure and arrange them in hierarchical Tag folders so that they are easy to understand, identify, and locate for all developers. By default, Ignition Tags are named after their OPC Server address when a Tag is dragged into the Tag Browser. You can change this name to just about anything that you want. We recommend using names that mean something to your process, such as "Motor 3 Amps." Alternatively you could create folders in your Tag Browser such as "Motor 3/Amps.". When renaming Tags and folders, there is really only one question to ask: "does this structure make sense?"

Another important concept to consider when naming and organizing your Tags, is to do this early in your project. If you rename or move any of your Tags to another folder, and your Tag is being used in other places, chances are you are going to break the reference to the Tag on your screen. So keeping your Tags organized and defining your Tag structure early on in your project is critical.

When you choose a new name for your Tags and folders, there are some rules that must be followed. The first character of the Tag name must be one of the following:

- Any alphanumeric
- Any valid unicode letter
- An underscore

The second character, and every character after that can then be one of the following:

- Any alphanumeric
- Any valid unicode letter
- An underscore
- A space
- Any of the following special characters:

'	-	:	(	)

All other special characters can not be used in a Tag name.

Working with Tags

Once you have Tags in your Tag Browser, they automatically start executing. The Designer is live, and in the Tag Browser you can see your Tags update in real time. The most commonly used type of Tag is an OPC Tag, which gets its value from a device like an OPC Server. You can connect to just about any type of device out there and show the data on screen, as history, and write back whenever you want. Tags make Ignition extremely flexible and easy to show data. See the Quick Start Guide to get some tags created and in your client.

Click these links to find out about

- Types of Tags
- Creating Tags
- Exporting and Importing Tags

Once you have some Tags created, you can edit them to include even more features. You can:

- Set up Alarms
- Store Historical data
- Add Security
- Add Scripting to Tag Events

In addition to all that, Ignition allows you to create UDTs (User Defined Types) out of your Tags to rapidly develop projects with structured objects. This gives you the ability to create a master type then quickly add many instances. UDTs allow you to make a change in one place and see it automatically propagated to every instance.

Related Topics ...

- Creating Tags
- User Defined Types - UDTs
- Tag Groups

In This Section ...

Types of Tags

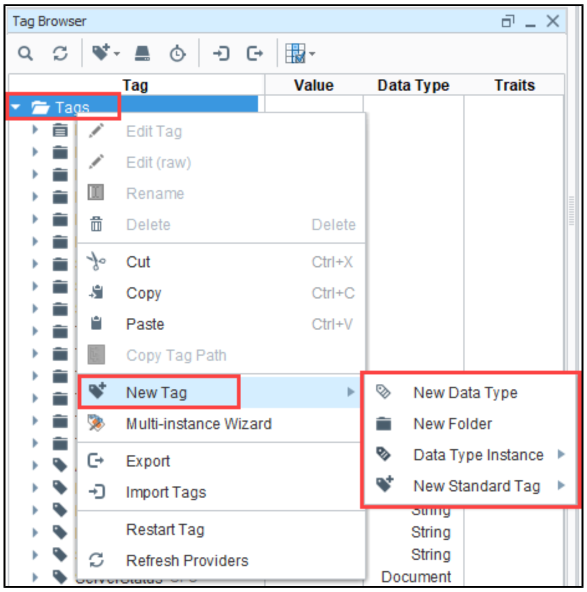
There are several different types of Tags you can create in Ignition: standard Tags, System Tags, and Vision Client Tags. All these Tag types are available in the Tag Browser.

Tags executed in the Gateway support all of the primary features of Tags: scaling, alarming, history, and role-based permissions. These Tags run in the Gateway, and the values of the Tags are shared among all running sessions and clients. They are identical in their configurations, apart from defining how the value is generated.

On this page ...

Tag Browser

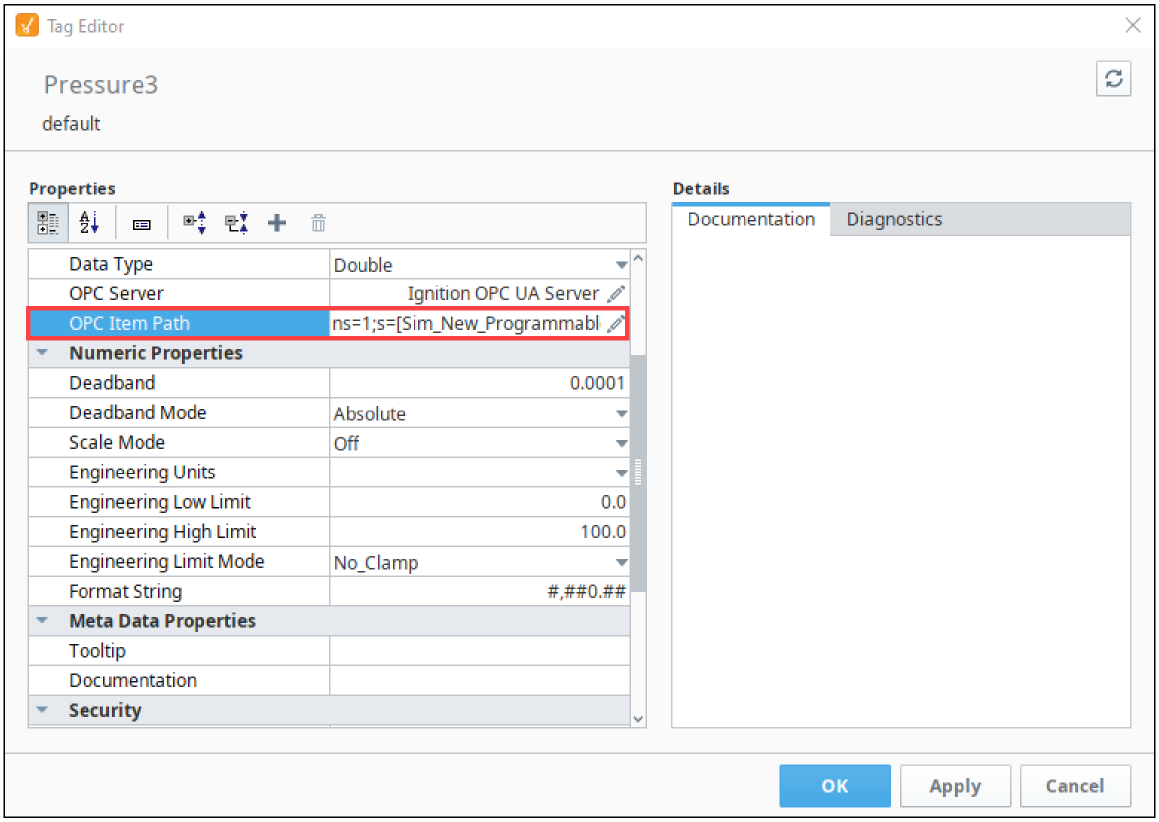
You can create Tags in the **Tag Browser** by right clicking on the Tags folder, scroll down to **New Tag** and select the desired type of Tag.



OPC Tags

An OPC Tag is driven by an OPC Item Path and OPC server. The OPC Item Path is a string path to a particular device connection. The exact path is defined by the driver and OPC server used to communicate with the device. Many drivers support browsing, allowing you to automatically create OPC Tags by dragging-and-dropping from the OPC Browser. However, in cases where browsing isn't supported, OPC Tags can manually be created.

In the **Tag Browser**, double click on any existing OPC Tag, to see the the **OPC Server** name and **Item Path**.



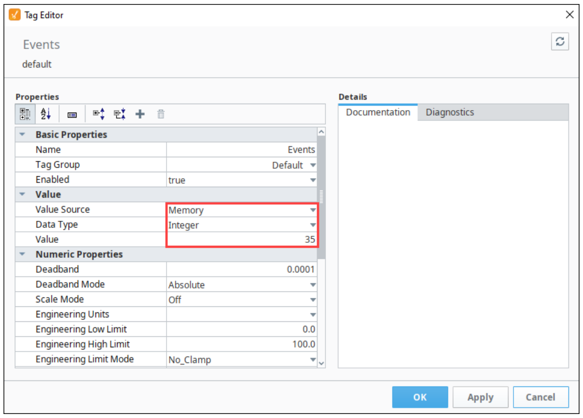
Pressure3

default

Data Type	Double
OPC Server	Ignition OPC UA Server
OPC Item Path	ns=1;s=[Sim_New_Programmabl
Numeric Properties	
Deadband	0.0001
Deadband Mode	Absolute
Scale Mode	Off
Engineering Units	
Engineering Low Limit	0.0
Engineering High Limit	100.0
Engineering Limit Mode	No_Clamp
Format String	#,##0.##
Meta Data Properties	
Tooltip	
Documentation	
Security	

Memory Tags

Memory Tags are simple tags, that do not automatically poll or update their value. They hold the same value until some other entity (script, binding, or some other user-created mechanism) changes their value. They're useful in situations where a value must be stored outside of a PLC or database.



Events

default

Basic Properties	
Name	Events
Tag Group	Default
Enabled	true
Value	
Value Source	Memory
Data Type	Integer
Value	35
Numeric Properties	
Deadband	0.0001
Deadband Mode	Absolute
Scale Mode	Off
Engineering Units	
Engineering Low Limit	0.0
Engineering High Limit	100.0
Engineering Limit Mode	No_Clamp

INDUCTIVE UNIVERSITY

Memory Tags

Watch the Video

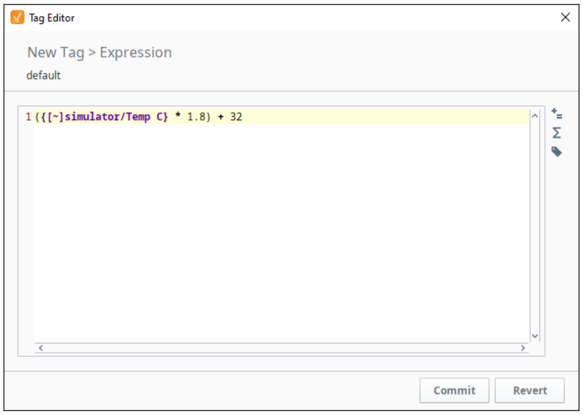
Expression Tags

The Expression Tags are driven by an expression, and its values are derived from a calculation. The expression syntax is the same as for property bindings, and allows mathematical operations, references to other Tags, logic operations, functions, and more. This expression example converts Celsius to Fahrenheit.

INDUCTIVE UNIVERSITY

Expression Tags

Watch the Video


Tag Editor

New Tag > Expression

default

```
({[~]simulator/Temp C} * 1.8) + 32
```

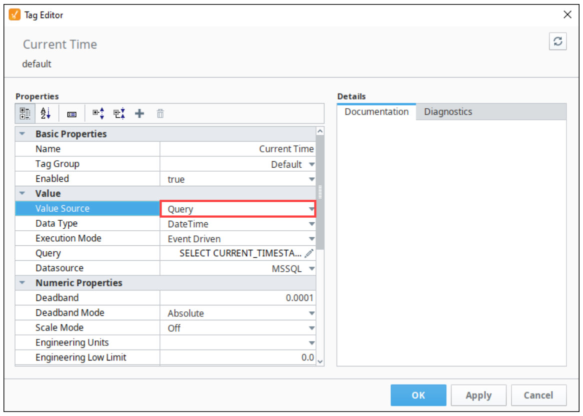
Query Tags

The Query Tags execute a SQL Query, the result of that query are returned to the value on the tag. Query Tags can reference other Tags to build dynamic queries. This SQL query example is getting the current timestamp.

> Each query Tag will run a separate query, so large numbers of this type of Tag will dramatically increase the number of queries running against the database. It is highly recommended to attempt to optimize and restrict the number of query Tags. Instead of having several query Tags pulling values from the same table, have one query Tag configured to a Dataset data type that returns large portions of the table, and use several expression Tags to extract values from the individual elements in the dataset.

Query Tags

Watch the Video

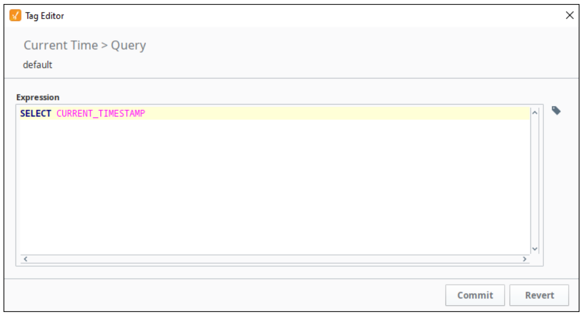

Tag Editor

Current Time

default

Properties

Basic Properties	
Name	Current Time
Tag Group	Default
Enabled	true
Value	
Value Source	Query
Data Type	DateTime
Execution Mode	Event Driven
Query	SELECT CURRENT_TIMESTA...
Datasource	MSSQL
Numeric Properties	
Deadband	0.0001
Deadband Mode	Absolute
Scale Mode	Off
Engineering Units	
Engineering Low Limit	0.0


Tag Editor

Current Time > Query

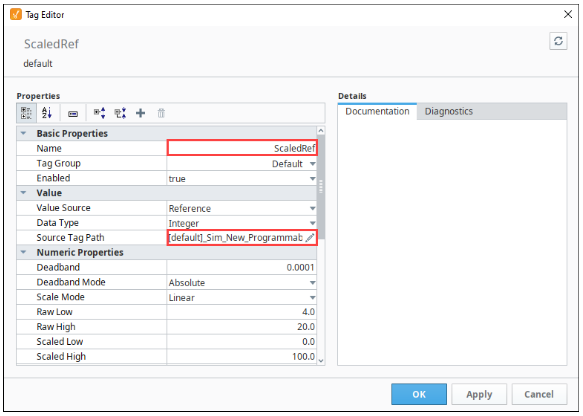
default

Expression

```
SELECT CURRENT_TIMESTAMP
```

Reference Tags

A Reference Tag simply refers to an existing tag. For example, it can be used to create an alias for the tag or create a scaled version of a tag. The Reference tag can write back to the original Tag.



ScaledRef
default

Basic Properties	
Name	ScaledRef
Tag Group	Default
Enabled	true
Value	
Value Source	Reference
Data Type	Integer
Source Tag Path	[default]_Sim_New_Programmab
Numeric Properties	
Deadband	0.0001
Deadband Mode	Absolute
Scale Mode	Linear
Raw Low	4.0
Raw High	20.0
Scaled Low	0.0
Scaled High	100.0

Reference Tags

Watch the Video

Derived Tags

A Derived Tag is an abstracted Tag that refers to another Tag. They are similar conceptually to Expressions Tags and Reference Tags, but Derived Tags have some additional functionality. Namely, they can apply a separate expression to an incoming write, before writing back to the source tag.

INDUCTIVE UNIVERSITY

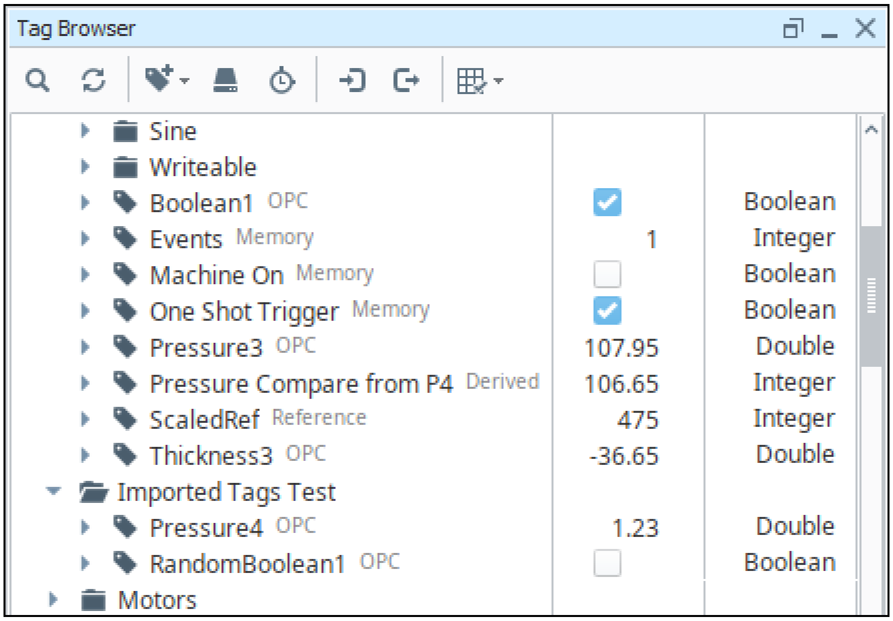
Derived Tags

Watch the Video

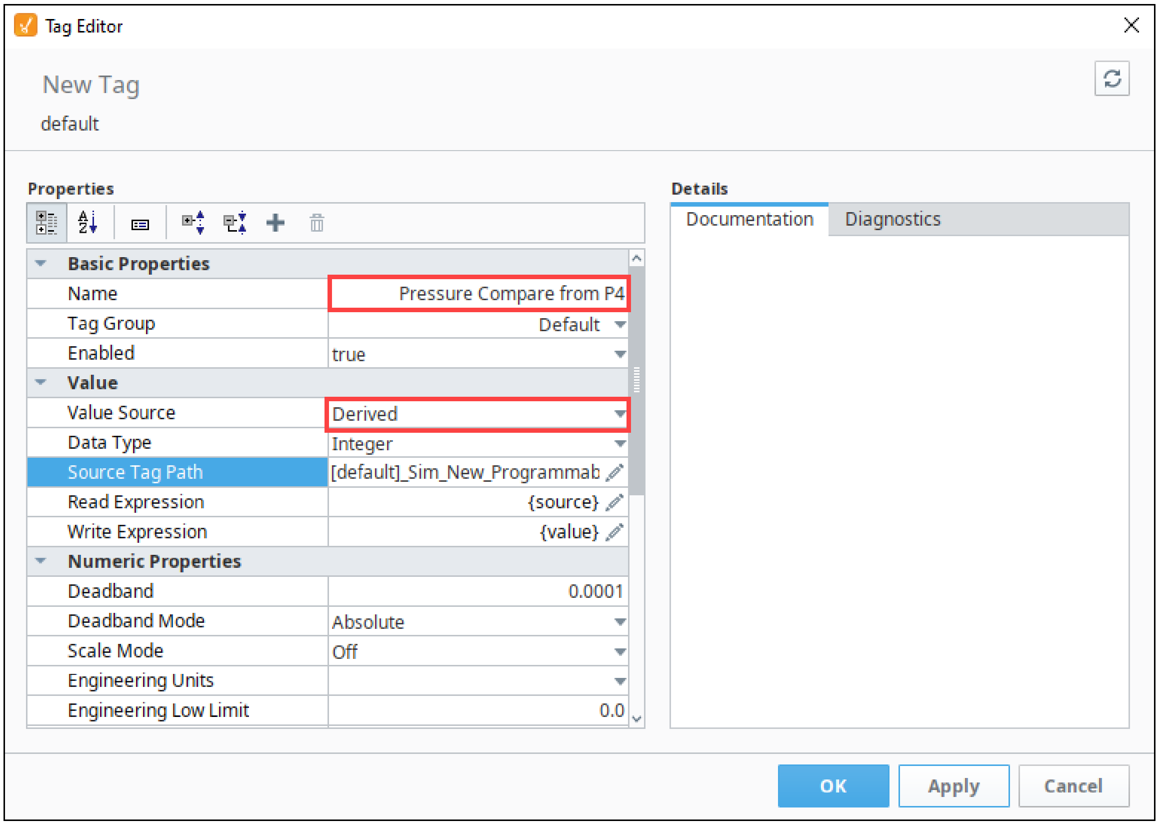
Source Tag Path

The Source Tag Path property of a Derived Tag depicts which Tag the Derived Tag should reference. This is similar to the OPC Item Path on an OPC Tag, except it points to an Ignition Tag.

The Derived Tag will update at the rate of the source Tag.



Sine		
Writeable		
Boolean1 OPC	☑	Boolean
Events Memory	1	Integer
Machine On Memory	☐	Boolean
One Shot Trigger Memory	☑	Boolean
Pressure3 OPC	107.95	Double
Pressure Compare from P4 Derived	106.65	Integer
ScaledRef Reference	475	Integer
Thickness3 OPC	-36.65	Double
Imported Tags Test		
Pressure4 OPC	1.23	Double
RandomBoolean1 OPC	☐	Boolean
Motors		

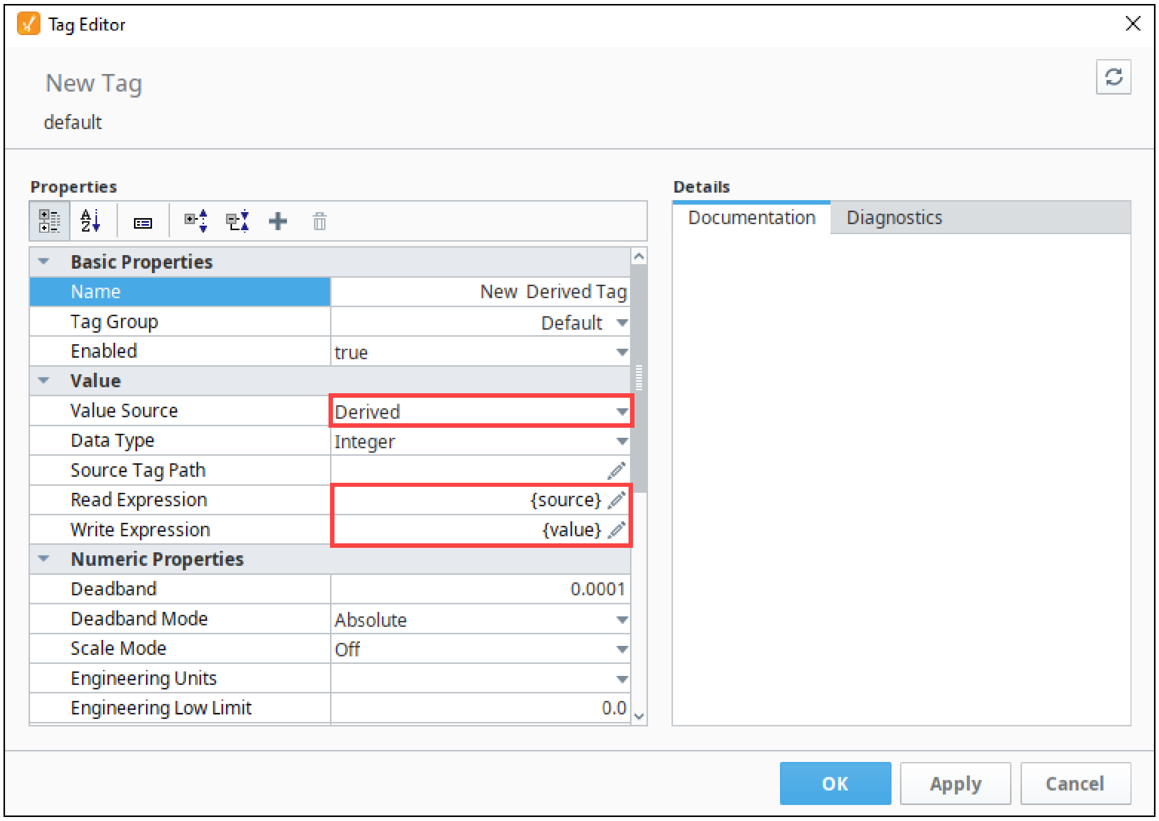


New Tag
default

Basic Properties	
Name	Pressure Compare from P4
Tag Group	Default
Enabled	true
Value	
Value Source	Derived
Data Type	Integer
Source Tag Path	[default]_Sim_New_Programmab
Read Expression	{source}
Write Expression	{value}
Numeric Properties	
Deadband	0.0001
Deadband Mode	Absolute
Scale Mode	Off
Engineering Units	
Engineering Low Limit	0.0

Value Derivation

Derived Tags have a Value section in the Tag Editor that contains two options that enable you to leverage Expressions when reading from or writing to the source Tag. This provides a greater degree of customization than the traditional Numeric scaling settings available to OPC Tags.



New Tag
default

Basic Properties	
Name	New Derived Tag
Tag Group	Default
Enabled	true
Value	
Value Source	Derived
Data Type	Integer
Source Tag Path	
Read Expression	{source}
Write Expression	{value}
Numeric Properties	
Deadband	0.0001
Deadband Mode	Absolute
Scale Mode	Off
Engineering Units	
Engineering Low Limit	0.0

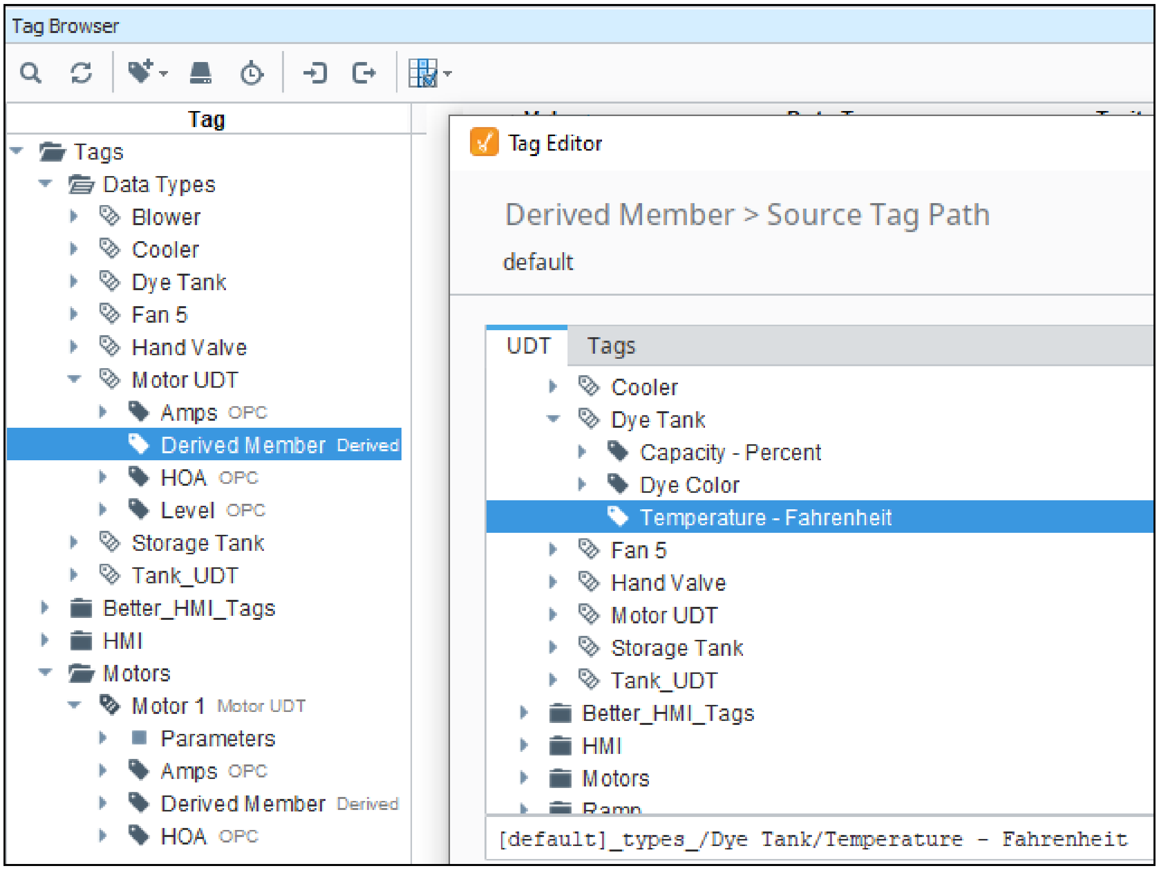
Read Expression	Determines what value should appear on the Derived Tag. The current value of the source Tag may be referenced with the **{source}** reference.
Write Expression	When writing to the Derived Tag, this expression determines what value should be written to the source Tag. The current value of the source Tag may be referenced with the **{source}** reference. The raw value passed to the Derived Tag may be referenced with the **{value}** reference.

This interface provides an opportunity to scale the written and read value. For example, if the source Tag was in Fahrenheit, and the derived Tag should be Celsius, we could use the following expressions:

```
//Read Expression
({source}-32)*(5/9)

//Write Expression
{value}*(9/5) + 32
```

Derived Tags in UDTs

When added as a member of a UDT, Derived Tags may reference peer members. Additionally, UDT parameters may be used in the Source Tag Path.



[default]_types_/Dye Tank/Temperature - Fahrenheit

Changing the Source

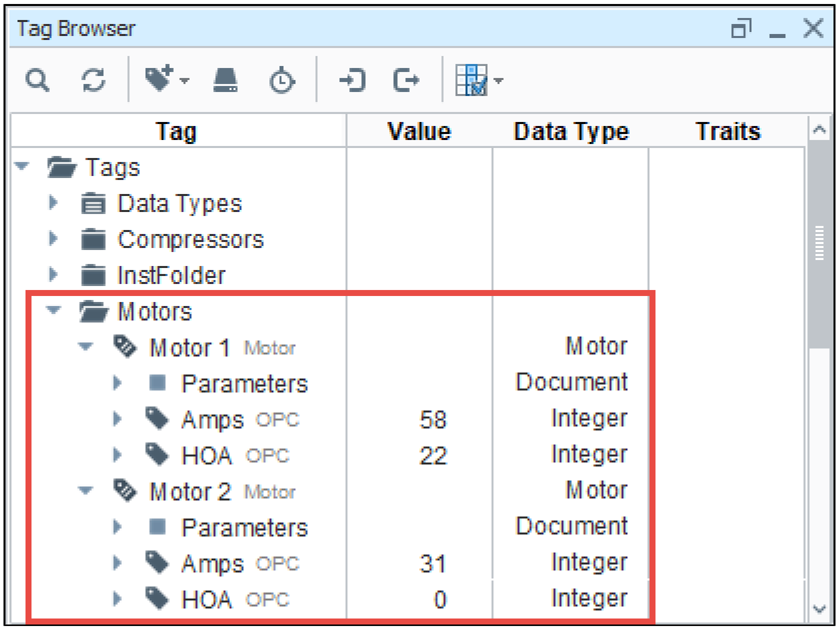
Aside from editing the Tag from the Designer, the Source Tag Path on a Derived Tag may be modified by writing directly to the **SourceTagPath** property on the Tag, or using a Tag Binding on a Vision component.

Additionally, the **SourceTagPath** property may be changed through scripting:

```
#Example of writing to the SourceTagPath attribute via Python Script
system.tag.writeBlocking(["Derived Example/Derived Tag.SourceTagPath"], ["[.]Folder/New Source Tag"])
```

User Defined Types (UDTs)

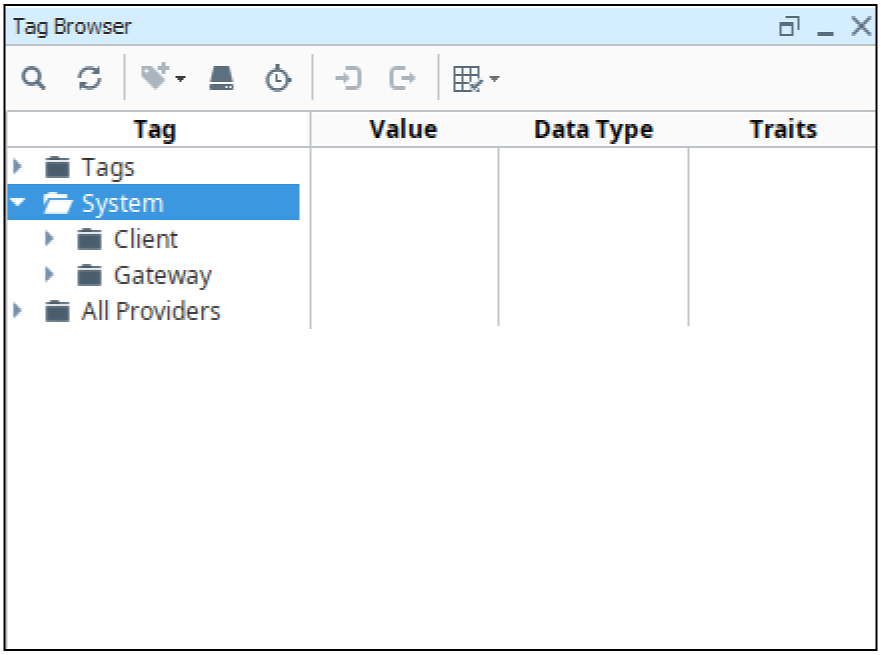
UDTs are created out of standard Tag types, but they offer a variety of additional features. You can think of them as a way to create "data templates", where a particular structure of Tags is defined, and can then be created as if it were a single Tag. This UDT example shows two Motor instances, the data type Motor, and all the Parameters and Tags that make up the structure (i.e., Amps and HOA). For more information, see User Defined Types - UDTs.



Tag	Value	Data Type	Traits
Tags			
Data Types			
Compressors			
InstFolder			
Motors			
Motor 1 Motor		Motor	
Parameters		Document	
Amps OPC	58	Integer	
HOA OPC	22	Integer	
Motor 2 Motor		Motor	
Parameters		Document	
Amps OPC	31	Integer	
HOA OPC	0	Integer	

System Tags

System Tags provide status about the Ignition system, such as memory usage, performance metrics, and so on. In the Tag Browser, under the System folder, there are folders for the Client System Tags and Gateway System Tags. The scope for each is slightly different. You can modify the Gateway System Tags to perform alarming, history, and scaling, but you cannot modify the Client System Tags. For more information, see System Tags.



Vision Client Tags

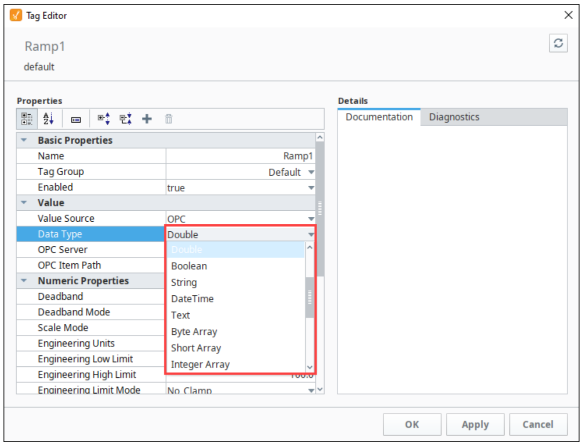
Within the Vision Module, you can also have Vision Client Tags that are specific to a Vision Client. Their values are isolated to a Client runtime . For more information, see Vision Client Tags.

Related Topics ...

- Creating Tags
- System Tags
- User Defined Types - UDTs

Tag Data Types

The Tag Editor has a dropdown list of options for Tag data types. It is important that this be set as correctly as possible with regards to the Tag's underlying data source. The Tags system will attempt to coerce any raw incoming value (for example, from OPC or a SQL query) into the desired type.



Name: Ramp1
Tag Group: Default
Enabled: true
Value Source: OPC
Data Type: Double
OPC Server
OPC Item Path
Deadband
Deadband Mode
Scale Mode
Engineering Units
Engineering Low Limit
Engineering High Limit: 100.0
Engineering Limit Mode: No Clamp

On this page ...

INDUCTIVE UNIVERSITY

Array and Dataset Tags

Watch the Video

The following table lists all the data types available for Tags in Ignition.

Datatype	String Value	Integer Value
Byte	`Int1`	`0`
Short	`Int2`	`1`
Integer	`Int4`	`2`
Long	`Int8`	`3`
Float	`Float4`	`4`
Double	`Float8`	`5`
Boolean	`Boolean`	`6`
String	`String`	`7`
DateTime	`DateTime`	`8`
Text	`Text`	`10`
Byte Array	`Int1Array`	`17`

Short Array	`Int2Array`	18
Integer Array	`Int4Array`	11
Long Array	`Int8Array`	12
Float Array	`Float4Array`	19
Double Array	`Float8Array`	13
Boolean Array	`BooleanArray`	14
String Array	`StringArray`	15
DateTime Array	DateTimeArray	16
Binary Data	ByteArray	20
Dataset	`DataSet`	9
Document	Document	29

Array and Dataset Data Types

The Array and Dataset data types available on Tags allow for multiple data points to be stored in a single Tag. Configuring a Tag as an array or dataset is as easy as changing the data type in the Tag Editor.

Array Tags

Many OPC servers and drivers already support array type Tags, and now each element in the array can easily be represented with the array data types in Ignition. Additionally, array data types can be used with devices that do not support array types, and will instead expose each bit in the value.

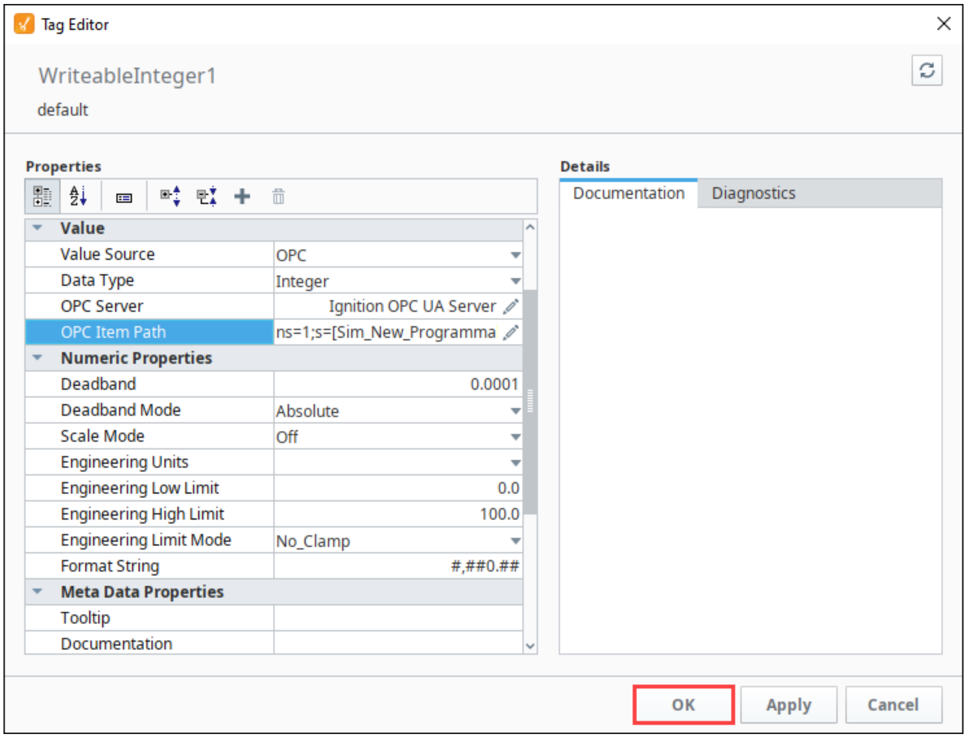
Array Tag Write-Back

OPC Array Tags now support writing back to the device. How this is done can vary, depending on the type of OPC Server in use. Some OPC Servers support writes to individual array elements, where a write would occur just like any other Tag write. However, some OPC Servers do not support individual element writes, which means the whole array will need to be written back to the array tag, even if only a single element is changing.

Array Alarm Example

Because the core data type of each element in the array is the same, it is possible to apply Tag History, Alarming, or Scaling configurations onto the array, and these configurations will be inherited by each element.

1. For this example, create the Tag WriteableInteger1.


Tag Editor

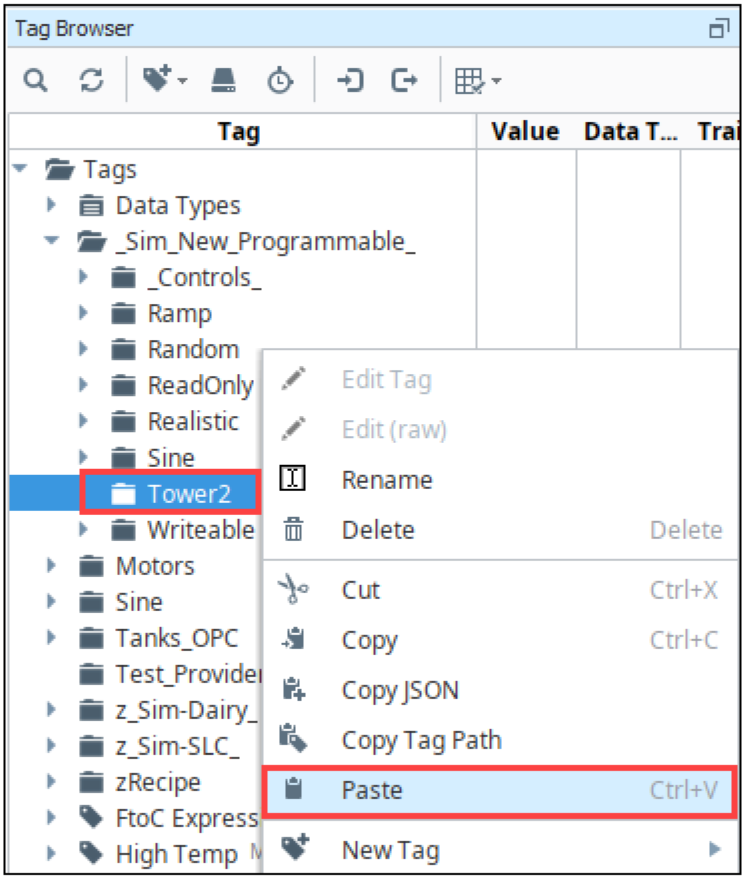
WriteableInteger1

default

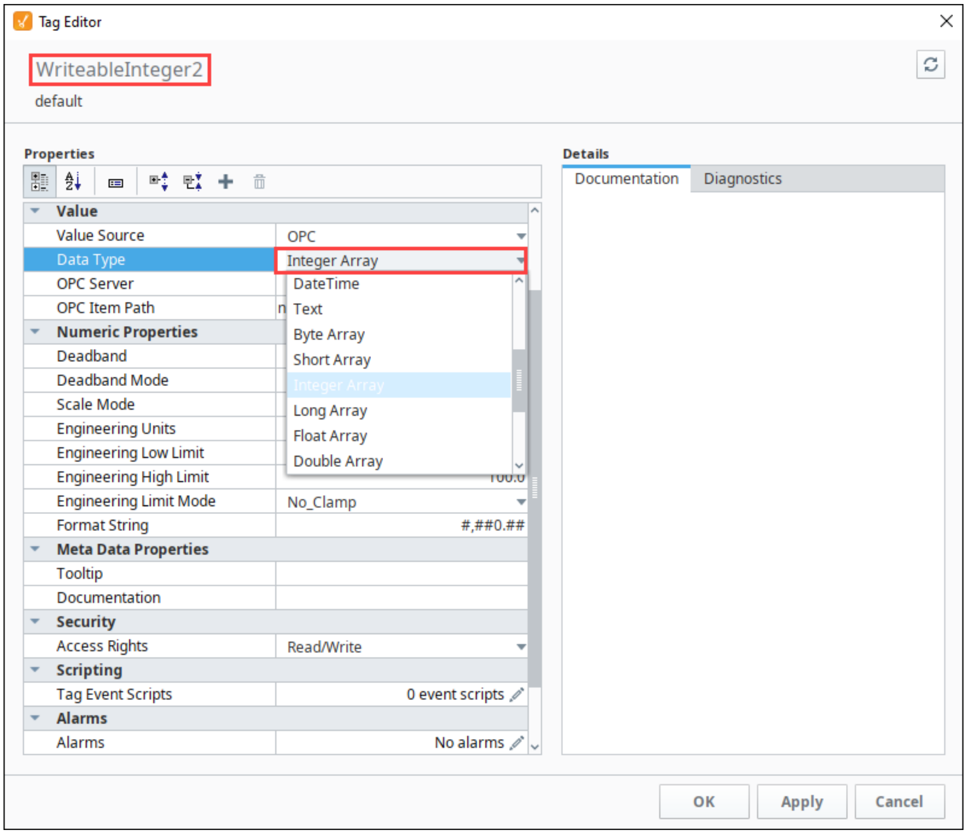
Properties

Value	
Value Source	OPC
Data Type	Integer
OPC Server	Ignition OPC UA Server
OPC Item Path	ns=1;s=[Sim_New_Programma
Numeric Properties	
Deadband	0.0001
Deadband Mode	Absolute
Scale Mode	Off
Engineering Units	
Engineering Low Limit	0.0
Engineering High Limit	100.0
Engineering Limit Mode	No_Clamp
Format String	#,##0.##
Meta Data Properties	
Tooltip	
Documentation	

2. In the **Tag Browser**, right click on the WriteableInteger1 Tag and choose **Copy**.
3. Right click on the Tag's folder, and choose **Paste**.



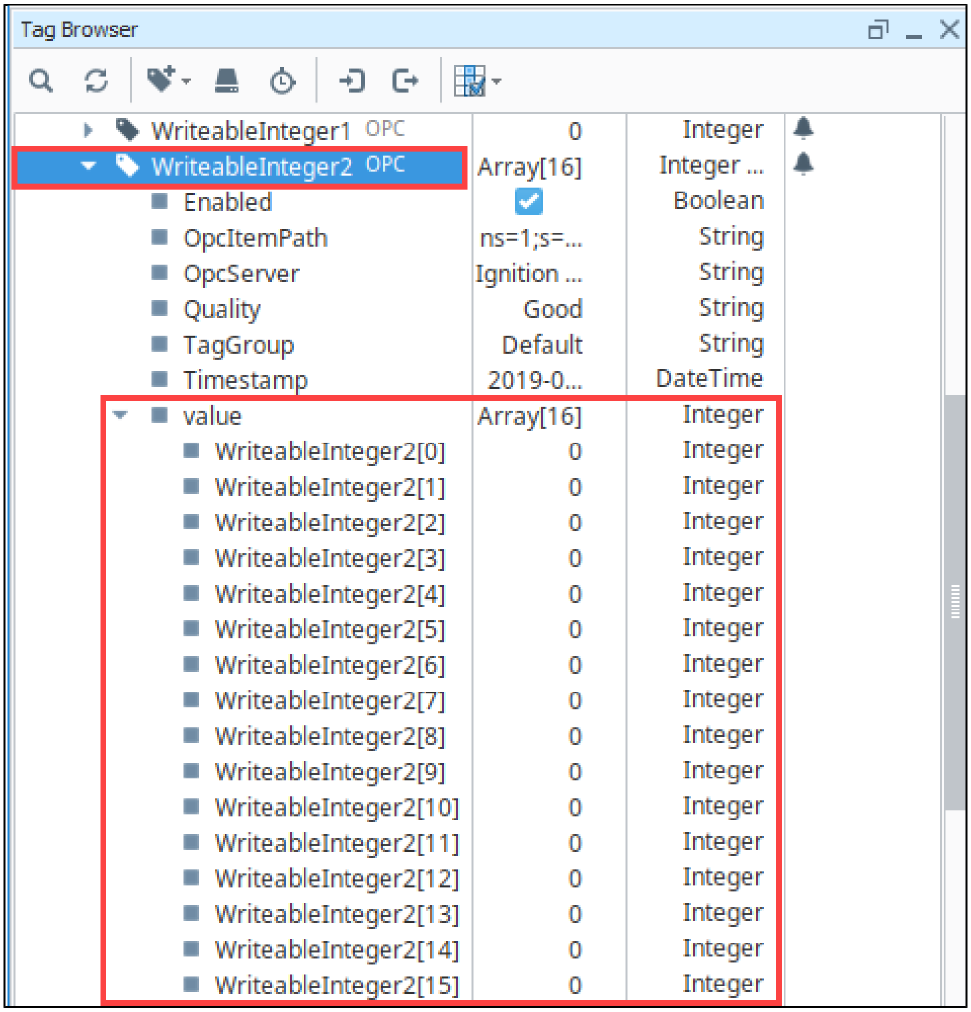
4. Ignition creates a copy of the first tag and names it WriteableInteger2. Double click on the new Tag to open the Tag Editor.
5. In the **Tag Editor**, change the Data Type to **Integer Array**. Then click **OK**.



WriteableInteger2
default

Value	
Value Source	OPC
Data Type	Integer Array
OPC Server	
OPC Item Path	n
Numeric Properties	
Deadband	
Deadband Mode	
Scale Mode	
Engineering Units	
Engineering Low Limit	
Engineering High Limit	100.0
Engineering Limit Mode	No_Clamp
Format String	#,##0.##
Meta Data Properties	
Tooltip	
Documentation	
Security	
Access Rights	Read/Write
Scripting	
Tag Event Scripts	0 event scripts
Alarms	
Alarms	No alarms

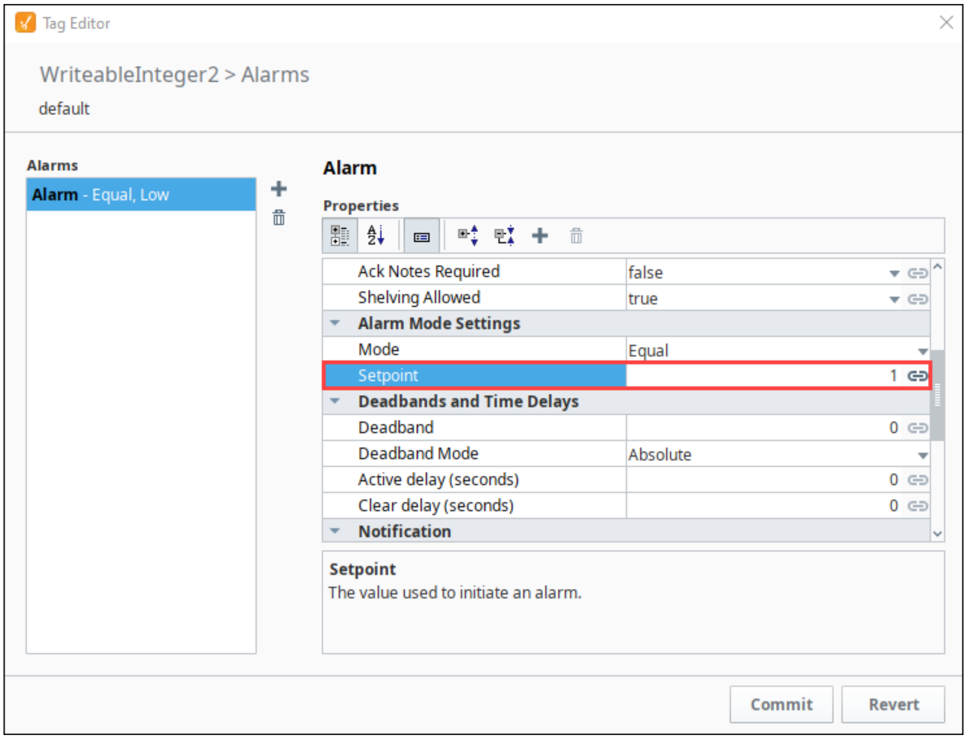
6. Once the Tag finishes re-subscribing, the value for WriteableInteger2 will now show "Array[16]" instead of the previous value. Expanding the Tag reveals a new Value section, and expanding this section will show the value of each bit.



WriteableInteger1 OPC	0	Integer	🔔
WriteableInteger2 OPC	Array[16]	Integer ...	🔔
Enabled	☑	Boolean	
OpcItemPath	ns=1;s=...	String	
OpcServer	Ignition ...	String	
Quality	Good	String	
TagGroup	Default	String	
Timestamp	2019-0...	DateTime	
value	Array[16]	Integer	
WriteableInteger2[0]	0	Integer	
WriteableInteger2[1]	0	Integer	
WriteableInteger2[2]	0	Integer	
WriteableInteger2[3]	0	Integer	
WriteableInteger2[4]	0	Integer	
WriteableInteger2[5]	0	Integer	
WriteableInteger2[6]	0	Integer	
WriteableInteger2[7]	0	Integer	
WriteableInteger2[8]	0	Integer	
WriteableInteger2[9]	0	Integer	
WriteableInteger2[10]	0	Integer	
WriteableInteger2[11]	0	Integer	
WriteableInteger2[12]	0	Integer	
WriteableInteger2[13]	0	Integer	
WriteableInteger2[14]	0	Integer	
WriteableInteger2[15]	0	Integer	

7. Double click on the WriteableInteger2 Tag again to edit it. Scroll down to the Alarming section.
8. Click on the **Edit** ✎ icon.
9. Create a new alarm with Mode set to **Equal**, but change the Setpoint to **"1,"** and then click **Commit**.
10. Click **OK**. Now an alarm will be active for each element that is set to 1.

11.



WriteableInteger2 > Alarms

default

Alarms

Alarm - Equal, Low

Alarm

Properties

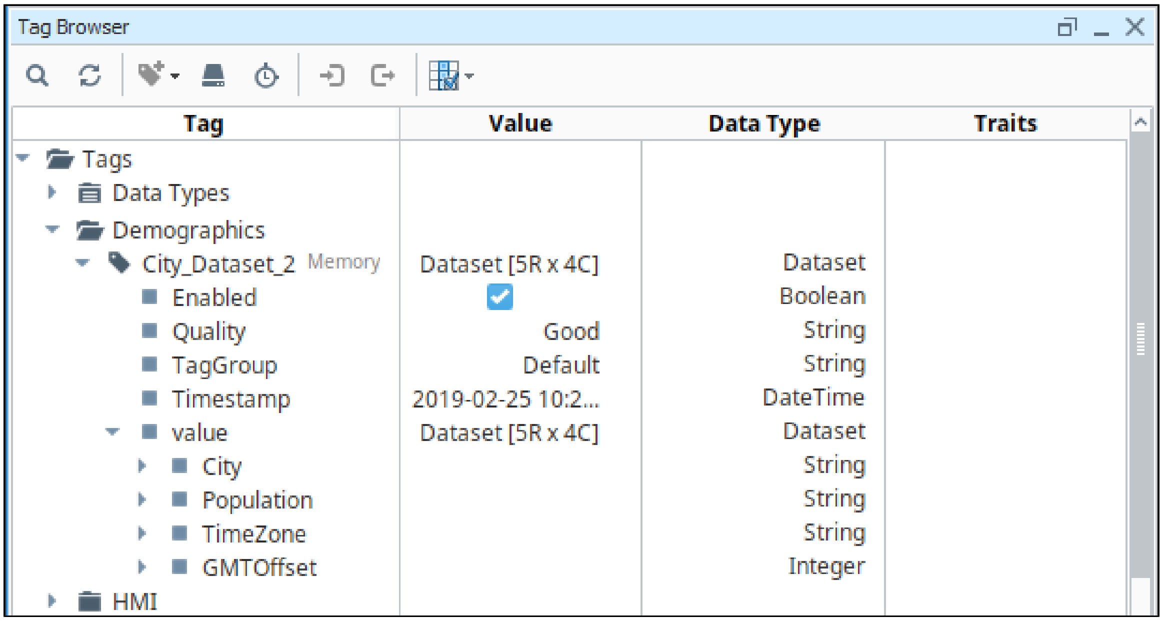
Ack Notes Required	false
Shelving Allowed	true
Alarm Mode Settings	
Mode	Equal
Setpoint	1
Deadbands and Time Delays	
Deadband	0
Deadband Mode	Absolute
Active delay (seconds)	0
Clear delay (seconds)	0
Notification	

Setpoint
The value used to initiate an alarm.

Dataset Tags

Dataset Tags allow multiple rows and columns worth of data to be stored in a Tag. Each column is exposed as a separate folder in the Tag (i. e., the "name" folder in the image below). Dataset Tags can be driven by a query, so it's possible to query for multiple columns on a row in a single Tag. This is more efficient than using multiple query Tags (and thus multiple queries) to retrieve the same data.

While dataset tags are convenient, the Tag History and Alarm systems do not support tags with dataset types.

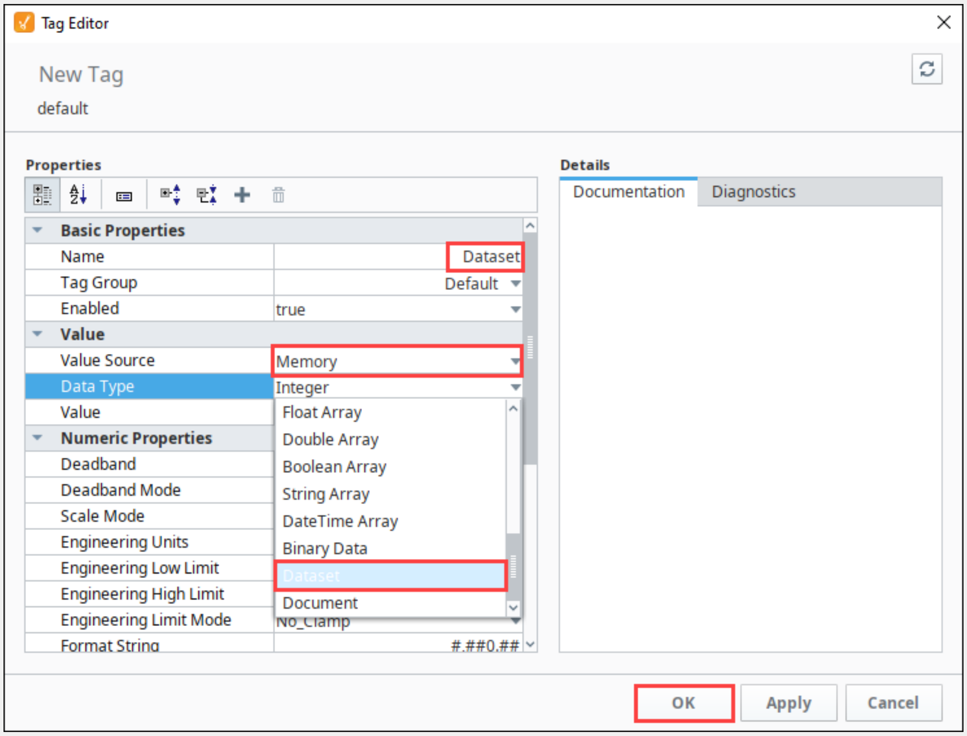


Tag	Value	Data Type	Traits
Tags			
Data Types			
Demographics			
City_Dataset_2 Memory	Dataset [5R x 4C]	Dataset	
Enabled	✓	Boolean	
Quality	Good	String	
TagGroup	Default	String	
Timestamp	2019-02-25 10:2...	DateTime	
value	Dataset [5R x 4C]	Dataset	
City		String	
Population		String	
TimeZone		String	
GMTOffset		Integer	
HMI			

Dataset Tag Example

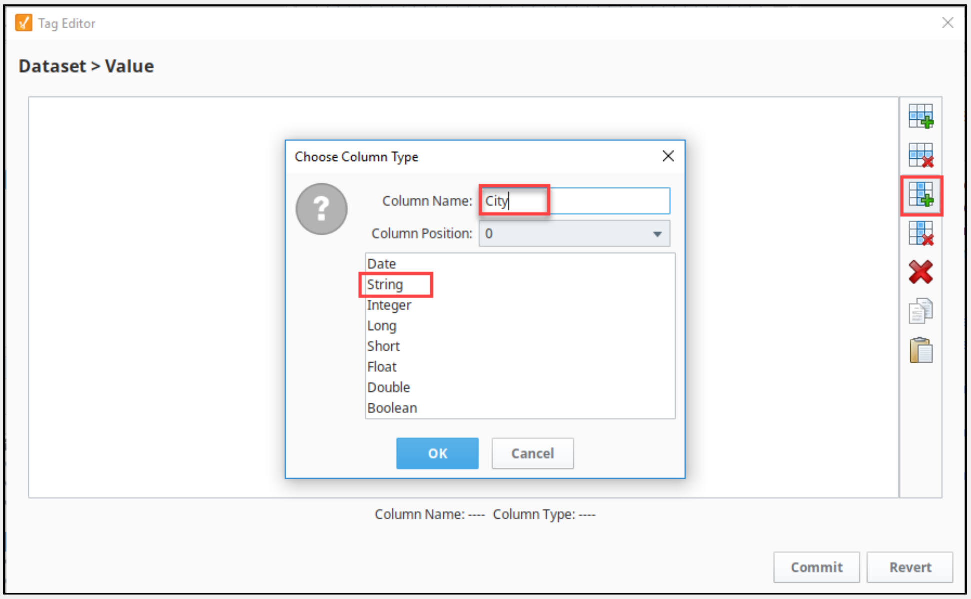
The following example will create a dataset memory Tag and display the contents in a Table component.

1. Create a new **Memory Tag**. Name it **Dataset**, and change the data type to **Dataset**. The Dataset will be empty by default.



Basic Properties	
Name	Dataset
Tag Group	Default
Enabled	true
Value	
Value Source	Memory
Data Type	Integer
Value	
Numeric Properties	
Deadband	
Deadband Mode	
Scale Mode	
Engineering Units	
Engineering Low Limit	
Engineering High Limit	
Engineering Limit Mode	No_Clamp
Format String	#,##0.##

2. Click the **Edit** icon next to **Value**. The Value screen is displayed. For this example, we created a simple dataset with four columns and five rows.
3. Click the **Add Column** icon. Name the first column **City** and set type to be **String**.



Column Name: City
Column Position: 0

Column Name: ---- Column Type: ----

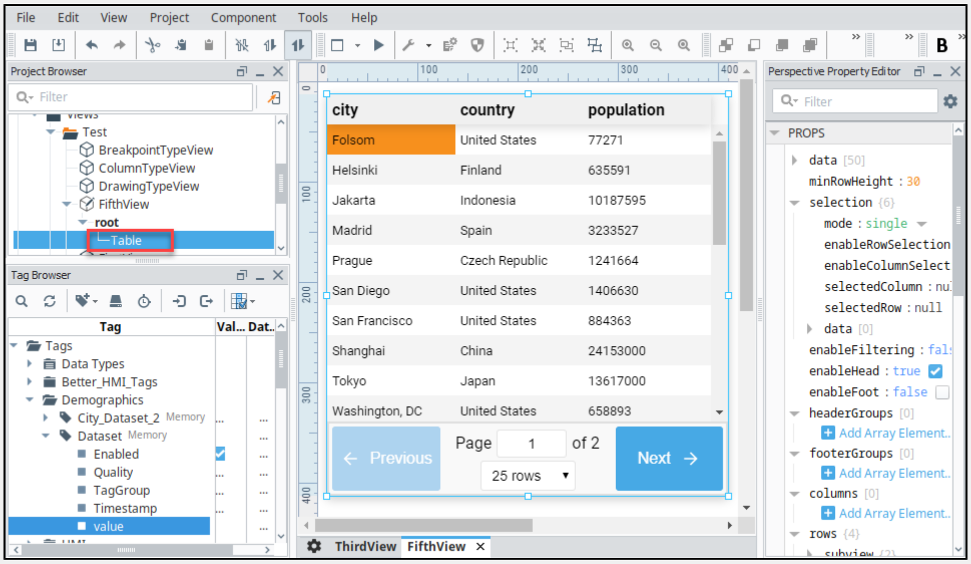
4. Repeat adding columns as follows:

Column Name: Population **Type**: Integer
Column Name TimeZone **Type**: String
Column Name GMTOffset **Type**: Integer

5. Click the **Add Row** icon. Add the row information as follows:

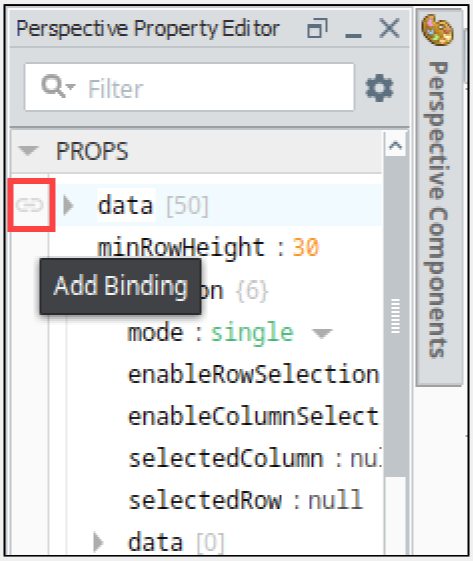
New York	8368710	EST	-5
Los Angeles	3833995	PST	-8
Chicago	2853114	CST	-6
Houston	2242193	CST	-6
Phoenix	1567924	MST	-7

6. Click the **Commit** button.
7. In the Designer, on a new View, add a **Table** component.

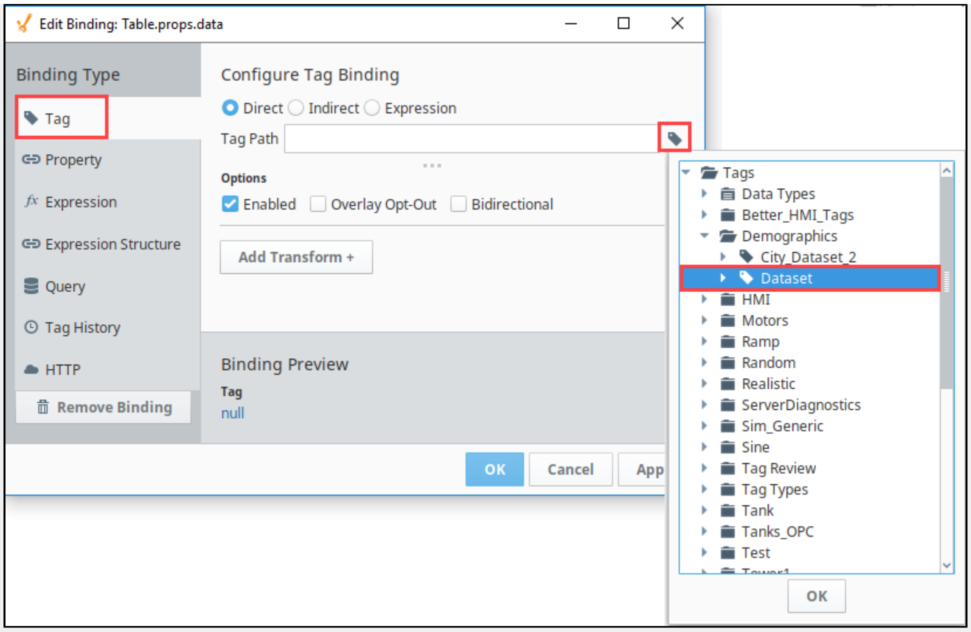


city	country	population
Folsom	United States	77271
Helsinki	Finland	635591
Jakarta	Indonesia	10187595
Madrid	Spain	3233527
Prague	Czech Republic	1241664
San Diego	United States	1406630
San Francisco	United States	884363
Shanghai	China	24153000
Tokyo	Japan	13617000
Washington, DC	United States	658893

8. Select the **Table** component. In the Property Editor, click on the **Binding** icon next to the **data** property.



9. On the **Edit Binding** screen, select **Tag** as the **Binding Type**.
10. Click the **Browse Tag** icon next to the **Tag Path**.
11. Navigate to the Dataset tag. Highlight the Tag in the list and then Click **OK**.



● Direct ○ Indirect ○ Expression

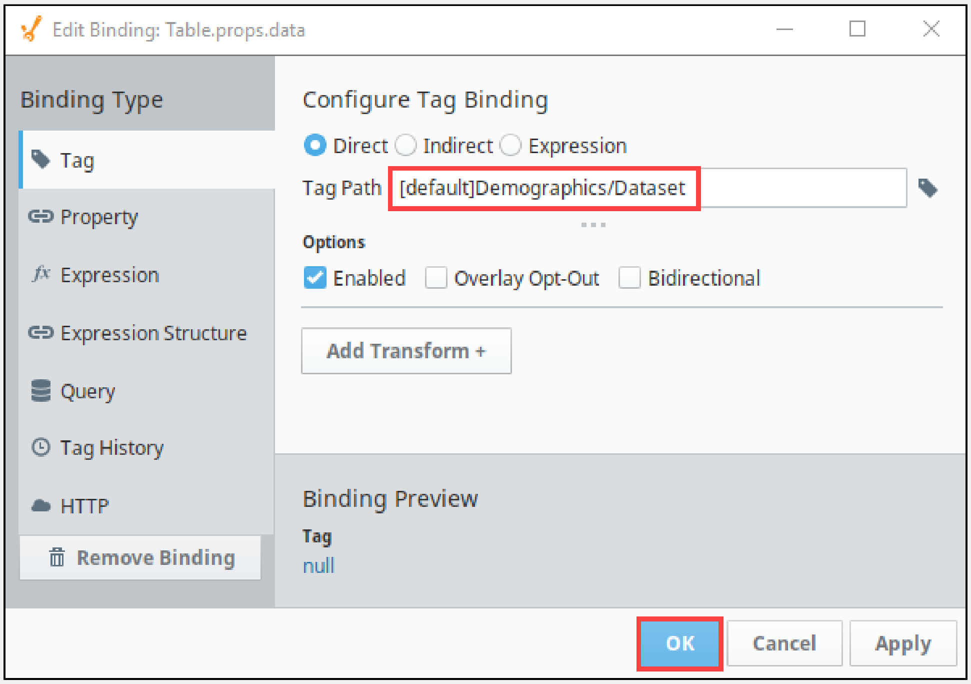
Tag Path

☑ Enabled □ Overlay Opt-Out □ Bidirectional

Tag
null

12. The Tag path is now displayed. Click OK.

> ⚠ These bindings, just like any other Tag binding, can be set to bi-directional. In this case, you would have to be able to somehow write to the data property using scripting for the bi-directional binding to work.



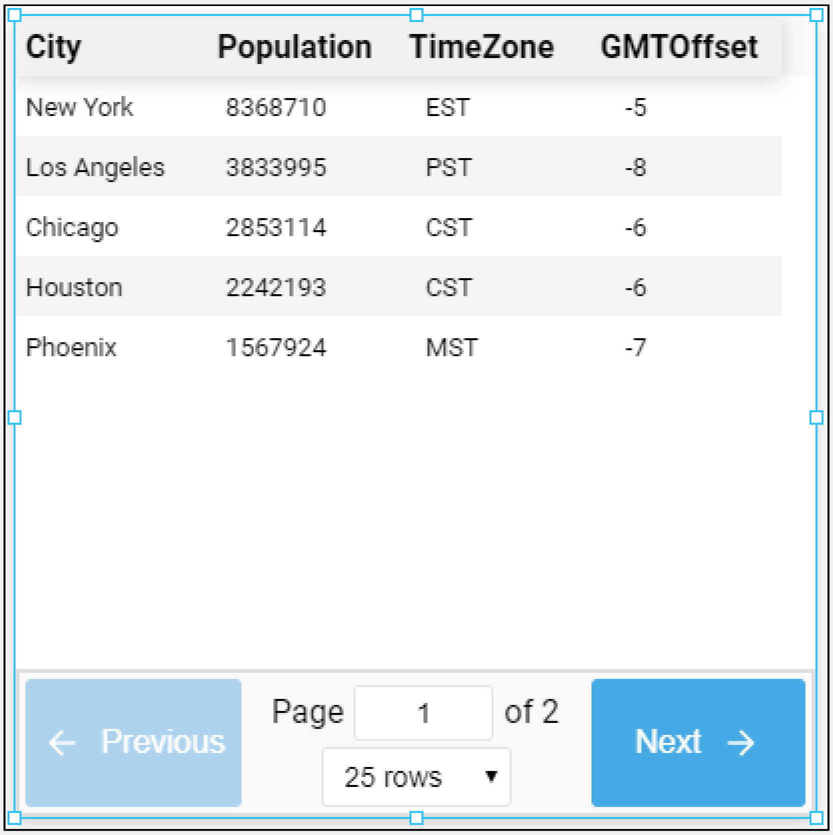
● Direct ○ Indirect ○ Expression

Tag Path [default]Demographics/Dataset

☑ Enabled □ Overlay Opt-Out □ Bidirectional

Tag
null

13. The Table component now displays the contents of the dataset.



City	Population	TimeZone	GMTOffset
New York	8368710	EST	-5
Los Angeles	3833995	PST	-8
Chicago	2853114	CST	-6
Houston	2242193	CST	-6
Phoenix	1567924	MST	-7

Related Topics ...

- Creating Tags

Tag Paths

Tags and their properties can be referenced by a string-based path. Each Tag has a unique absolute path and often has many equivalent relative paths when referenced from other Tags. You most often generate these paths by browsing or through dragging. However, it's a good idea to understand how Tag paths work, particularly if you get into indirect Tag binding or Expressions, or scripting.

A Tag path looks something like this: `[Tag Provider]folder/path/tag.property`

The `folder/path/tag.property` portion of the path may contain the following:

- A Tag
- Any number of nested folders followed by a Tag, separated by forward slashes (/)
- A period (.) followed by a property name after the Tag. Omitting this is equivalent to using the. Value property.

The `[Tag Provider]` portion surrounded by square braces can have the following options:

Source Option	Meaning
[Tag Provider Name]	The name of the Tag provider that hosts the Tag.
[] or not specified	The default Tag provider for the current project. If used in the gateway scope, then using this notation can (generally) result in an invalid path, as the gateway doesn't have a default tag provider.
[.]	Relative to the folder of the Tag that is being bound. This is especially useful in UDT definitions.
[~]	Relative to the Tag provider of the Tag that is being bound (root node).
[Client]	Refers to the Vision Client Tag provider. Only Vision Client tags.
[System]	Refers to a System Tag.

What about Perspective Client Tags?

There are no "Client Tags" in a Perspective Session. Take a look at the Session Properties for similarly scoped variables.

On this page ...

Using Relative Tag Paths

Tag paths that begin with [.] or [~] are known as *relative paths*. They are used inside Tags that bind to other Tags, and they are relative to the host Tag's path. Using the relative path syntax helps to avoid problems caused by moving Tags and renaming providers.

> [.] refers to the Tag's current folder. By using [.], Tags can be moved from folder to folder without problem (provided that all of the applicable Tags are moved together). Additionally, you can use "**..**" (two periods) to go back one folder from the current relative position, for example [.]../../tag allows you to reference a Tag two folders up.
>
> [~] refers to the Tag's provider root. It can replace the explicit provider name, and thus protect against provider renames and importing /exporting/moving Tags between different providers.

Tag Path Manipulation

Ignition provides a great deal of flexibility for Tag addressing since Tag paths and Tag properties are string-based. The underlying strings that compose a valid Tag path can be assembled from many different parts in with the eventual construction resulting in a valid Tag path.

The following scripting demonstrates this concept. Suppose there was a Tag path to a level indicator in a tank. In this case it is the default Tag provider, Tanks folder, Tank 1 Folder, and the Level Tag.

```
tagPath = "[default]Tanks/Tank 1/Level"
```

But suppose that there was more than just Tank 1 and instead there was Tank 2, Tank 3, Tank 4, etc. Dynamically changing the Tag paths is simple because Ignition's Tag paths are string representations. The following takes the tank number and inserts it into a new Tag path. The tankNumber variable changes the eventual creation of the tagPath. Using this method in scripting or in an expression binding will look slightly different.

Python Dynamic Tag Path

```
tankNumber = 2
tagPath = "[default]Tanks/Tank %i/Level" % tankNumber
```

Expression Dynamic Tag Path

```
tag("[default]Tanks/Tank "+{Root Container.tankNumber}+"/Level").value
```

The result of the tagPath variable will be **[default]Tanks/Tank 2/Level** which is a valid Tag path to the the level sensor for Tank 2.

Using the {this} Keyword

Tags have a built-in "this" keyword, that can be used as a reference to the tag. The keyword is useful in cases where an expression is being configured on a tag property, and you want to reference the value of another property.

```
// The expression below always returns the name of the tag
{this.name}
```

"this" in Alarms

The "this" keyword is also available for use on expressions on alarms. However when used like this, the keyword still refers to the Tag, not the alarm. Thus, "this.name" on an alarm property expression would return the name of the Tag, not the name of the alarm.

For more information on Ignition's Expression language, see Expression Overview and Syntax.

Related Topics ...

- Tag Event Scripts

Tag Event Scripts

Scripts can be attached to Tags. When you edit a Tag, you can navigate to the Tag Events section and click on the **Edit** ✏ icon to see a list of all of the available events. Those events are

- Value Changed
- Quality Changed
- Alarm Active
- Alarm Cleared
- Alarm Acknowledged

Each event is unique and can have specialized arguments depending on the event. Also, because Tags are stored in the Gateway, all these scripts fire in the Gateway scope.

⚠ Because these scripts are Gateway scoped, certain things like print statements will not print to the Designer console, but will print instead to the wrapper log file in Ignition's installation directory. Keep that in mind when doing any testing.

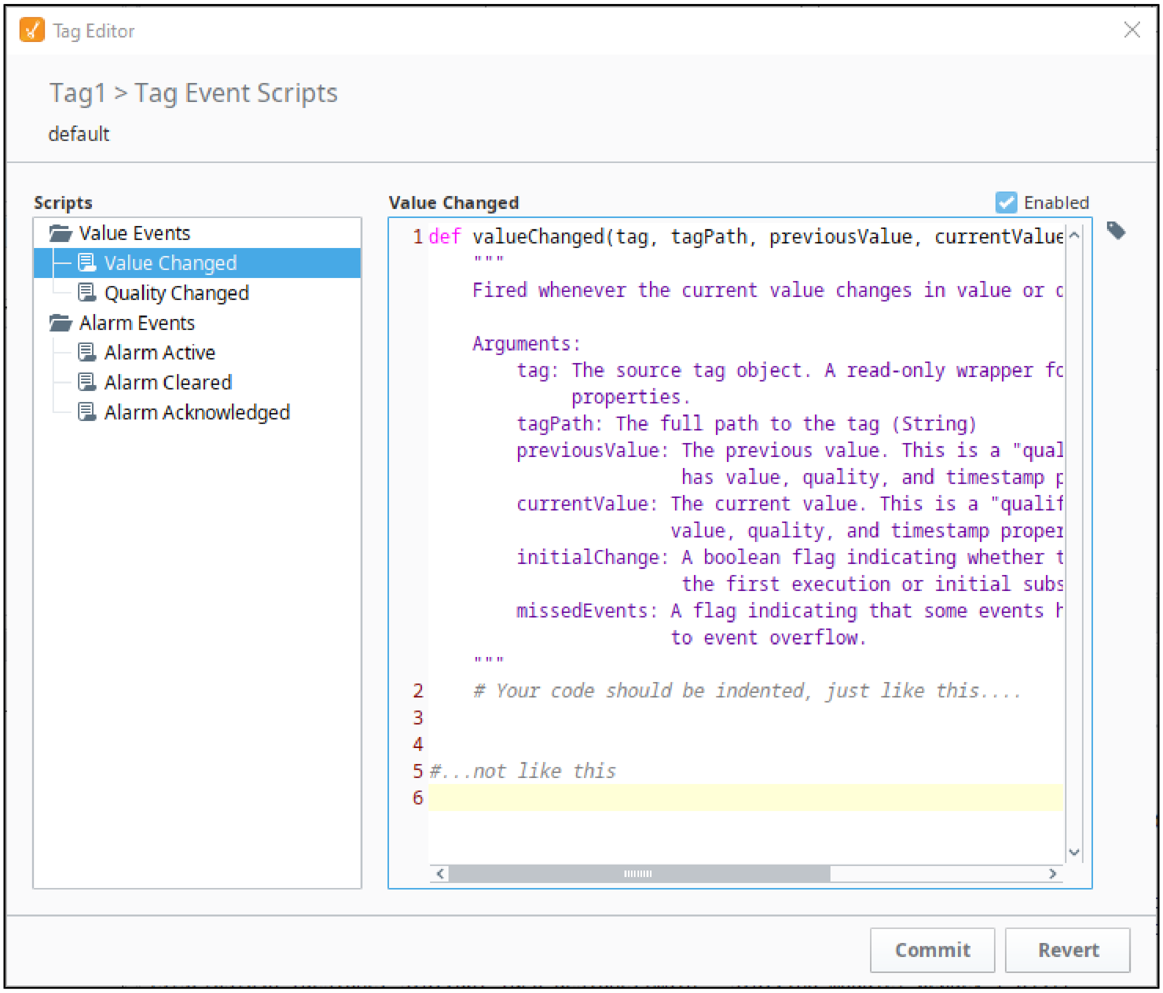
On this page ...

Tag Event Scripts

Watch the Video

Event Options

Once an event has been selected, note that the top of the text area is defining a function. As a result, all code that should execute when this event occurs should be indented at least once to be included in the definition.



```
def valueChanged(tag, tagPath, previousValue, currentValue
    """
    Fired whenever the current value changes in value or c

    Arguments:
        tag: The source tag object. A read-only wrapper fo
             properties.
        tagPath: The full path to the tag (String)
        previousValue: The previous value. This is a "qual
                       has value, quality, and timestamp p
        currentValue: The current value. This is a "qualif
                      value, quality, and timestamp proper
        initialChange: A boolean flag indicating whether t
                       the first execution or initial subs
        missedEvents: A flag indicating that some events h
                      to event overflow.
    """
    # Your code should be indented, just like this....



#...not like this

```

Value Changed

The Value Changed event is fired whenever the value of this particular Tag changes. This event has a variety of arguments available for use in the script:

- **String tagPath** - The full path to the Tag. Example: `[tagProvider]Folder/Folder/Tag`
- **Object previousValue** - The previous value. This is a qualified value, so it has value, quality, and timestamp properties.
- **Object currentValue** - The current value. This is a qualified value, so it has value, quality, and timestamp properties.
- **Boolean initialChange** - A boolean flag indicating whether this event is due to the initial subscription or the first execution after a Tag update.
- **Boolean missedEvents** - A flag indicating that some events have been skipped due to an event overflow.

The currentValue and previousValue arguments are qualified values: objects that contain a value, timestamp, and quality. This means that to get to the value of the currentValue, your script would need to access `currentValue.value`

Quality Changed

The Quality Changed event is fired whenever the quality of this particular Tag changes. Since Tags use a 'qualified value', which include a value, quality, and timestamp, this script will fire whenever any of those change. This event has a variety of arguments available for use in the script:

- **String tagPath** - The full path to the Tag. Example: `[tagProvider]Folder/Folder/Tag`
- **Object previousValue** - The previous value. This is a qualified value, so it has value, quality, and timestamp properties.
- **Object currentValue** - The current value. This is a qualified value, so it has value, quality, and timestamp properties.

- **Boolean initialChange** - A boolean flag indicating whether this event is due to the initial subscription or the first execution after a Tag update.
- **Boolean missedEvents** - A flag indicating that some events have been skipped due to an event overflow.

Alarm Active

The Alarm Active event fires whenever a new active alarm event occurs. This event has a variety of arguments available for use in the script:

- **String tagPath** - The full path to the Tag. Example: `[tagProvider]Folder/Folder/Tag`
- **String alarmName** - The name of the alarm. This does not include the full alarm path.
- **Object alarmEvent** - The full alarm event object. The properties available to this object are:
  - eventId
  - source
  - name
  - priority
  - displayPath
  - displayPathOrSource (the display path if it is set, otherwise the source)
  - state (the current state, active/cleared + acked/unacked)
  - eventState (the last transition, active, clear, acknowledged)
  - isClear (a boolean if the alarm is currently cleared)
  - isAcked (a boolean if the alarm is currently acknowledged)
  - isShelved (a boolean if the alarm is currently shelved)
  - notes
- **String alarmPath** - The full alarm path.
- **Boolean missedEvents** - A flag indicating that some events have been skipped due to an event overflow.

Alarm Cleared

The Alarm Cleared event fires whenever an alarm becomes cleared on the Tag.This event has a variety of arguments available for use in the script:

- **String tagPath** - The full path to the Tag. Example: `[tagProvider]Folder/Folder/Tag`
- **String alarmName** - The name of the alarm. This does not include the full alarm path.
- **Object alarmEvent** - The full alarm event object. See the Alarm Active alarmEvent object for the full list of available properties.
- **String alarmPath** - The full alarm path.
- **Boolean missedEvents** - A flag indicating that some events have been skipped due to an event overflow.

Alarm Acknowledged

The Alarm Acknowledged event fires whenever an alarm event becomes acknowledged on the Tag.This event has a variety of arguments available for use in the script:

- **String tagPath** - The full path to the Tag. Example: `[tagProvider]Folder/Folder/Tag`
- **String alarmName** - The name of the alarm. This does not include the full alarm path.
- **Object alarmEvent** - The full alarm event object. See the Alarm Active alarmEvent object for the full list of available properties.
- **String alarmPath** - The full alarm path.
- **String ackedBy** - The full path to the user that acknowledged the alarm.
- **Boolean missedEvents** - A flag indicating that some events have been skipped due to an event overflow.

Using the Project Library in a Tag Event Script

Scripts defined in a project script can be called from a Tag Event Script. However, only scripts defined in the Gateway Script Project may be used. For more information on configuring the Gateway Scripting Project, please see the Project Library page.

UDT Parameters in Tag Event Scripts

Parameters on UDTs can be interacted with from Tag Event Scripts. There are two main approaches.

Modern Approach

The following feature is new in Ignition version **8.0.7**
Click here to check out the other new features

As of 8.0.7, parameters values can be accessed as dictionary values. The benefit of this approach is that value changes to UDT parameters will be reflected in subsequent calls. In most cases, this modern approach is preferable. Thus, if the script needs to access the most recent parameter values on a UDT, and the parameters can change through a means other than editing the UDT (which would restart the tag), then this approach should be used.

If trying to access the value of a parameter named "myParam" from a Tag Event Script within the UDT, the script would look like:

```
# Reminder: "tag" is a built-in argument on all Tag Event Scripts. Accessing the "parameters" key on the tag argument will
# provide access to all UDT parameters.
paramValue = tag['parameters']['myParam']
```

Legacy Approach

In 7.9 and prior, UDT parameters could be access in Tag Event Scripts with the familiar curly brackets approach. Thus, if trying to access the value of a parameter named "myParam" from a Tag Event Script within the UDT, the script would look like:

```
paramValue = {myParam}
```

A large caveat with this approach is that value changes made to the parameter ("myParam", in the example above) would not be reflected in scripts until the UDT was restarted. UDT parameters are pre-compiled, which in this case means value changes are mostly ignored until the UDT is restarted.

In all cases, the modern approach above is preferred.

Tag Script Examples

Printing all parameters

```
# This script will fetch all of the possible parameters in the tag changed script.
# Note that this will not print out to the console, but it will print to the wrapper logs located on the gateway.

path = tagPath
prev = previousValue
cur = currentValue
init = initialChange
missed = missedEvents
print path, prev.value, cur.quality, init, missed
```

Automatic Reset

```
# This code can be used on a Value Changed script, and will automatically reset the value of the tag to 0 after it goes above 3000.
# This can be useful for counter tags.
if currentValue.value > 3000:
	system.tag.writeBlocking(["[default]IntegerCounterTag"], 0)
```

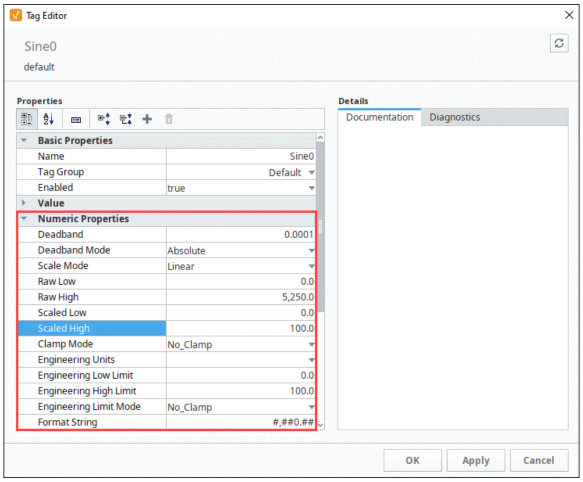
Copy to another Tag

```
# This code can be used on a Value Changed script, and will record the highest value of the current tag onto another memory tag.
# This can be useful for recording the highest temperature.
highestRecordedTemp = system.tag.readBlocking(["[default]HighestTempTag"])[0].value
if currentValue.value > highestRecordedTemp:
	system.tag.writeBlocking(["[default]HighestTempTag"], [currentValue.value])
```

Tag Scaling Properties

Configuring a Tag's scaling will condition the data for use within the Ignition Designer. Scaling will take the raw PLC value driving a Tag, do some math, and use the resulting value as the value of that Tag. Scaling works both ways. When you write to that Tag, Ignition will scale it in the opposite direction before writing to the PLC.

For example, if the capacity of a tank is 5250 gallons, but the tank's fill level is better represented in the Designer as percentage of 0 through 100, configuring the Tag's scaling property, will result in the Tag displaying 0 through 100, while the actual tank fill moving is between 0 and 5250 gallons. For this example, you can double-click on the Tag to open the **Tag Editor**, and expand **Numeric Properties** to configure the scaling of the Tag. When scaling between a Raw Low and and High, and Scaled Low and High, select the **Linear** Scale Mode. So what Ignition is actually doing, is setting up the calculations behind the scenes to scale the value appropriately.



Basic Properties	
Name	Sine0
Tag Group	Default
Enabled	true
Value	
Numeric Properties	
Deadband	0.0001
Deadband Mode	Absolute
Scale Mode	Linear
Raw Low	0.0
Raw High	5,250.0
Scaled Low	0.0
Scaled High	100.0
Clamp Mode	No_Clamp
Engineering Units	
Engineering Low Limit	0.0
Engineering High Limit	100.0
Engineering Limit Mode	No_Clamp
Format String	#,##0.##

On this page ...

INDUCTIVE UNIVERSITY

Tag Scaling

Watch the Video

If you are using scaling...

The numbers and units don't have to match-up. Scaling is straight from one number to another number. It doesn't matter what the units are, and it doesn't matter what the conversion is. What is important, is that the data type of the Tag must match the data type of the scale value (i.e., dividing an integer in the PLC by 10 probably means your Ignition Tag should be a float).

Scaling is not available on Memory Tags: Memory Tags are not driven by an external source such as a PLC or SQL query, so scaling will never be applied. In these scenarios, it is recommended to scale the mechanism that is writing to the Memory Tag instead. Numerical properties of Tags can be scaled allowing automatic bi-directional conversion outside of the PLC. Scaling types include **Linear** scaling, **Square Root** scaling, and **Exponential Filter** scaling. The numerical properties are available to **OPC**, **Expression**, **Database**, and **Client** Tags whose data types are numeric. For a complete list of all of the Tag Scaling properties, see Tag Properties.

Linear Scaling

The value will be scaled linearly between the low and high values, and clamped as appropriate.
The linear equation is:

```
ScaledValue = S * (Value-RL)/R + SL
```

Where:

S = (ScaledHigh-ScaledLow)
R = (RawHigh - RawLow)
RL = RawLow
SL = ScaledLow

Square Root Scaling

The equation for square root scaling is:

```
ScaledValue = S *((Value-RL)/R) + SL
```

Where:

S = (ScaledHigh-ScaledLow)
R = (RawHigh - RawLow)
RL = RawLow
SL = ScaledLow

Exponential Filter Scaling

This mode implements a simple linear recursive filter to smooth values. The scale factor corresponds to the weight of the smoothing effect, and is a value between 0.0 and 1.0. The smaller the factor, the greater the degree of smoothing.

The equation for the filter is:

```
y(t) = (1-f)*y(t-1)+f*x(t)
```

Where:

y(t) = the output at time t
y(t-1) = the previous output
x(t) = the input value (current value)
f = the scale factor, with 0.0<=f<=1.0

Note: Only **good** quality values are considered for the filter. **Bad** quality values are ignored.

Bit Inversion Scaling

This simple scaling mode will generate the complement of a binary value. If the current value is coming in as 0001_0101 (21), this will return a 1110_1010 (-22) instead. A popular use for this scale mode is that it can be used to invert modbus values if your device stores them in reverse bit order. Note that Bit Inversion Scaling uses a little-endian format.

Scaling Examples

Simple Divide by Ten

This is common when storing a single decimal point of precision as an Integer in the PLC. This is to save space by not using a Float type.

Raw Low: 0.0
Raw High: 100.0
Scaled Low: 0.0
Scaled High: 10.0

4-20 Milliamp Signal to Percent

This is common when using a simple pressure sensor. The sensor is calibrated to send 4 milliamps (minimum value) when the tank is empty, and 20 milliamps (maximum value) when the tank is full.

Raw Low: 4.0
Raw High: 20.0

Scaled Low: 0.0
Scaled High: 100.0

4-20 Milliamp Signal to Gallons

This is common when using a simple pressure sensor. The sensor is calibrated to send 4 milliamps (minimum value) when the tank is empty, and 20 milliamps (maximum value) when the tank is full (5000 gallons).
Note: There is no direct conversion between amps and gallons. In scaling it doesn't matter.

Raw Low: 4.0

Raw High: 20.0
Scaled Low: 0.0
Scaled High: 5000.0

Related Topics ...

- Exporting and Importing Tags

Tag Quality and Overlays

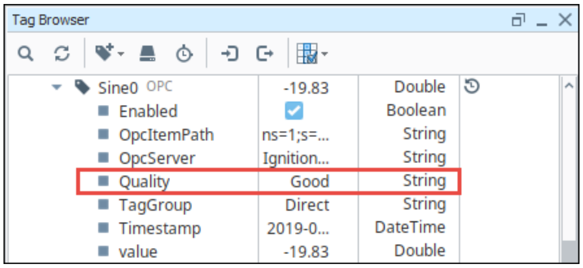
Tag Quality

Ignition has Quality built into Tags automatically. Data Quality is the measure of how reliable a particular Tag's data is. If a Tag's quality is not *Good (192)*, the value generally should *not* be trusted, and an overlay is shown to make the Tag's bad quality visible. There are a wide variety of causes of bad data, from network disconnections, to software failure, to invalid Tag configurations.

Quality is extremely important and exists on every Tag, and is visible in the Designer, Perspective Sessions and Vision Clients through an Overlay system. There are a few of ways to check Tag Quality.

Tag Quality in the Designer

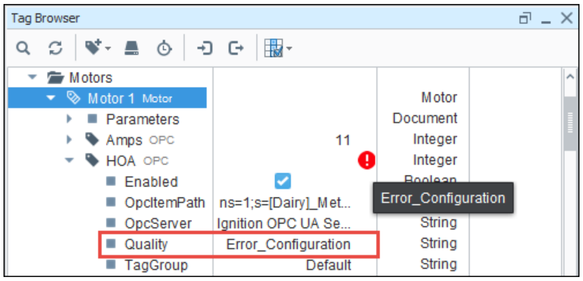
In the **Tag Browser**, find your Tag, expand it, and scroll down to the meta property called **Quality**. Here, you can check the quality of the Tag. This example shows a Good Quality Tag, meaning the Tag can be trusted.



Tag Browser

Sine0 OPC	-19.83	Double
Enabled	☑	Boolean
OpcItemPath	ns=1;s=...	String
OpcServer	Ignition...	String
Quality	Good	String
TagGroup	Direct	String
Timestamp	2019-0...	DateTime
value	-19.83	Double

One obvious indicator if the Tag is of bad quality is if there is a a red exclamation mark ❗ icon next to the Tag. Hover over the icon to see the error message and also expand the Tag to see the quality issue. This example shows the HOA Tag with with a Configuration Error. Ignition identifies the Quality issue to help you resolve the issue promptly.



Tag Browser

Motors		
Motor 1 Motor		Motor
Parameters		Document
Amps OPC	11	Integer
HOA OPC	❗	Integer
Enabled	☑	Boolean
OpcItemPath	ns=1;s=[Dairy]_Met...	
OpcServer	Ignition OPC UA Se...	String
Quality	Error_Configuration	String
TagGroup	Default	String

Error_Configuration

Component Overlays

It is especially important in HMI screens to be able to gauge the health and accuracy of what is displayed at a glance. In a highly distributed system like Ignition, it is especially important as the client may be located at quite a distance (maybe across the world) from the physical process it is monitoring and controlling.

For these reasons, Perspective and Vision components display visual overlays for various reasons to indicate that the data they are displaying is not good, or pending a reply from the device. Each data binding that drives a component is evaluated for quality. If any of these qualities becomes

On this page

...

INDUCTIVE UNIVERSITY

Tag Quality and Overlays

Watch the Video

INDUCTIVE UNIVERSITY

Component Overlays

Watch the Video

poor, the Perspective or Vision component will show an overlay. The different overlays can mean different things, denoting their underlying cause. What they indicate is based on the Quality properties of Tags.

Component overlays appear in the Designer workspace, Perspective Session, and Vision Client to let designers and operators know when there is a problem with one of the bindings on a component. What is cool about component overlays is that they not only tell you that there is a problem, but they also help diagnose the problem. Vision and Perspective overlay systems are similar, but each look a little bit different.

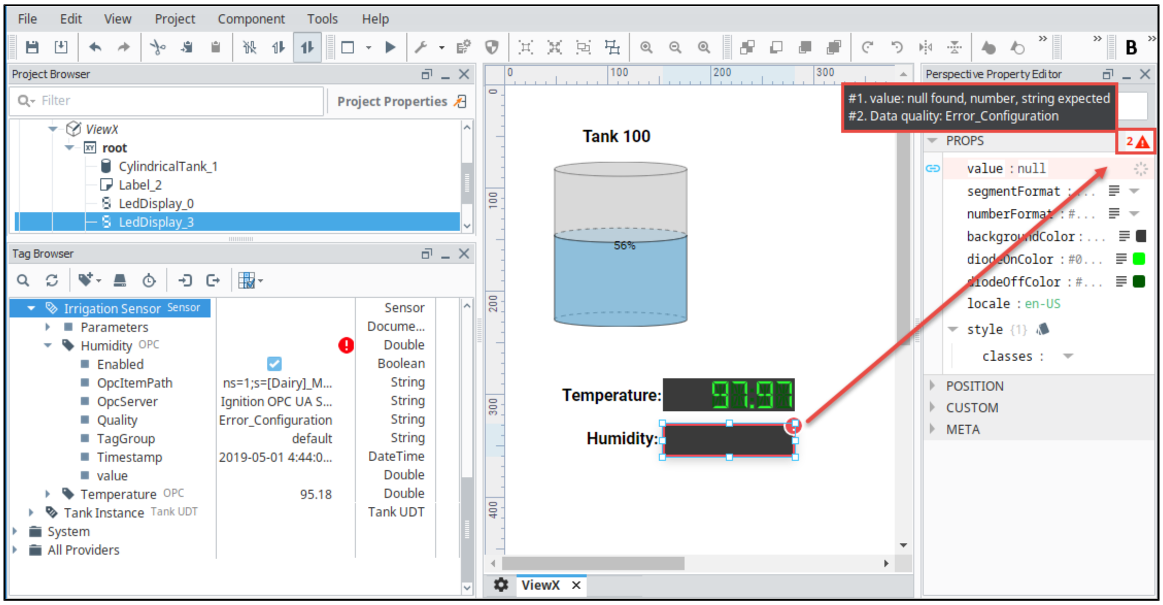
The sections below describe in detail Perspective and Vision overlays. Each module has its own Tag Quality Code Reference Table displaying the error codes and what they mean.

Perspective Component Overlays

Perspective overlays give designers and operators a lot of information to help them diagnose and correct a problem. Component overlays look and behave slightly different in Designer Mode, Preview Mode, and in a Perspective Session. Each is described below.

Component Overlays in Designer Mode

In the following example, you see the LED Display component has an overlay with a red border around it and an exclamation mark ❗ icon. If you select the component, a red triangle will appear in upper right corner of the Property Editor giving the designer some information about the error. Click the red triangle to open the message box to identify the data quality error and cause . You'll also notice that the affected property (i.e., value) is highlighted in light red indicating this property is the one having a problem. If the error is binding related, click on the binding icon next to the property to open the Edit Binding window.

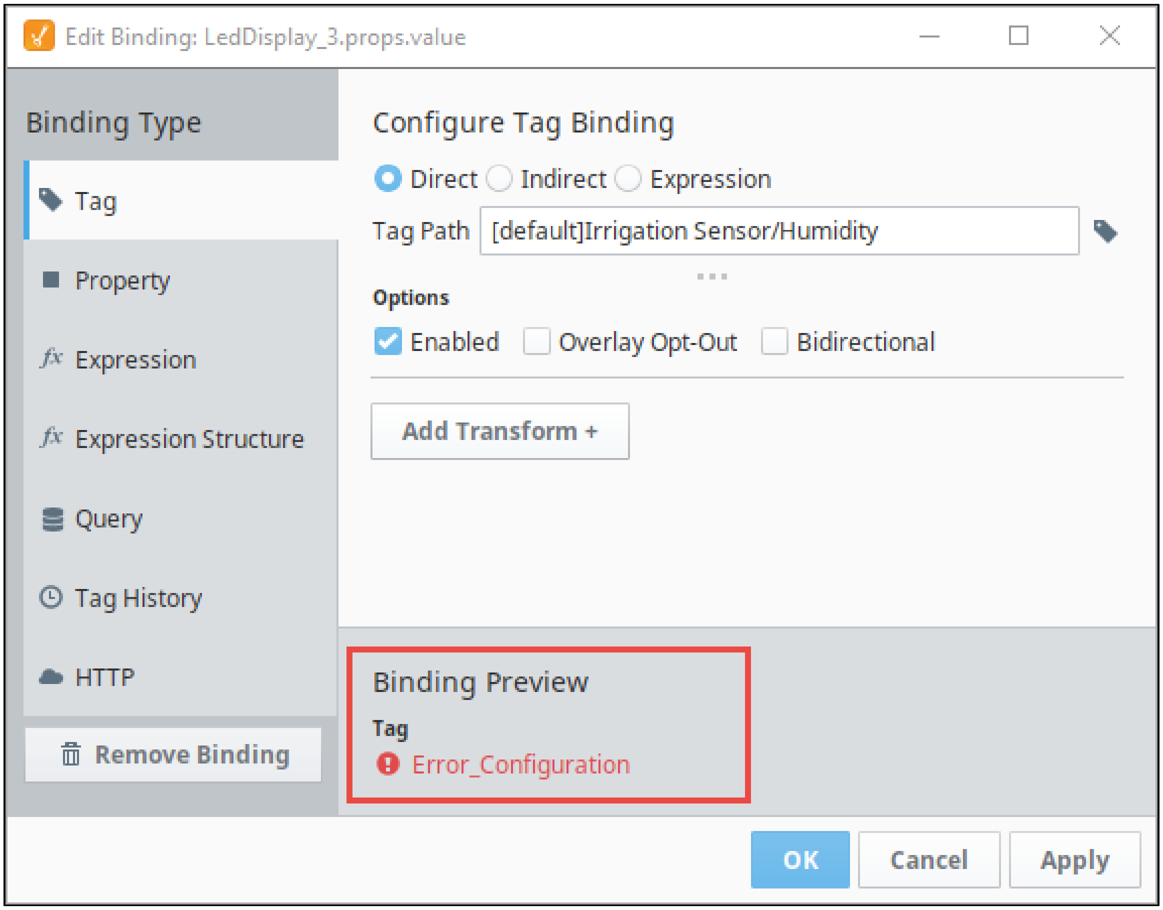


Irrigation Sensor		Sensor
Parameters		Docume...
Humidity		Double
Enabled	☑	Boolean
OpcItemPath	ns=1;s=[Dairy]_M...	String
OpcServer	Ignition OPC UA S...	String
Quality	Error_Configuration	String
TagGroup	default	String
Timestamp	2019-05-01 4:44:0...	DateTime
value		Double
Temperature	95.18	Double
Tank Instance		Tank UDT

#1. value: null found, number, string expected
#2. Data quality: Error_Configuration

value : null
segmentFormat : ...
numberFormat : #...
backgroundColor : ...
diodeOnColor : #0...
diodeOffColor : #...
locale : en-US
style {1}
classes :

Under the Binding Preview area, you can see there is a Configuration Error. If the problem is with a binding type, chances are you might be able to correct it here, or you may need to go directly to the Tag, or check other resources like a database, OPC server, etc.



● Direct ○ Indirect ○ Expression

Tag Path [default]Irrigation Sensor/Humidity

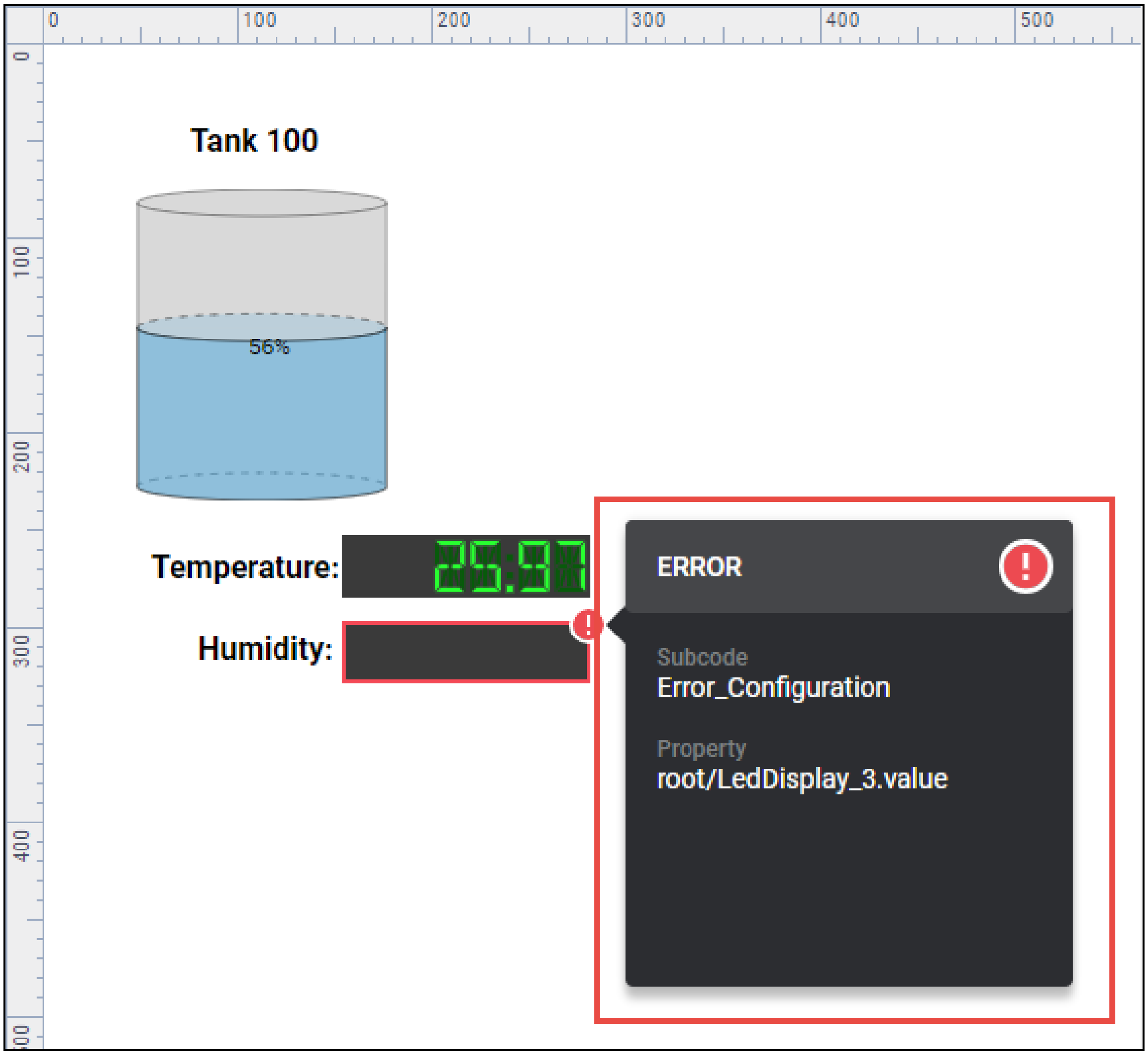
☑ Enabled □ Overlay Opt-Out □ Bidirectional

Tag

Error_Configuration

Component Overlays in Preview Mode

Switch from the Designer mode to the Preview Mode. To put your active view in **Preview Mode,** press the Preview / Designer mode ▶ icon in the top menubar. Components that have a problem will have a red overlay and exclamation mark around them. To see the error, click on the exclamation mark ❗. This opens a message box that informs you that you have an error and identifies affected property.



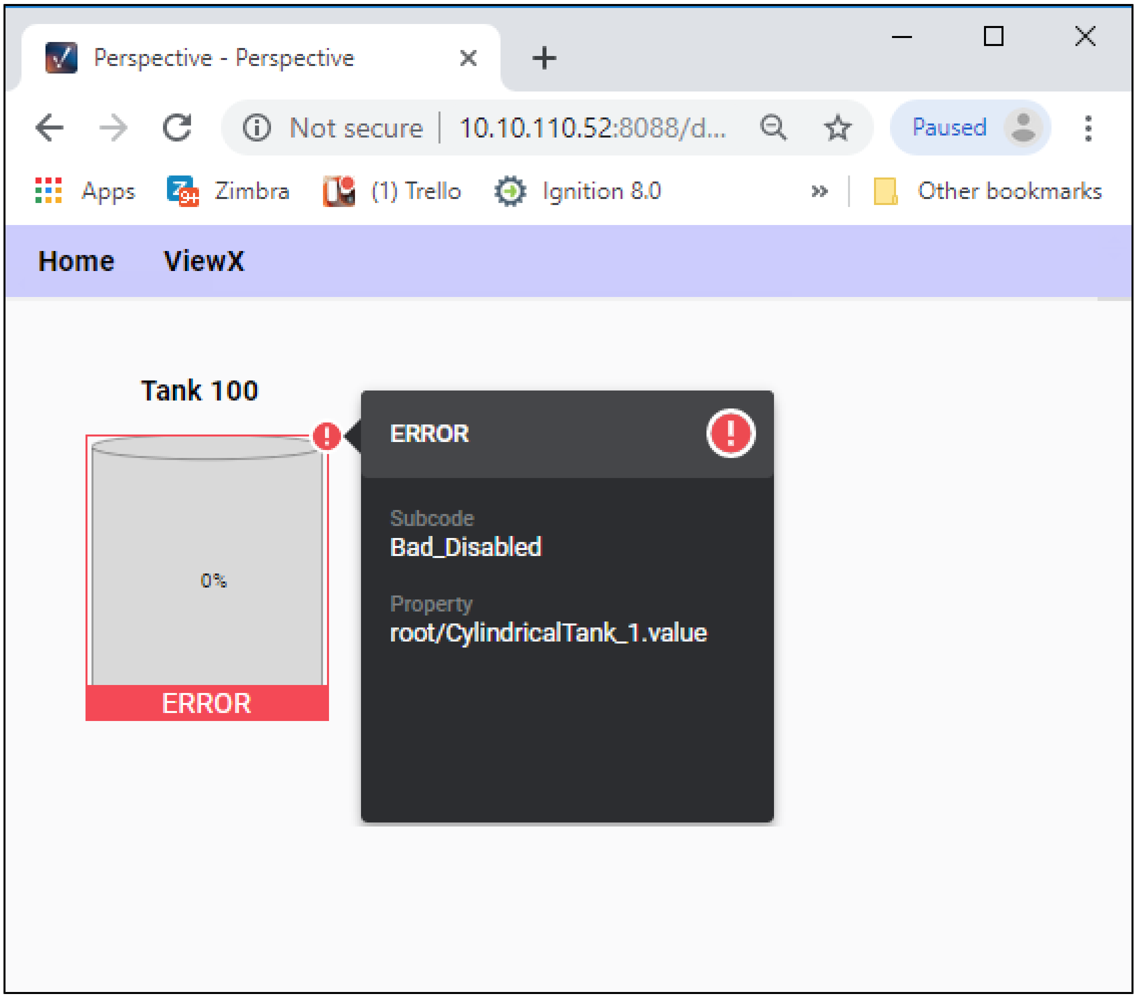
ERROR

Subcode
Error_Configuration

Property
root/LedDisplay_3.value

Component Overlays in a Perspective Session

Component overlays in a Perspective Session work the same way as they do in Preview Mode. The component with the problem will have a red overlay with an exclamation mark. Click on the exclamation mark to open the message box. It will identify the data quality error and affected property to help to identify the problem. To correct the problem, you'll have to go back to the Designer.



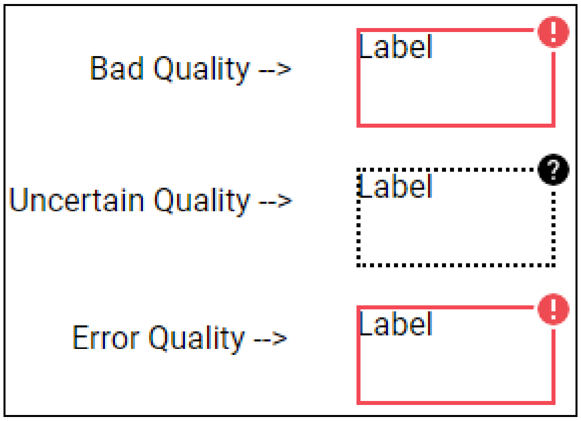
ERROR

Subcode
Bad_Disabled

Property
root/CylindricalTank_1.value

Perspective Quality Code Reference Table

There are four primary quality codes which quickly inform the user of the quality of the Tag: Good, Uncertain, Bad, and Error.



Each quality code has a range of subcodes that provide more specific information about the value of the Tag. The following tables outline the primary data qualities. The most important is **Good** which has a value of **192** and does not have an overlay.

Good Quality	Subcodes	Meaning
Good_Unspec	0	A generic "good" code. Generally used in conjunction with a matching good quality subcode, (1,2, or 192).

ified		
Good_Provisional	1	Good data that should not be considered valid over long periods of time. Provisional values will not cache.
Good_WritePending	2	Used when a write is in progress. Generally, values use this code until the system knows the write through successfully, which would then result in a 192 code.
Good	192	This data has met all criteria for being considered reliable.
Uncertain Quality	**Subcodes 256 - 511**	**Meaning**
Uncertain	256	An unspecified degree of uncertainty exists in this value.
Uncertain_LastKnownValue	257	The current value is unavailable and represents the last known value.
Uncertain_InitialValue	258	Indicates that a subscription has been made and a good value should be arriving shortly.
Uncertain_DataSubNormal	259	Insufficient good-quality sources required for the derivation of this value.
Uncertain_EngineeringUnitsExceeded	260	Indicates that a value has gone beyond its configured engineering units.
Uncertain_IncompleteOperation	261	An async operation is currently pending and its result is unknown.
Bad Quality	**Subcodes 512 - 767**	**Meaning**
Bad	512	General quality for a bad quality.
Bad_Unauthorized	513	An unauthorized request was made for data that requires authorization.
Bad_AccessDenied	514	Data requested that requires credentials not held by the requesting user.
Bad_Disabled	515	Data source is currently not enabled.
Bad_Stale	516	Data is out-of-date based upon the requested refresh interval.
Bad_TrialExpired	517	The Trial Mode's timer expired.
Bad_LicenseExceeded	518	The license limit has been exceeded.
Bad_NotFound	519	Object requested was not found.
Bad_ReferenceNotFound	520	Derived or referenced value required an object which was not found.
Bad_AggregateNotFound	521	Requested aggregate was not found.
Bad_NotConnected	522	A conncection required for this valueis not currently connected.
Bad_GatewayCommOff	523	Connection to the Ignition Gateway is currently turned off.
Bad_OutofRange	524	This value exceeded its allowed range.
Bad_DatabaseNotConnected	525	A database connection required for this value is not connected.
Bad_ReadOnly	526	The target is not writable / read only.

Bad_Failure	527	A "failure" code was received from the underlying system. Additional details may be in the diagnostic message. This generally does not indicate an exception, which would be handled by Error_Exception, but instead a simple failure from a system that can return success or failure.
Bad_Unsupported	528	The operation is not supported by the target.
Error Quality	**Subcodes 768 - 1023**	**Meaning**
Error	768	An unexpected error occurred while retrieving or calculating this value.
Error_Configuration	769	The source of this value is not configured correctly.
Error_ExpressionEval	770	The source expression was unable to be executed.
Error_TagExecution	771	The source Tag could not be executed.
Error_TypeConversion	772	The actual value was not able to be coerced into the configured data type for the source of this value.
Error_DatabaseQuery	773	A database query required for this value caused an error upon execution.
Error_IO	774	An input/output error occurred while attempting to retrieve or calculate this value.
Error_TimeoutExpired	775	An async operation failed dure to a timeout.
Error_Exception	776	An exception was caught, and logged in the relevant system.
Error_InvalidPathSyntax	777	A path (i.e., Tag path, property path, etc.,) was not able to be parsed because the syntax is invalid.
Error_Formatting	778	Attempted formatting (i.e., numeric, date formatting) failed.
Error_ScriptEval	779	A script needed to create this value failed to execute.
Error_CycleDetected	780	Calculating the value involved an execution cycle.

Vision Component Overlays

An overlay on a Vision component lets the operator know that they could be looking at a bad value for that Tag. When the overlay goes away and the values start coming in again, the operator knows that it's a valid Tag, and the values can be trusted.

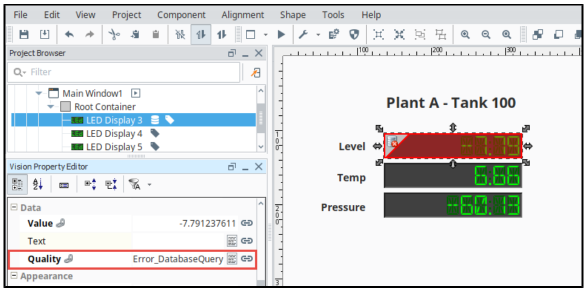
Component Overlays in Designer Mode

In the following example, you see a red overlay with an icon in the top left corner of the selected LED Display component. The icon gives you a clue to the source of the problem. In this example, it is an SQL Database error. In the Vision Property Editor, the Quality property is highlighted and you'll notice there is a "Error_DatabaseQuery" error message.

The overlays table in the next section show all the possible Vision overlays and what they mean.

Tag and Component Overlays

Watch the Video



Value -7.791237611

Text

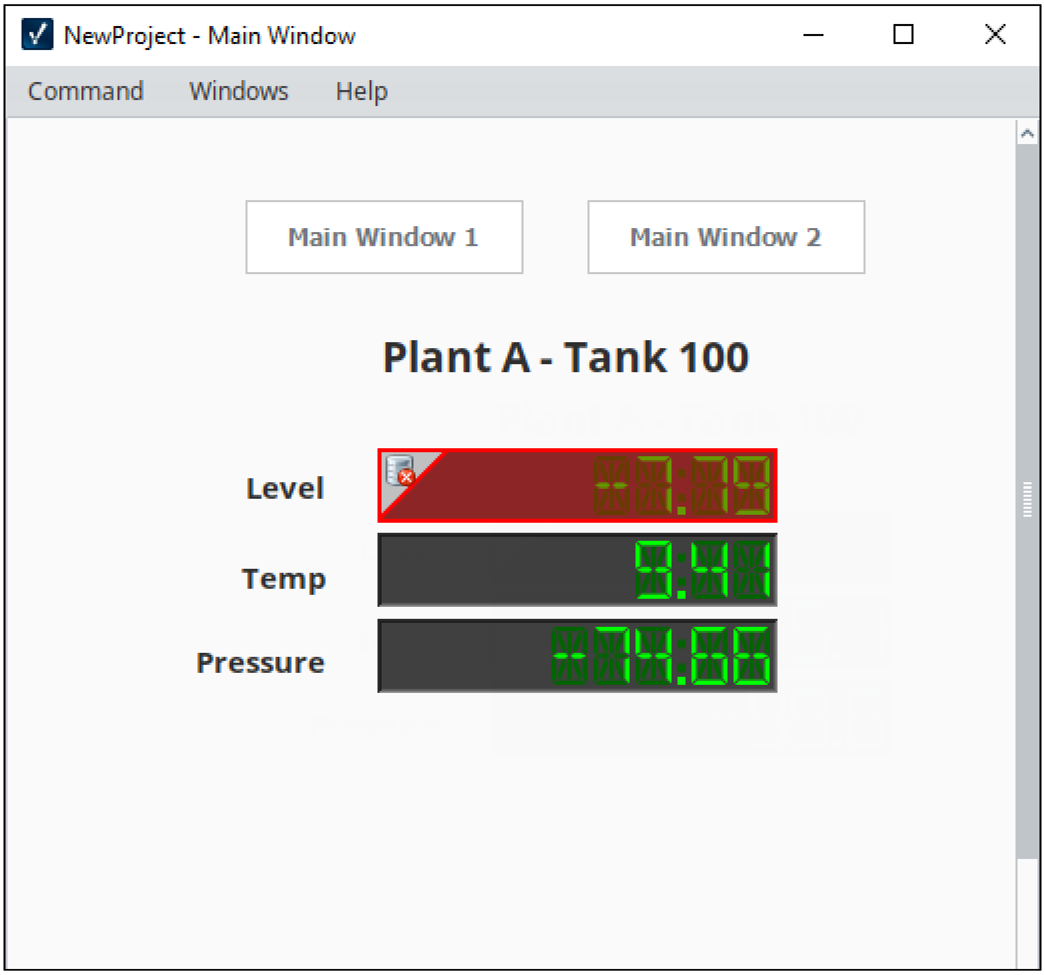
Quality Error_DatabaseQuery

Component Overlays in Preview Mode

Let's switch from the Designer mode to the Preview Mode. To put your active view in **Preview Mode,** press the Preview / Designer mode icon ▶ in the top menubar. Components that have a problem will have a red overlay and an icon in the top left of the component overlay to indicate the problem. The overlay is identical to the overlay that is displayed in the Designer, but the component cannot be selected.

Component Overlays in the Vision Client

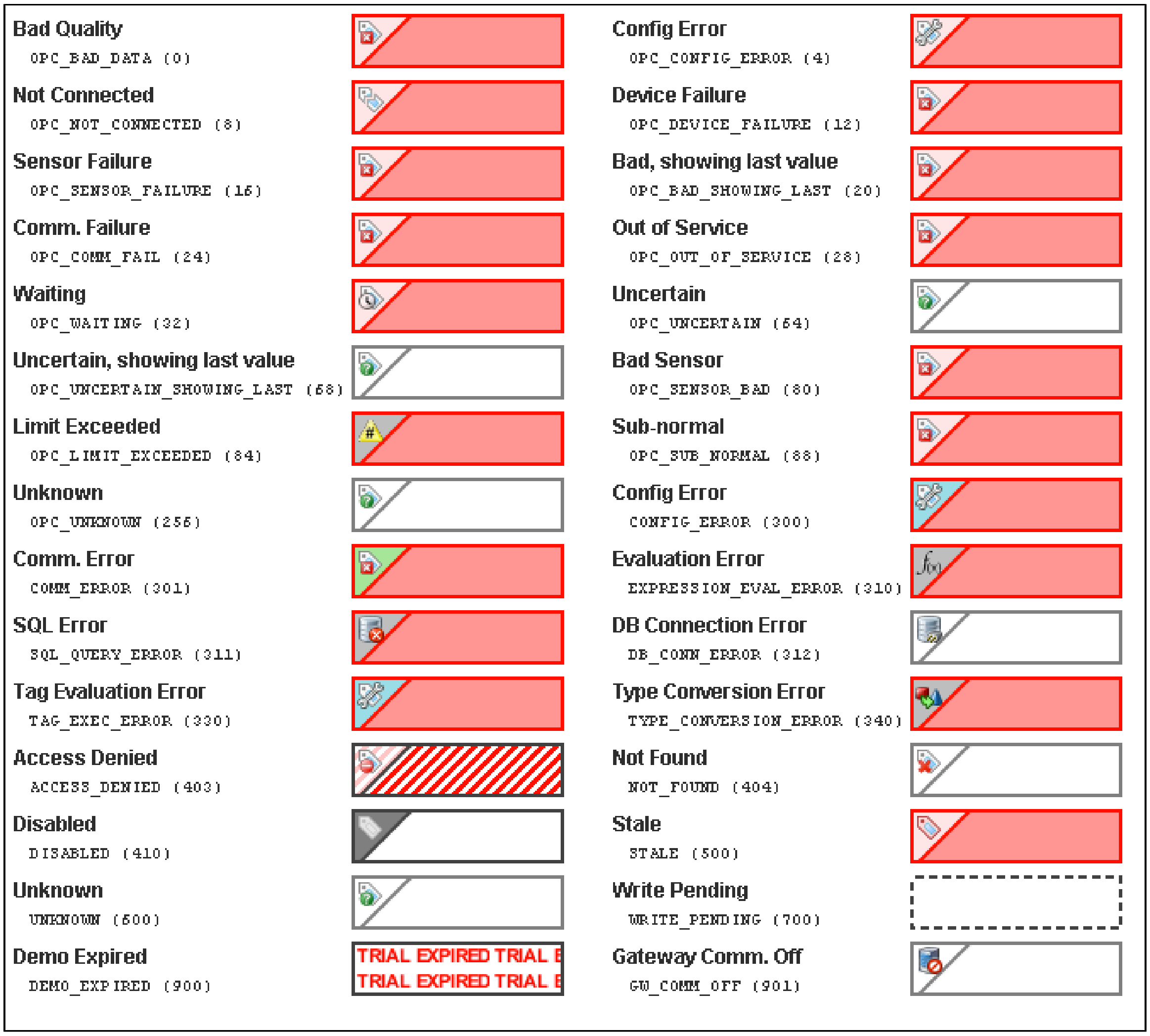
Component overlays in a Vision Client work the same way as they do in Preview Mode of the Designer. You have to look at the icon on the overlay to help you diagnose the problem. Go back to the Designer to correct the problem.



Vision Component Overlay Chart

The quality codes for Vision are different from Perspective's quality codes. Each code has a description and a subcode. When you encounter a component overlay that you are unfamiliar with, check the following table because each quality code has a range of subcodes that provide more specific information about the value of the Tag. The most important is **Good** which has a value of **192**.

The following tables outline the primary data qualities.



Bad Quality	OPC_BAD_DATA (0)		Config Error	OPC_CONFIG_ERROR (4)	
Not Connected	OPC_NOT_CONNECTED (8)		Device Failure	OPC_DEVICE_FAILURE (12)	
Sensor Failure	OPC_SENSOR_FAILURE (16)		Bad, showing last value	OPC_BAD_SHOWING_LAST (20)	
Comm. Failure	OPC_COMM_FAIL (24)		Out of Service	OPC_OUT_OF_SERVICE (28)	
Waiting	OPC_WAITING (32)		Uncertain	OPC_UNCERTAIN (64)	
Uncertain, showing last value	OPC_UNCERTAIN_SHOWING_LAST (68)		Bad Sensor	OPC_SENSOR_BAD (80)	
Limit Exceeded	OPC_LIMIT_EXCEEDED (84)		Sub-normal	OPC_SUB_NORMAL (88)	
Unknown	OPC_UNKNOWN (256)		Config Error	CONFIG_ERROR (300)	
Comm. Error	COMM_ERROR (301)		Evaluation Error	EXPRESSION_EVAL_ERROR (310)	
SQL Error	SQL_QUERY_ERROR (311)		DB Connection Error	DB_CONN_ERROR (312)	
Tag Evaluation Error	TAG_EXEC_ERROR (330)		Type Conversion Error	TYPE_CONVERSION_ERROR (340)	
Access Denied	ACCESS_DENIED (403)		Not Found	NOT_FOUND (404)	
Disabled	DISABLED (410)		Stale	STALE (500)	
Unknown	UNKNOWN (600)		Write Pending	WRITE_PENDING (700)	
Demo Expired	DEMO_EXPIRED (900)	TRIAL EXPIRED TRIAL E TRIAL EXPIRED TRIAL E	Gateway Comm. Off	GW_COMM_OFF (901)	

Vision Quality Code Reference Table

The following table outlines the primary data qualities. The most important is **Good**, and that has a value of **192**. There are more values, but these represent the most common:

Quality	Value	Meaning
Type_Conversion_Error	340	The value of the tag could not be converted to the requested data type. Check the assigned data type of the tag.
Tag_Exec_Error	330	There was an error when evaluating the tag.
Stale	500	The tag has not been evaluated within the expected time frame. There is likely a deeper problem with the tag provider.
OPC_Not_Connected	8	The OPC server driving the tag is not currently connected OR a value has not yet been received by the tag from the server.
OPC_Bad_Data	0	The data is not reliable, further data isn't available.

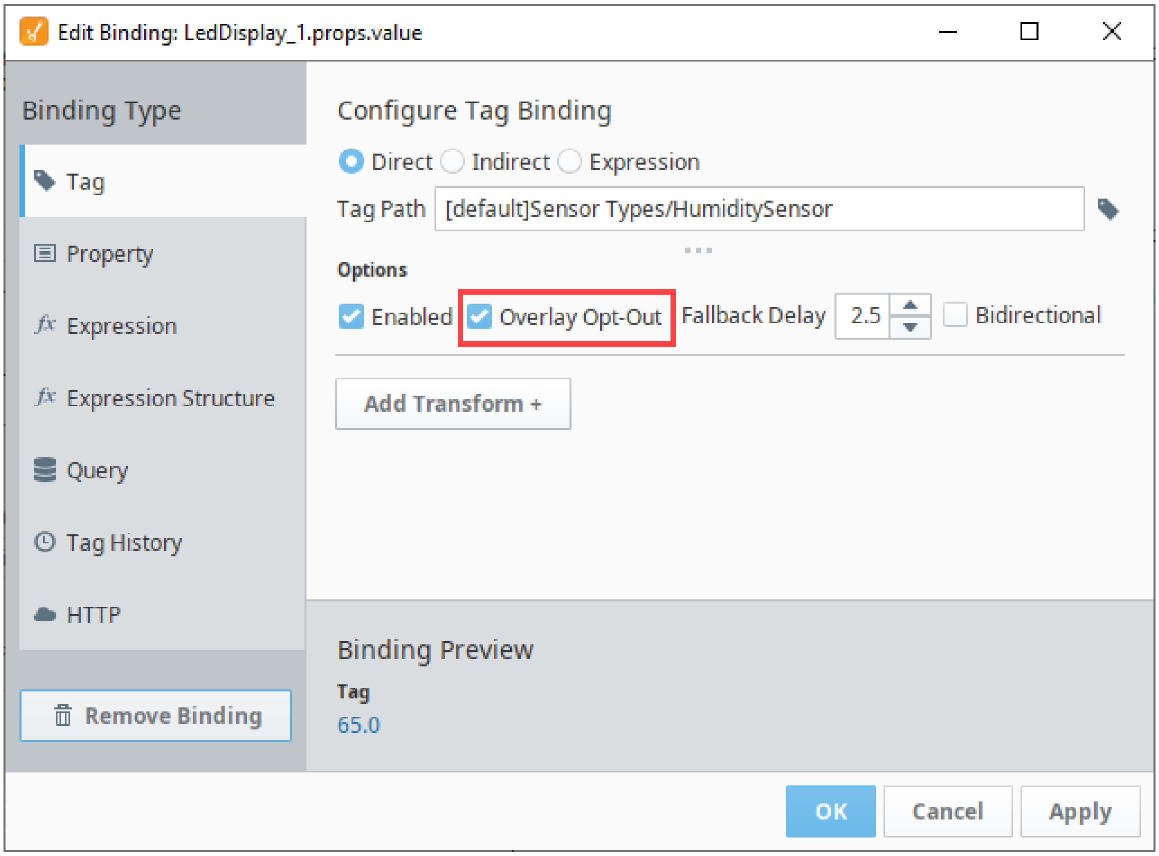
Not_Found	404	The tag, or a tag referenced from inside of it, could not be found (incorrect reference path).
GW_Comm_Off	901	When viewing Tags in the Designer, the Tags will have this value if communication with the Gateway is turned off from the toolbar.
Good (Provisional)	320	Temporary "Good" value. The value isn't cached because the underlying quality may differ than what appears on the Tag.
Good	192	The data has met all criteria for being considered reliable.
Expression_Eval_Error	310	The expression in the Tag generated an error during execution. The error log should provide more information on the error.
Driver_Demo_Timeout	900	The system driving the Tag is operating in demo mode and has timed out.
Disabled	410	The Tag's "enabled" property has been set to false.
Config_Error	300	There is a problem with the tag's configuration. The error log may provide more information as to the exact problem.
Comm_Error	301	There is a problem in communication somewhere between the tag and its data source.
Access_Denied	403	The Tag permission settings do not allow the current user to view the Tag.

Tag Quality and Referenced Tags

When Tags reference other Tags, such as in expressions, they will often pass the worst sub-quality up as their own. For example, even though a particular Tag's expression executes without problem, if the expression references a Tag whose quality is "Bad", the expression Tag will also report "Bad."

Overlay Opt-Out

Choosing the Overlay Opt-Out option will ignore the quality of the chosen Tag, making it have no effect on the component's quality overlay. The Overlay Opt-Out option is located in the Tag bindings for both Perspective and Vision components. If this option is enabled, the operator will not see any overlays and will have no indication that the underlying Tag quality is something other than good. A word of caution when you use the Opt-Out option because you always want to give the operator some indication that the values they are seeing on the screen can be trusted, and by opting out, you are removing that indicator for the operator.



Edit Binding: LedDisplay_1.props.value

Configure Tag Binding

● Direct ○ Indirect ○ Expression

Tag Path [default]Sensor Types/HumiditySensor

Options

☑ Enabled ☑ Overlay Opt-Out Fallback Delay 2.5 □ Bidirectional

Binding Preview

Tag

65.0

Related Topics ...

- Tag Scaling Properties
- Tag Properties
- Bindings in Perspective
- Tag Bindings in Vision
- Indirect Tag Bindings in Vision

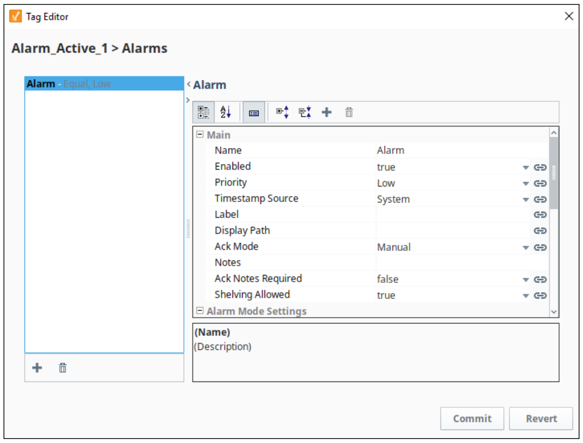
Tag Alarm Properties

Overview

Tags have the ability to define any number of alarms. Each alarm is a condition that will be evaluated when the value of the Tag changes. When the condition becomes true, the alarm is said to be active. When it becomes false, the alarm is said to be cleared.

To add an alarm to a Tag, click the **Edit** ✎ icon next to Alarms. The Alarms screen is displayed.

Click the **Add** ✚ icon to add an alarm. The table below lists all the Tag Alarm property settings.



Tag Editor

Alarm_Active_1 > Alarms

Alarm - Equal, Low

Alarm

Main

Name	Alarm
Enabled	true
Priority	Low
Timestamp Source	System
Label	
Display Path	
Ack Mode	Manual
Notes	
Ack Notes Required	false
Shelving Allowed	true

Alarm Mode Settings

(Name)
(Description)

On this page ...

INDUCTIVE UNIVERSITY

Tag Scripts

Watch the Video

Alarm Property Reference

The table in this section provides a description of alarm properties.

Alarms in Scripting

When interacting with the tags system in via scripting, such as with the system.tag.configure function, alarms are represented as a JSON array of JSON objects, where each object contains the configurations for a single alarm. Thus, any scripting names here are assumed to exist under the alarms array.

Reference Table

Property Name	Binding /Scripting Name	Description	Datatype	Applicable Tag Type
Main				
Name	`name`	Identifier of the alarm. Combined with the location of the alarm, this will be the unique alarm ID. For dynamic values, used Label or Display Path.	String	OPC, Query, Expression, Derived, Reference, Memory
Enabled	`enabled`	This boolean determines whether or not the alarm will be evaluated. A disabled alarm's condition will not be evaluated, and thus will	Boolean	OPC, Query, Expression, Derived,

<table>
<thead>
<tr><th></th><th></th><th></th><th></th><th></th></tr>
</thead>
<tbody>
<tr><td></td><td></td><td>not generate any alarm events.</td><td></td><td>Reference, Memory</td></tr>
<tr><td>Priority</td><td>`priority`</td><td>The priority or severity of this alarm. Alarm priorities can be examined by many other systems in Ignition, including the visualization modules, pipelines, and even scripting.</td><td>String
<table>
<tr><th>Priority</th><th>JSON Name</th></tr>
<tr><td>Diagnostic</td><td>`Diagnostic`</td></tr>
<tr><td>Low</td><td>`Low`</td></tr>
<tr><td>Medium</td><td>`Medium`</td></tr>
<tr><td>High</td><td>`High`</td></tr>
<tr><td>Critical</td><td>`Critical`</td></tr>
</table></td><td>OPC, Query, Expression, Derived, Reference, Memory</td></tr>
<tr><td>Timestamp Source</td><td>`timestampSource`</td><td>Indicates where the timestamp for the alarm event should come from: the system time of when the event was generated (i.e., the Gateway's time), or the timestamp of the value that triggered the the event (i.e., the timestamp of the value from the OPC server).</td><td>String
<table>
<tr><th>Timestamp Source</th><th>JSON Name</th><th>Int Value</th></tr>
<tr><td>System</td><td>`System`</td><td>0</td></tr>
<tr><td>Value</td><td>`Value`</td><td>1</td></tr>
</table></td><td>OPC, Query, Expression, Derived, Reference, Memory</td></tr>
<tr><td>Label</td><td>`label`</td><td>An optional name that will be used for display purposes. Provides a dynamic alternative to the static **name** property. If left blank, the **name** will be used.</td><td>String</td><td>OPC, Query, Expression, Derived, Reference, Memory</td></tr>
<tr><td>Display Path</td><td>`displayPath`</td><td>Optional text that will be used for display and browsing purposes. If this is blank, this property will show the path to the Tag and the name of the alarm instead.</td><td>String</td><td>OPC, Query, Expression, Derived, Reference, Memory</td></tr>
<tr><td>Ack Mode</td><td>`ackMode`</td><td>Dictates how acknowledgement works for the alarm.

- **Unused** - Any alarm event that is generated will automatically be marked as acknowledged.
- **Auto** - The alarm is acknowledged automatically when the alarm becomes cleared.
- **Manual** - The alarm is never set to be acknowledged by the system, and it is up to the user to manually acknowledge alarms.</td><td>String
<table>
<tr><th>Ack Mode</th><th>JSON Name</th><th>Int Value</th></tr>
<tr><td>Unused</td><td>`Unused`</td><td>0</td></tr>
<tr><td>Auto</td><td>`Auto`</td><td>1</td></tr>
<tr><td>Manual</td><td>`Manual`</td><td>2</td></tr>
</table></td><td>OPC, Query, Expression, Derived, Reference, Memory</td></tr>
<tr><td>Notes</td><td>`notes`</td><td>A place for any free-form documentation about the alarm that can be displayed to users.</td><td>String</td><td>OPC, Query, Expression, Derived, Reference, Memory</td></tr>
<tr><td>Ack Notes Required</td><td>`ackNotesReqd`</td><td>If this setting is true, the operators will be required to provide some explanation when the alarm is acknowledged.</td><td>Boolean</td><td>OPC, Query, Expression, Derived, Reference, Memory</td></tr>
<tr><td>Shelving Allowed</td><td>`shelvingAllowed`</td><td>If this setting is false, the shelving feature will be unavailable for this alarm.</td><td>Boolean</td><td>OPC, Query, Expression, Derived,</td></tr>
</tbody>
</table>

<table>
<tr><td></td><td></td><td></td><td></td><td>Reference, Memory</td></tr>
<tr><td colspan="5"><b>Alarm Mode Settings</b></td></tr>
<tr><td>Mode</td><td><code>mode</code></td><td>This setting controls what condition this alarm is evaluating. The available modes are as follows:<ul><li><b>Equal</b> - Active when the Tag's value equals the alarm's setpoint.</li><li><b>Not Equal</b> - Active when the Tag's value does not equal the alarm's setpoint.</li><li><b>Above Setpoint</b> - Active when the Tag's value is above the alarm's setpoint.</li><li><b>Below Setpoint</b> - Active when the Tag's value is below the alarm's setpoint.</li><li><b>Between Setpoints</b> - Active when the Tag's value is between the low and high setpoints. If any change is true, an event will be generated for each value change between the setpoints.</li><li><b>Outside Setpoints</b> - Active when the Tag's value falls outside the low and high setpoints. If any change is true, an event will be generated for each value change outside the setpoints.</li><li><b>Out of Range</b> - The same as Outside Setpoints, but uses the Tag's Engineering High and Engineering Low as the high and low setpoints.</li><li><b>Bad Quality</b> - Active if the Tag value becomes a bad quality, for example, on communication loss.</li><li><b>Any Change</b> - An alarm event is generated every time the Tag value changes. Note that this alarm will never be "active" because each active event is paired with a matching clear event, instantly.</li><li><b>Bit State</b> - This alarm mode is used to alarm when a specific bit out of an integer Tag becomes high. You must specify which bit position to use, with zero being the least significant bit. The On Zero property is used to invert the logic and alarm when the bit is low.</li><li><b>On Condition</b> - This free-form alarm mode is used for when you want to specify the condition using an expression or another Tag. To do this, bind the "Is Active" property to an appropriate expression or Tag.</li></ul></td><td>String<table><tr><th>Mode</th><th>JSON Name</th></tr><tr><td>Equal</td><td><code>Equality</code></td></tr><tr><td>Not Equal</td><td><code>Inequality</code></td></tr><tr><td>Above Setpoint</td><td><code>AboveValue</code></td></tr><tr><td>Below Setpoint</td><td><code>BelowValue</code></td></tr><tr><td>Between Setpoints</td><td><code>BetweenValues</code></td></tr><tr><td>Outside Setpoints</td><td><code>OutsideValues</code></td></tr><tr><td>Out of Range</td><td><code>OutOfEngRange</code></td></tr><tr><td>Bad Quality</td><td><code>BadQuality</code></td></tr><tr><td>Any Change</td><td><code>AnyChange</code></td></tr><tr><td>Bit State</td><td><code>Bit</code></td></tr><tr><td>On Condition</td><td><code>OnCondition</code></td></tr></table></td><td>OPC, Query, Expression, Derived, Reference, Memory</td></tr>
<tr><td><b>Setpoint /Low Setpoint</b></td><td><code>setpointA</code></td><td>Used to determine if the alarm is active by comparing this value to the the tag value.

Some modes under the <b>Mode</b> property allow for multiple setpoints (i.e., a low setpoint and a high setpoint). In these cases, this property is considered to be the Low setpoint.</td><td>Numeric</td><td>OPC, Query, Expression, Derived, Reference, Memory</td></tr>
<tr><td>Inclusive</td><td><code>inclusiveA</code></td><td>If true, the Setpoint value is used inclusively for the condition to alarm.</td><td>Boolean</td><td>OPC, Query, Expression, Derived, Reference, Memory</td></tr>
<tr><td>High Setpoint</td><td><code>setpointB</code></td><td>The high value used to initiate an alarm when the alarm mode calls for two setpoints. Available for modes: Between Setpoint, Outside Setpoints.</td><td>Numeric</td><td>OPC, Query, Expression, Derived, Reference, Memory</td></tr>
</table>

High Inclusive	`inclusiveB`	If true, the High Setpoint value is used inclusively for the condition to alarm. Available for modes: Between Setpoint, Outside Setpoints.	Boolean	OPC, Query, Expression, Derived, Reference, Memory
Any Change	`anyChange`	If true, will alarm on each value change given the alarm mode conditions are met. ⚠ This alarm will never be "active" because each active event is paired with a matching clear event, instantly. Available for modes: Above Setpoint, Below Setpoint, Between Setpoints, and Outside Setpoints.	Boolean	OPC, Query, Expression, Derived, Reference, Memory
On Zero	`bitOnZero`	If true, will alarm when the specified bit is not high (when the bit is 0).	Boolean	OPC, Query, Expression, Derived, Reference, Memory
Bit Position	`bitPosition`	The position of the bit, starting at 0 that will be watched. Available for modes: Bit State.	Numeric	OPC, Query, Expression, Derived, Reference, Memory
Is Active	`activeCondition`	When this property is active, the alarm will be active. Typically has a binding of some sort that will be used to determine when the alarm goes active. If the expression evaluates to True, the alarm is active. If the expression evaluates to False, the alarm is not active.	Boolean	OPC, Query, Expression, Derived, Reference, Memory

Deadbands and Time Delays

<table>
<tr><td>Deadband</td><td><code>deadband</code></td><td>The value for the deadband, interpreted according to the Deadband mode. Note that all alarms are only evaluated after the Tag's value changes, which means that the Tag's own deadband will be considered first.
When the deadband is positive, an active alarm condition needs to clear its setpoint(s) by the amount of the deadband for the alarm to clear. For example, suppose you had a Between Setpoints alarm with a low setpoint of 50 and a high setpoint of 70, and with a deadband of 2. The alarm will go active if the value is between 50 and 70, but will only clear if the value falls below 48 or rises above 72.</td><td>Numeric</td><td>OPC, Query, Expression, Derived, Reference, Memory</td></tr>
<tr><td>Deadband Mode</td><td><code>deadbandMode</code></td><td>Defines how the deadband value is used.<ul><li><b>Absolute</b> - The deadband setting is considered to be an absolute value.</li><li><b>Percent</b> - The actual deadband is calculated as a percent of the Tag's engineering unit span.</li></ul></td><td>Numeric<table><tr><th>Alarming Deadband Mode</th><th>JSON Name</th></tr><tr><td>Absolute</td><td><code>Absolute</code></td></tr><tr><td>Percent</td><td><code>Percent</code></td></tr></table></td><td>OPC, Query, Expression, Derived, Reference, Memoryy</td></tr>
<tr><td>Active Delay</td><td><code>timeOnDelaySeconds</code></td><td>The time, in seconds, before the alarm will be considered active after the alarm's condition becomes true. Also known as a "rising edge time deadband."</td><td>Numeric</td><td>OPC, Query, Expression, Derived, Reference, Memory</td></tr>
</table>

Clear Delay	`timeOffDelaySeconds`	The time, in seconds, before an active alarm will be considered clear after the alarm's condition becomes false. Also known as a "falling edge time deadband."	Numeric	OPC, Query, Expression, Derived, Reference, Memory
Notification Properties				
Active Pipeline	`activePipeline`	The name of an alarm notification pipeline to put this alarm into when it becomes active in order to send out active alarm messages. Many alarms may share a single pipeline.	String	OPC, Query, Expression, Derived, Reference, Memory
Clear Pipeline	`clearPipeline`	The name of an alarm notification pipeline to put this alarm into when it becomes cleared in order to send out cleared messages.	String	OPC, Query, Expression, Derived, Reference, Memory
Ack Pipeline	`ackPipeline`	The name of the alarm notification pipeline to put this alarm into when the alarm has been acknowledged.	String	OPC, Query, Expression, Derived, Reference, Memory
Email Notification Properties				
Custom Subject	`CustomEmailSubject`	A string that will be used as the subject line of an email notification message. If blank, the message settings defined on the notification block that sent the email out will be used instead.	String	OPC, Query, Expression, Derived, Reference, Memory
Custom Message	`CustomEmailMessage`	A string that will be used as the body of this alarm's email notification message. If blank, the message settings defined on the notification block that sent the email out will be used instead.	String	OPC, Query, Expression, Derived, Reference, Memory
SMS Notification Properties				
Custom Message	`CustomSmsMessage`	If specified, will be used for the SMS message. If blank, the message defined in the notification block will be used.	String	OPC, Query, Expression, Derived, Reference, Memory
Associated Data				
User Defined Data		Associated Data are custom alarm properties that can be added to any alarm. These properties will often be bound to other Tags that represent associated contextual data that may be related to the alarm. A snapshot of the values of these properties will be taken when the alarm becomes active. These values will be attached to the alarm event as it moves through the rest of the alarming system, meaning that the values will be available from the alarm status system, the alarm journal system, and in the alarm notification system.	String	OPC, Query, Expression, Derived, Reference, Memory

The following feature is new in Ignition version **8.0.3**
Click here to check out the other new features

Runtime Tag Alarm Properties

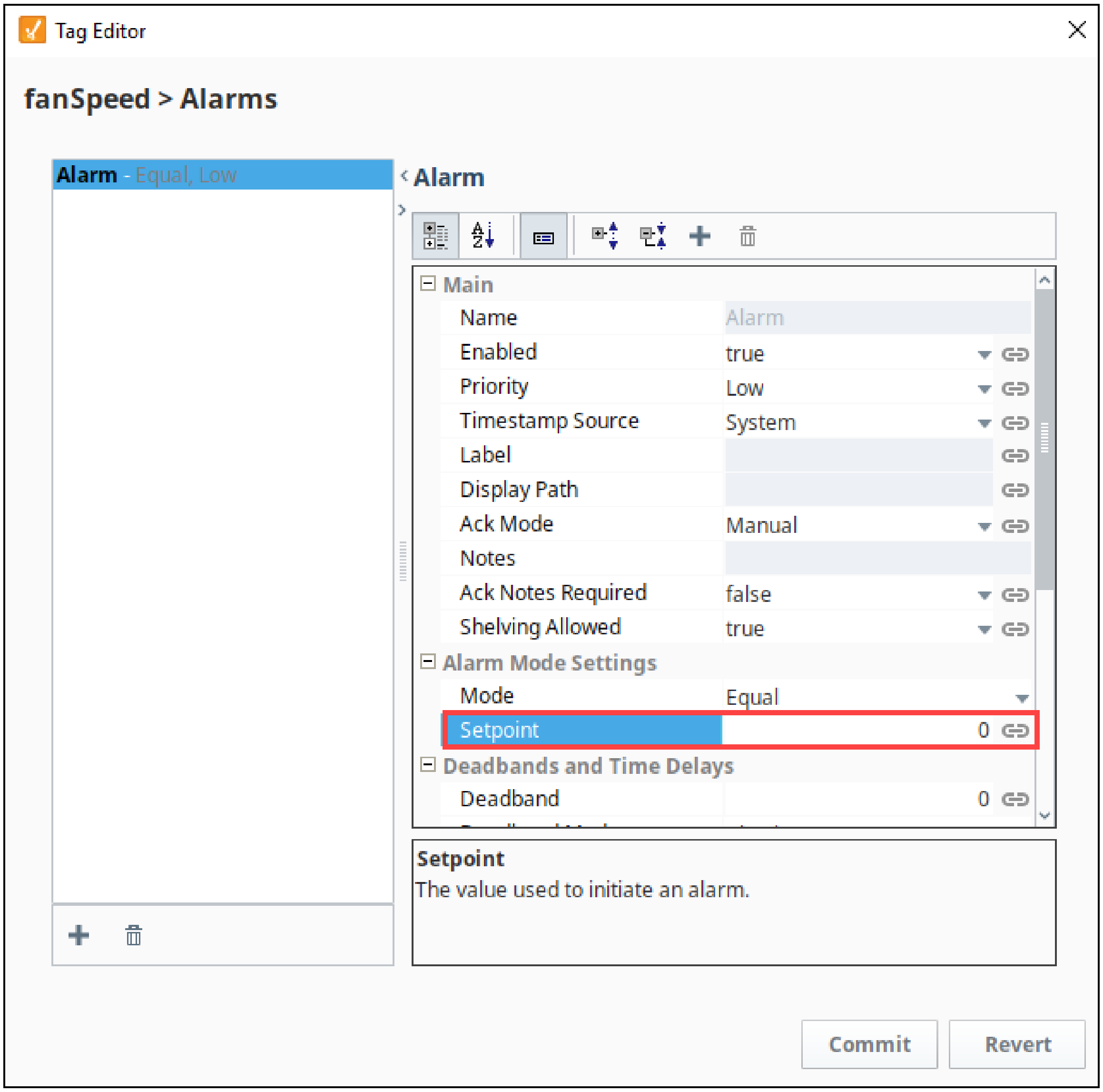
There are a number of very useful Runtime Alarm Tag Properties that expose helpful information on the count of alarms in various states, priority, and shelved alarms.

Property Name	Description
ActiveAckCount	The number of alarms on the Tag that are both Active and Acknowledged.
ActiveUnackCount	The number of alarms on the Tag that are both Active and Unacknowledged.
ClearUnackCount	The number of alarms on the Tag that are both Clear and Unacknowledged.
HasActive	True, if the Tag has at least one Active alarm. False, if there are zero alarms.
HasUnacknowledged	True, if the Tag has at least one Unacknowledged alarm. False, if there are zero Unacknowledged alarms.
HighestAckedName	The Name of the highest Acknowledged alarm, ranked by Priority.
HighestAckedPriority	The highest Priority of all Acknowledged alarms on the Tag.
HighestActiveName	The Name of the highest Active alarm, ranked by Priority.
HighestActivePriority	The highest Priority of all Active alarms on the Tag.
HighestUnackedName	The Name of the highest Unacknowledged alarm, ranked by Priority.
HighestUnackedPriority	The highest Priority of all Unacknowledged alarms on the Tag.
LastActiveTime	A timestamp representing the last time an alarm went Active on the Tag.
ShelvedCount	The number of currently shelved alarms on the Tag.

Binding

Many alarm properties are bindable, which means they can be bound to other Tags in the system, or expressions. For example, you might bind the enabled property to another Tag which represents whether or not your process is running, thereby, disabling the alarm when production is stopped. Another example is you might bind the setpoint of an alarm to a Tag that operators can manipulate, thereby, letting the setpoint be changed at runtime. For more information, see Configuring Alarms.

To bind an alarm property of a Tag, click on the **Binding** 🔗 icon, and the binding UI will slide in from the right.



fanSpeed > Alarms

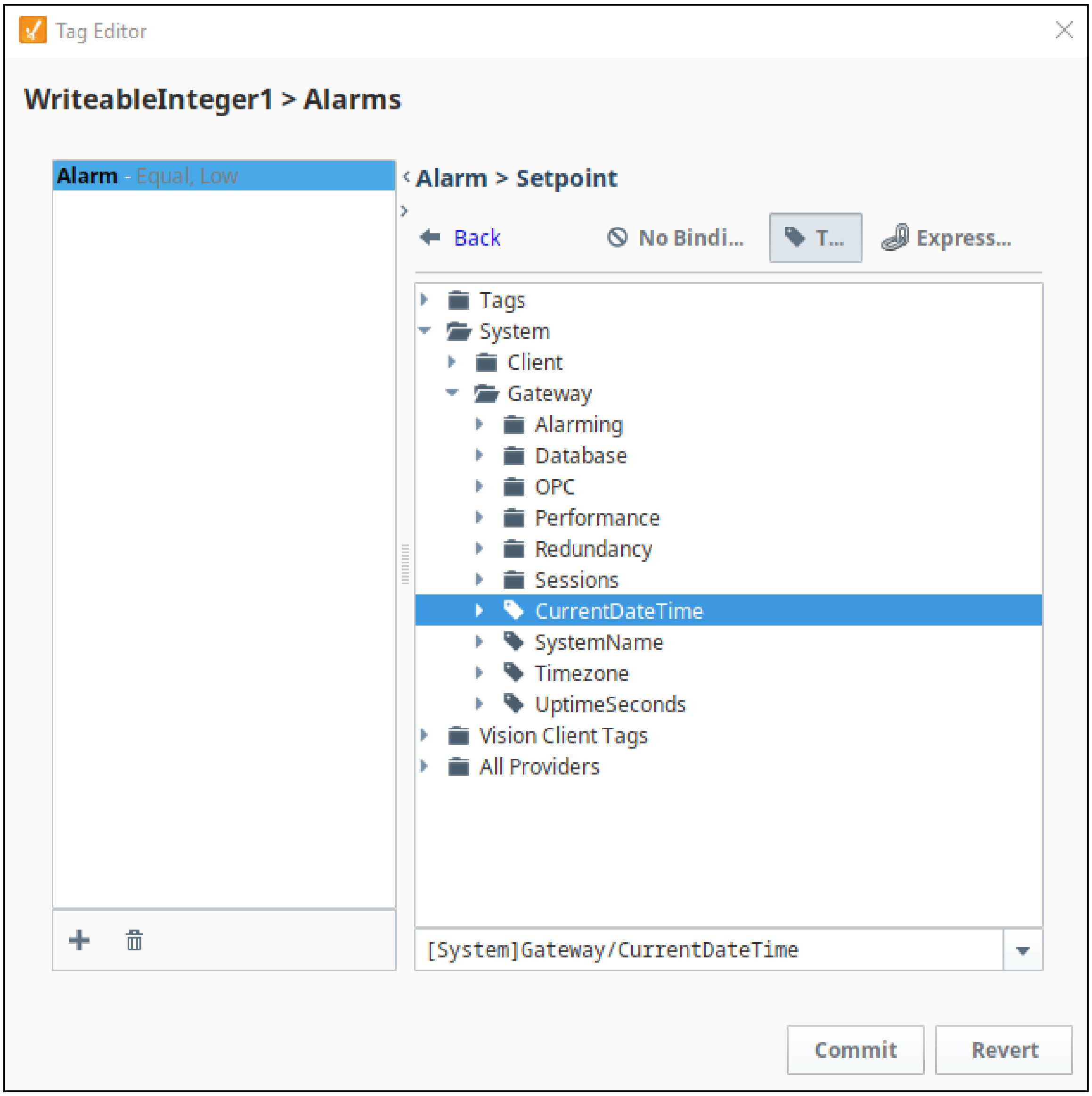
Alarm - Equal, Low

Alarm

Main	
Name	Alarm
Enabled	true
Priority	Low
Timestamp Source	System
Label	
Display Path	
Ack Mode	Manual
Notes	
Ack Notes Required	false
Shelving Allowed	true
Alarm Mode Settings	
Mode	Equal
Setpoint	0
Deadbands and Time Delays	
Deadband	0

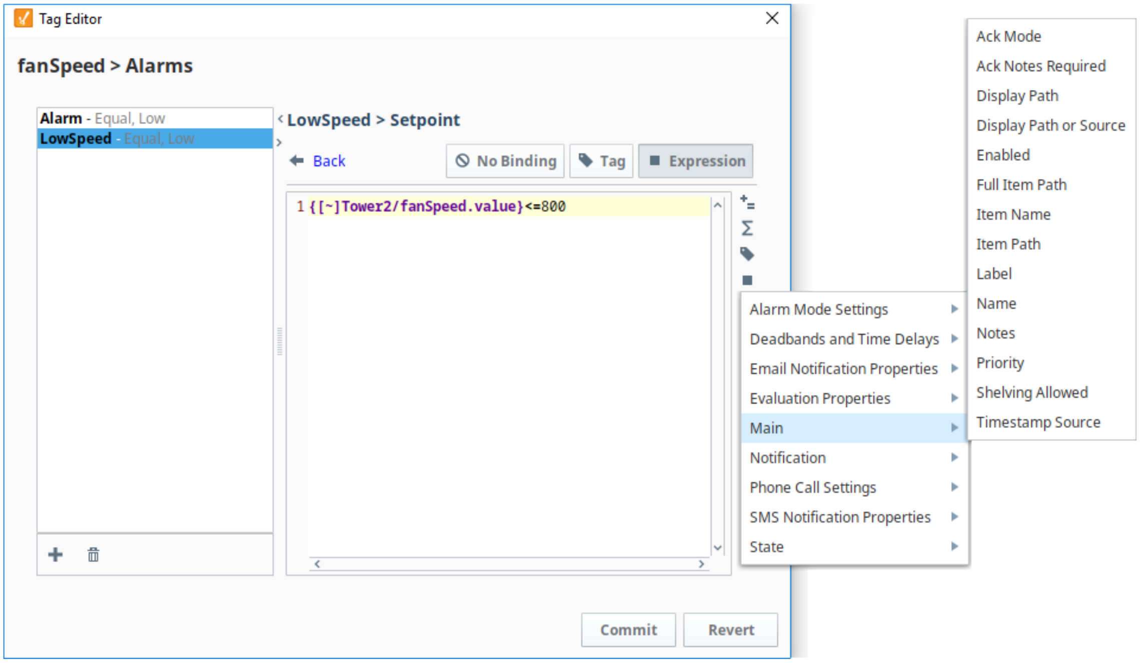
Setpoint
The value used to initiate an alarm.

From here, you can select the binding type (No Binding, Tag, Expression, or UDT Parameter, if applicable).



```
[System]Gateway/CurrentDateTime
```

Binding to an Expression can reference many useful values such as the Tag's value and other settings of the alarm.



```
{[~]Tower2/fanSpeed.value}<=800
```

When you configured the binding to your liking, click the **Commit** button.

Creating Tags

OPC Tags

On this page ...

By default, there is one Internal Tag Provider created for you when Ignition is installed. Because of this, you can get started right away with creating Tags.

Tags are created in the Designer, either manual by right-clicking in the Tag Browser and selecting New Tag, or if you already have a device set up, by opening the OPC Browser and dragging tags or folders in.

There are two ways to create OPC Tags.

- **From OPC Browser**
  Use the OPC Browser to browse and find the Tag and then drag it to the Tag Browser to create the Tag. This method is the easiest and most common way to create Tags.

- **From Tag Editor**
  When the Tag is not available for browsing from the OPC Browser, you create the Tag manually in the Tag Editor window.

Both of these methods require that you first have a device connection made.

Browsing for OPC Tags

INDUCTIVE UNIVERSITY

Browsing for OPC Tags

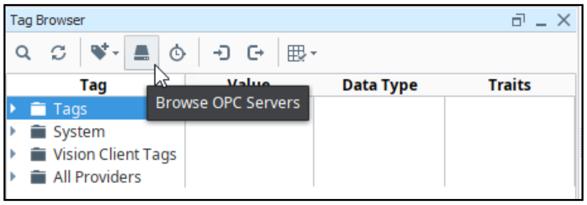
Watch the Video

The easiest and most common way to create Tags in Ignition is by dragging the tags from the OPC Browser into the Tag Browser window in the Designer. With this method you can bring in large numbers of Tags quickly.

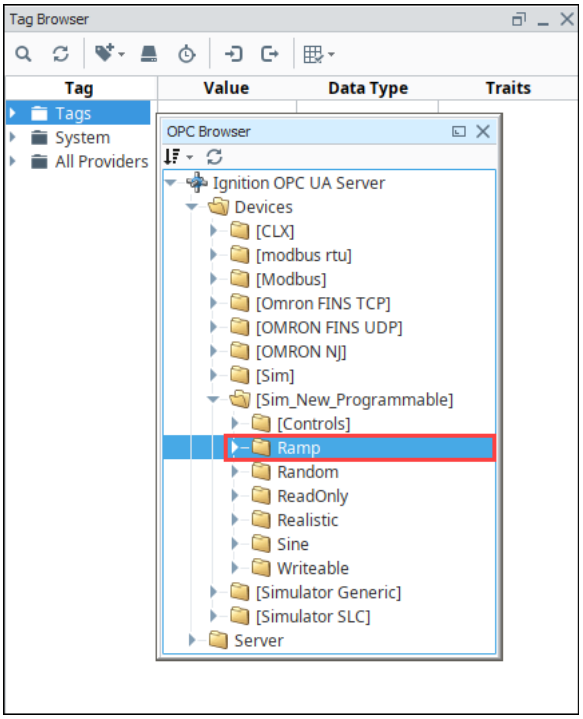
> ⚠ In order to have any OPC tags available in the browser, you must have first added a device connection.

Browse for Tags

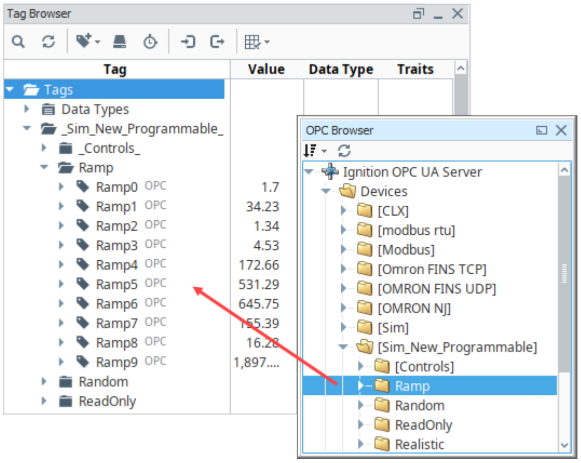
1. In the **Tag Browser**, click the **Browse OPC Servers** icon. The OPC Browser is displayed and you can browse all of your OPC connections. By default, you've got a connection to the internal **Ignition OPC-UA Server**, which has any devices that may be connected. Browse the devices and find some tags that you're interested in.



2. From the **OPC Browser**, highlight the tags that you want to create OPC tags from. You can also select a folder containing many tags.
3. Drag the folders onto the **Tags** folder in the **Tag Browser**. You can drag individual tags or folders. When you drag folders, Ignition keeps the same hierarchy as the PLC.



4. As soon as you drag the tags into Tag Browser, you can see their values being updated in the Tag Browser. You can see their values come in and start updating automatically. By default, they update at a rate of 1 per second.



Tag	Value	Data Type	Traits
Ramp0 OPC	1.7		
Ramp1 OPC	34.23		
Ramp2 OPC	1.34		
Ramp3 OPC	4.53		
Ramp4 OPC	172.66		
Ramp5 OPC	531.29		
Ramp6 OPC	645.75		
Ramp7 OPC	155.39		
Ramp8 OPC	16.28		
Ramp9 OPC	1,897....		

Creating OPC Tags Manually

The above method works for OPC Tags, but only for browsable Tags. For OPC Tags that cannot be obtained through browsing, you can create Tags manually in the Tag Browser.

For manually created tags to work, you must have first added a device connection.

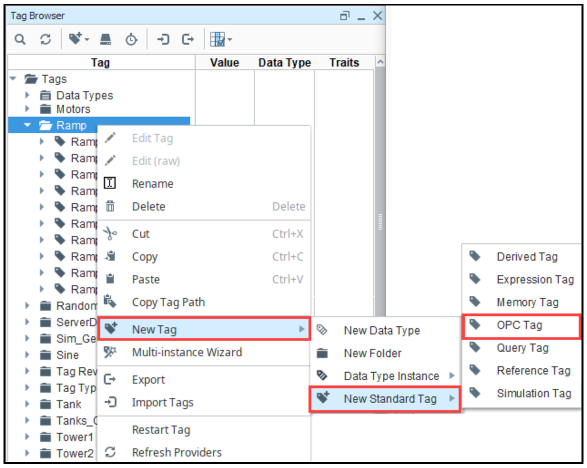
INDUCTIVE UNIVERSITY

Creating OPC Tags Manually

Watch the Video

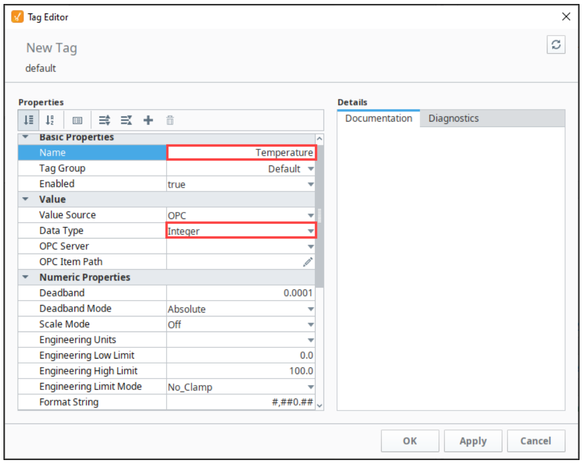
Manually Create an OPC Tag

1. To start, click on the **New Tag** icon or right-click on the provider node and select **New Tag > New Standard Tag > OPC Tag**.



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

   Name: **Temperature**
   Data Type:**Integer**



Tag Editor

New Tag
default

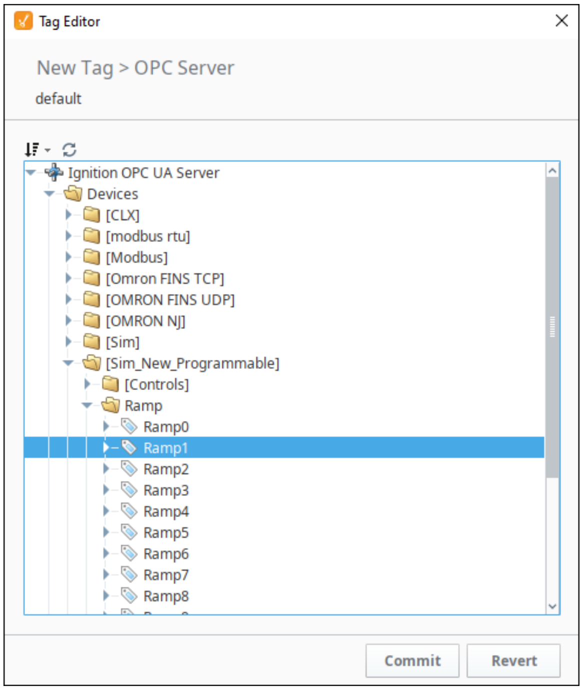
Properties

Basic Properties	
Name	Temperature
Tag Group	Default
Enabled	true
Value	
Value Source	OPC
Data Type	Integer
OPC Server	
OPC Item Path	
Numeric Properties	
Deadband	0.0001
Deadband Mode	Absolute
Scale Mode	Off
Engineering Units	
Engineering Low Limit	0.0
Engineering High Limit	100.0
Engineering Limit Mode	No_Clamp
Format String	#,##0.##

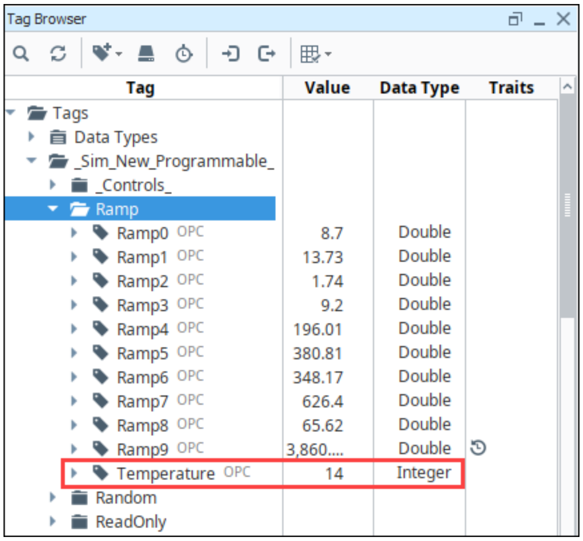
3. Next set the OPC Server and Item Path as follows:

   a. OPC Server: Click on the **Edit** icon and choose the OPC server you want to use. Alternately, selecting an Item Path will automatically fill in the appropriate OPC Server.

b. OPC Item Path: Click on the **Edit** icon then navigate to the path you want.



4. Click **Commit**.
5. Click **OK** to exit the Tag Editor. Now you can see the Temperature Tag in the Tag Browser.



Tag	Value	Data Type	Traits
Tags			
Data Types			
_Sim_New_Programmable_			
Controls			
Ramp			
Ramp0 OPC	8.7	Double	
Ramp1 OPC	13.73	Double	
Ramp2 OPC	1.74	Double	
Ramp3 OPC	9.2	Double	
Ramp4 OPC	196.01	Double	
Ramp5 OPC	380.81	Double	
Ramp6 OPC	348.17	Double	
Ramp7 OPC	626.4	Double	
Ramp8 OPC	65.62	Double	
Ramp9 OPC	3,860....	Double	⟲
Temperature OPC	14	Integer	
Random			
ReadOnly			

Creating Memory Tags

Memory Tags are simple tags that do not automatically change value. You can use them as setpoints (that are not stored in the PLC) or just for storing extra information. The value is specified during configuration and is stored globally in the Gateway when written.

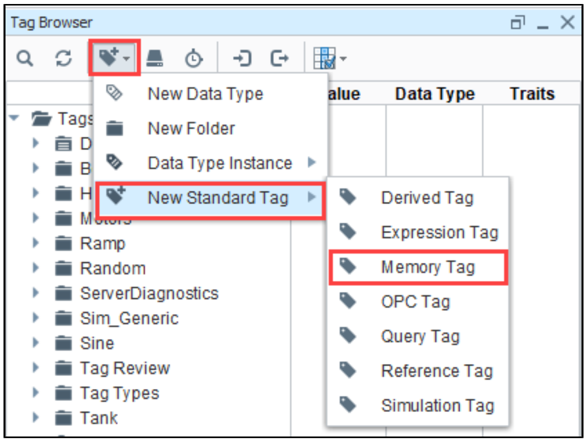
Create a Memory Tag

1. From Tag Browser, click on the New Tag icon or right click on a folder and select **New Tag > New Standard Tag > MemoryTag**.

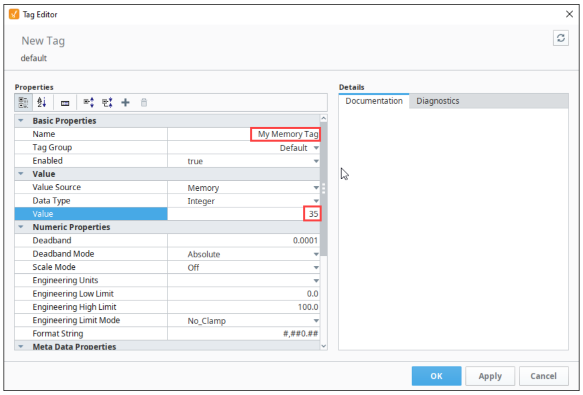
INDUCTIVE UNIVERSITY

Memory Tags

Watch the Video



2. In the Tag Editor, type a **Name** (for example, Memory Tag), initial **Value**, and **Data Type**.



Basic Properties	
Name	My Memory Tag
Tag Group	Default
Enabled	true
Value	
Value Source	Memory
Data Type	Integer
Value	35
Numeric Properties	
Deadband	0.0001
Deadband Mode	Absolute
Scale Mode	Off
Engineering Units	
Engineering Low Limit	0.0
Engineering High Limit	100.0
Engineering Limit Mode	No_Clamp
Format String	#,##0.##
Meta Data Properties	

3. Click **OK**. You now have a Memory Tag that you can read from or write to. This is just like having a global variable for your project.

Creating Query Tags

Query Tags execute a SQL Query, whose result provides the value for the Tag. Like SQL binding in Vision, SQL Query Tags can reference other Tags to build dynamic queries. Query Tags cannot be written to. The value of a Query Tag comes from the database through a SQL query.

Create a Query Tag

In this example, we'll create an query tag that selects the current timestamp.

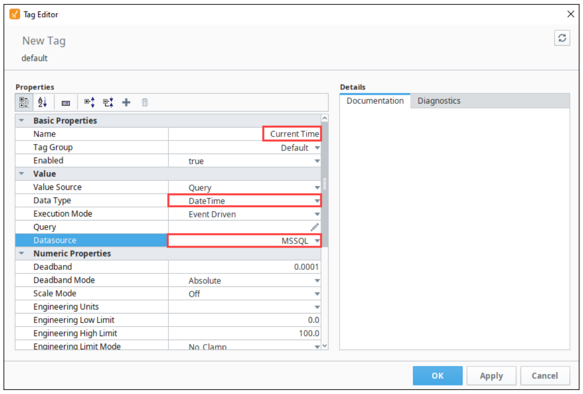
1. From Tag Browser, click on the New Tag icon or right click on a folder and select **New Standard Tag > Query Tag**.
2. In the Tag Editor, enter:

   Name: **Current Time**
   Data Type: **DateTime**
   Datasource: **MySQL**

INDUCTIVE UNIVERSITY

Query Tags

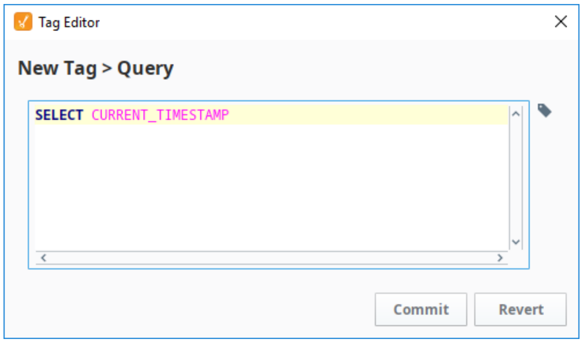
Watch the Video



Basic Properties	
Name	Current Time
Tag Group	Default
Enabled	true
Value	
Value Source	Query
Data Type	DateTime
Execution Mode	Event Driven
Query	
Datasource	MSSQL
Numeric Properties	
Deadband	0.0001
Deadband Mode	Absolute
Scale Mode	Off
Engineering Units	
Engineering Low Limit	0.0
Engineering High Limit	100.0
Engineering Limit Mode	No Clamp

3. Next to the Query property, click on the Edit ✎ icon.
4. Copy in the following query:

```
SELECT CURRENT_TIMESTAMP
```



```
SELECT CURRENT_TIMESTAMP
```

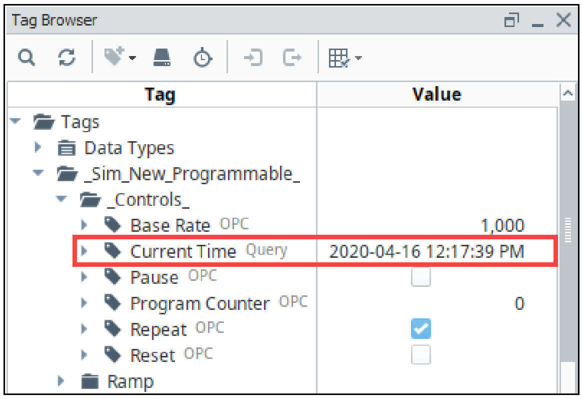
Query Type

In certain circumstances you may want to update the database instead of querying it for data. The Query Tag will accept an update query and update the target data source. The value attribute of the Tag becomes the number of rows affected by the update query.

The data source refers to the default data source of the Tag provider as opposed to the default data source of the project.

In the Gateway **Configure** Section, go to **Tags > Realtime** and edit the appropriate Tag provider. Set the default database connection that this Tag provider will use for its Query Tags.

5. Click **Commit** then click **OK**. You will see the value from your SQL query on the **Current Time** Tag.



Tag	Value
Tags	
Data Types	
_Sim_New_Programmable_	
Controls	
Base Rate OPC	1,000
Current Time Query	2020-04-16 12:17:39 PM
Pause OPC	□
Program Counter OPC	0
Repeat OPC	☑
Reset OPC	□
Ramp	

Creating Expression and Derived Tags

Expression Tags are driven by an expression. The expression syntax is the same as for property bindings and allows mathematical operations, references to other Tags, and logic operations. Expression Tags cannot be written to.

With Expression Tags you can do a calculation based on certain values, other Tags, or any one of the built-in functions. In this way, you can do the calculation one time and view it on any window in your project. For information on Ignition's Expression language, see Expression Overview and Syntax.

A Derived Tag is an abstracted Tag that refers to another Tag. They are similar conceptually to Expressions Tags that reference another Tag, but Derived Tags have some additional functionality. Namely, they can write back to the referenced Tag.

For information on Ignition's Expression language, see Expression Overview and Syntax.

INDUCTIVE UNIVERSITY

Expression Tags

Watch the Video

INDUCTIVE UNIVERSITY

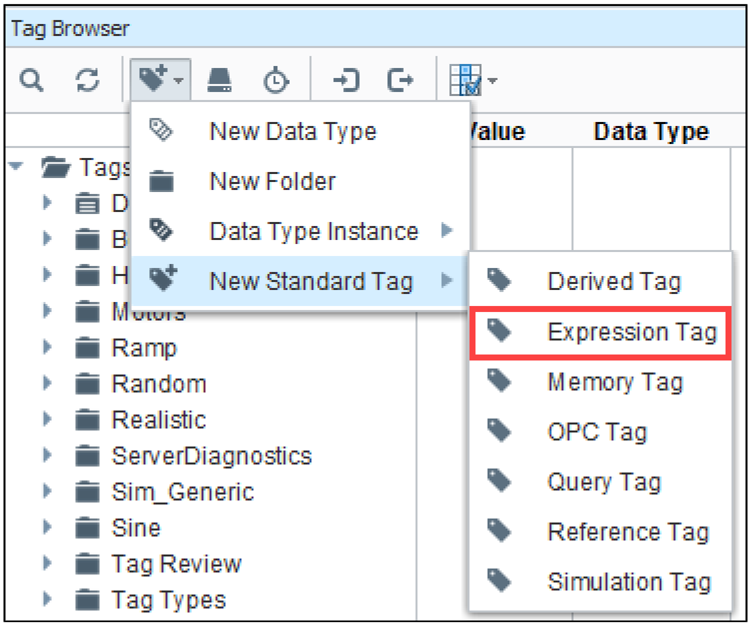
Derived Tags

Watch the Video

Create an Expression Tag

In this example, we'll create an expression tag that converts Fahrenheit to Celsius.

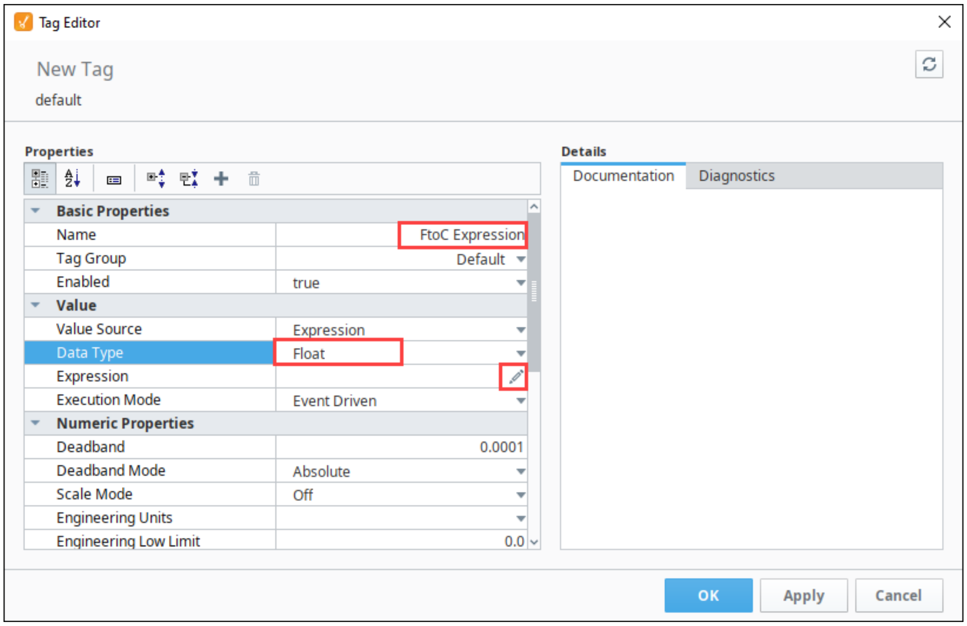
1. From Tag Browser, click on the **New Tag** icon or right click on a folder and select **New Standard Tag > Expression Tag**.



2. In the Tag Editor, enter:

   Name: **FtoC Expression**
   Data Type: **Float**

   Next to the Expression property, click on the **Edit** icon.

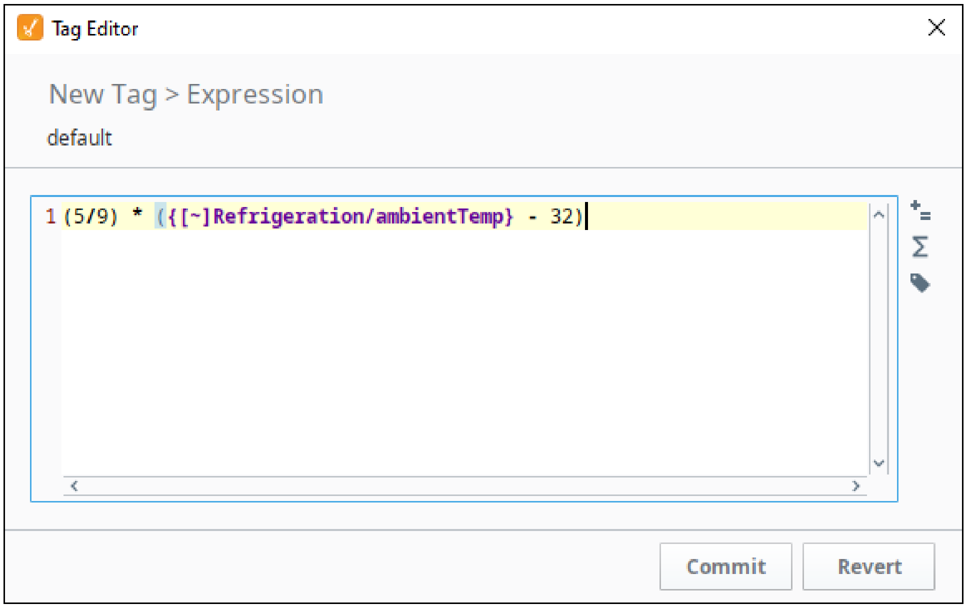


New Tag
default

Basic Properties	
Name	FtoC Expression
Tag Group	Default
Enabled	true
Value	
Value Source	Expression
Data Type	Float
Expression	
Execution Mode	Event Driven
Numeric Properties	
Deadband	0.0001
Deadband Mode	Absolute
Scale Mode	Off
Engineering Units	
Engineering Low Limit	0.0

3. In the **New Tag > Expression** window, type in any expression using Ignition's Expression Language. See Expression Functions in the Appendix for information on all expression operations and functions.
   Copy in the following expression:

```
(5/9) * ({[~]Refrigeration/ambientTemp} - 32)
```

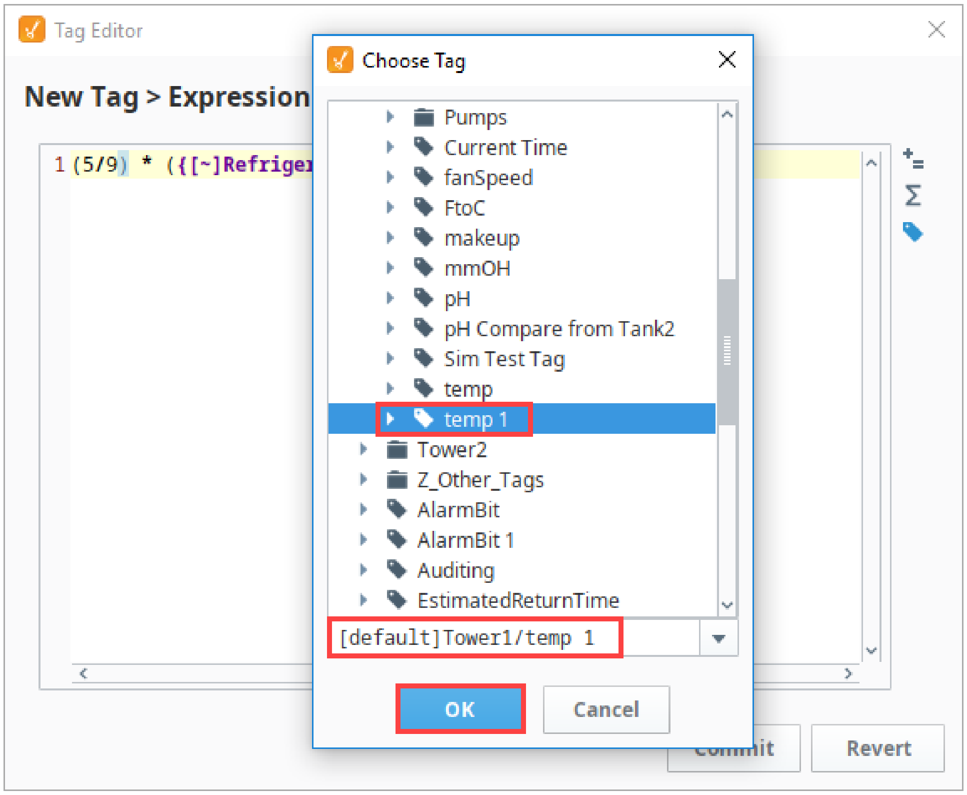


New Tag > Expression
default

```
(5/9) * ({[~]Refrigeration/ambientTemp} - 32)
```

4. If you want to use a different Tag, replace the `{[~]Refrigeration/ambientTemp}` Tag path with any Tag of your choice.

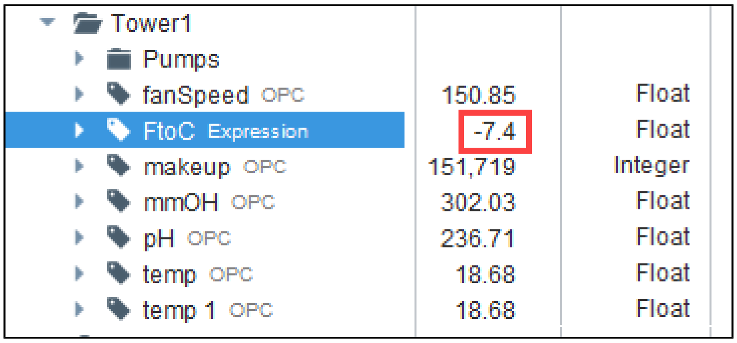
   Instead of typing the referenced tag path, you can use the **Tag** 🏷 icon on the right side of the **Tag Editor > Expression** window to select the Tag path. In the example we chose a temperature Tag from a tower.



```
1 (5/9) * ({[~]Refriger
```

```
[default]Tower1/temp 1
```

5. Click **OK** to accept the Tag you've highlighted.
6. Click **Commit** in the **New Tag > Expression** window.
7. Click **OK** to save the Tag . You will see the value from your calculation on the Tag. Note that the Expression Tags are evaluated at the rate the Tag Group is set to.



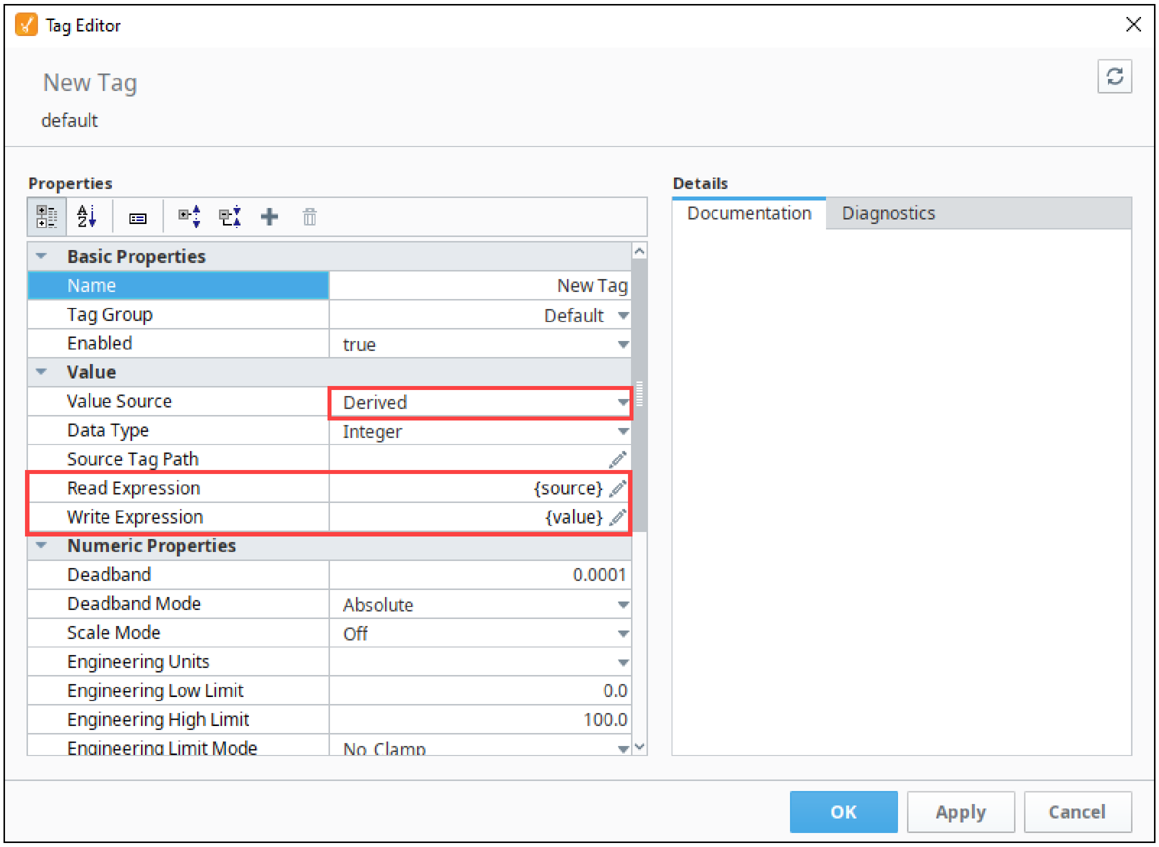
Tower1		
Pumps		
fanSpeed OPC	150.85	Float
FtoC Expression	-7.4	Float
makeup OPC	151,719	Integer
mmOH OPC	302.03	Float
pH OPC	236.71	Float
temp OPC	18.68	Float
temp 1 OPC	18.68	Float

Create a Derived Tag

Creating a Derived Tag is nearly identical to creating an Expression tag, with the exception that you can provide both a Read Expression and a Write Expression for the tag. Follow the same steps as for an Expression tag.

1. From Tag Browser, click on the New Tag icon or right click on a folder and select **New Standard Tag > Derived Tag**.
2. Once in the Tag editor, click on the **Edit** icon next to the Read Expression property.
3. In the New Tag > Read Expression window, type in any expression using Ignition's Expression Language. Click **Commit**.
4. Back in the Tag editor, click on the **Edit** icon next to the Write Expression property.
5. In the New Tag > Write Expression window, type in any expression using Ignition's Expression Language. Click **Commit**.
6. In the Tag Editor, click **OK**.

See Expression Functions in the Appendix for information on all expression operations and functions.



Basic Properties	
Name	New Tag
Tag Group	Default
Enabled	true
Value	
Value Source	Derived
Data Type	Integer
Source Tag Path	
Read Expression	{source}
Write Expression	{value}
Numeric Properties	
Deadband	0.0001
Deadband Mode	Absolute
Scale Mode	Off
Engineering Units	
Engineering Low Limit	0.0
Engineering High Limit	100.0
Engineering Limit Mode	No Clamp

See Expression Functions in the Appendix for information on all expression operations and functions.

Creating Reference Tags

A Referenced Tag is a Tag that refers to another Tag. It can be used to create an alias for the Tag. The Referenced Tag can write back to the original Tag.

INDUCTIVE UNIVERSITY

Reference Tags

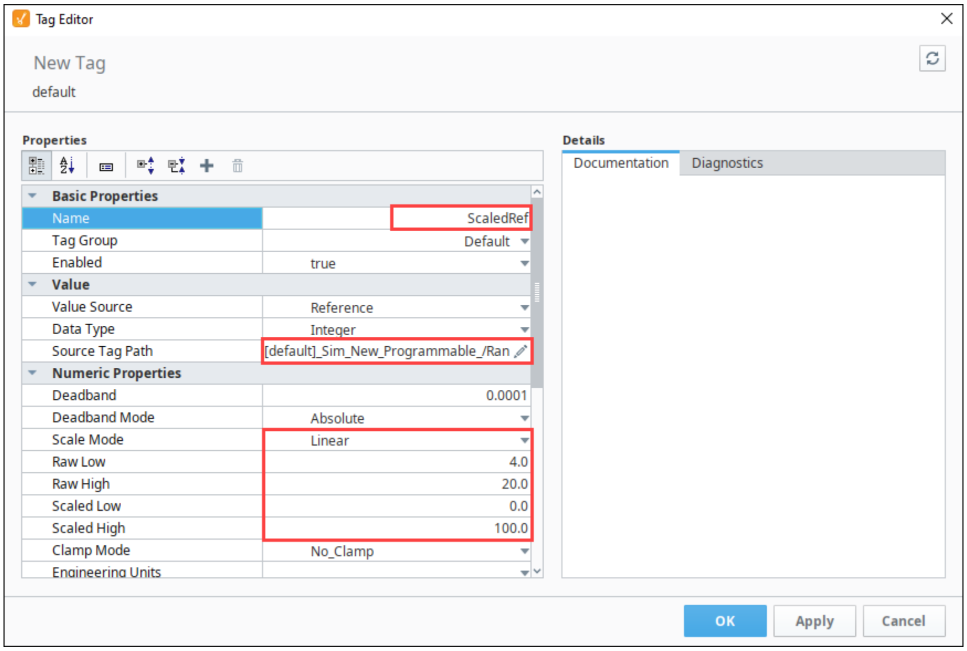
Watch the Video

Create a Referenced Tag

In this example, we'll create an Referenced Tag that is scaled.

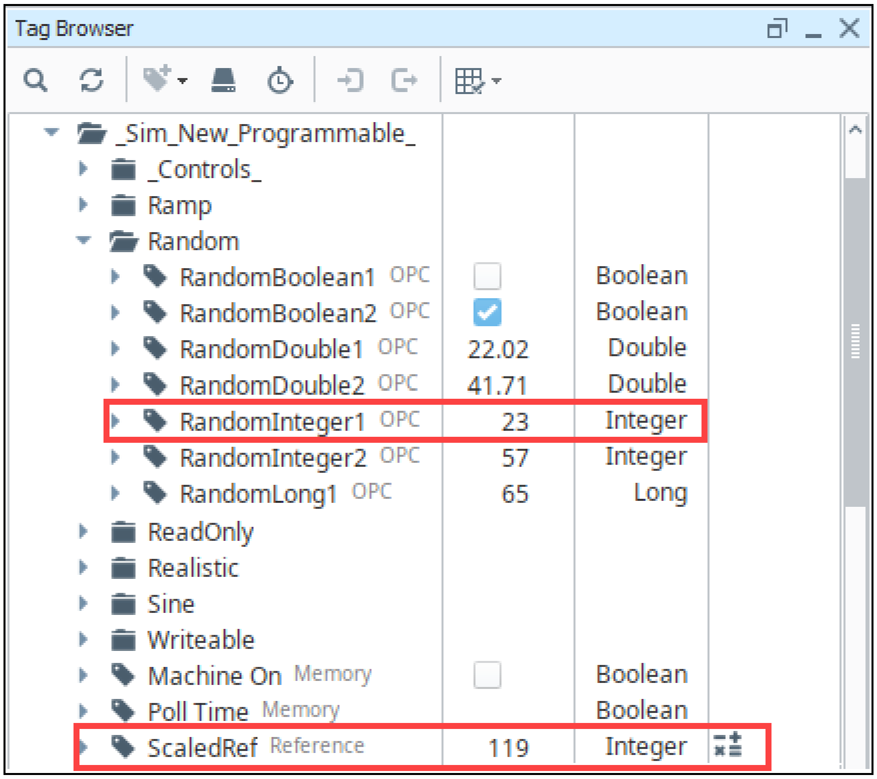
1. From Tag Browser, click on the New Tag icon or right click on a folder and select **New Standard Tag > Reference Tag**.
2. In the Tag Editor, enter:

   Name: ScaledRef
   Source Tag Path: [default]_Sim_New_Programmable_/Random/RandomInteger1 (or any Tag you want to reference)
   Scale Mode: Linear
   Raw Low: 4.0
   Raw High: 20.0
   Scaled Low: 0.0
   Scaled High: 10.0



Property	Value
Basic Properties	
Name	ScaledRef
Tag Group	Default
Enabled	true
Value	
Value Source	Reference
Data Type	Integer
Source Tag Path	[default]_Sim_New_Programmable_/Ran
Numeric Properties	
Deadband	0.0001
Deadband Mode	Absolute
Scale Mode	Linear
Raw Low	4.0
Raw High	20.0
Scaled Low	0.0
Scaled High	100.0
Clamp Mode	No_Clamp
Engineering Units	

3. Click **OK**. The Reference Tag is created and shows up in the Tag Browser. Note that the Scaled ± icon appears indicating that scaling has been applied to the Tag.



Tag	Type	Value	Data Type	
_Sim_New_Programmable_				
Controls				
Ramp				
Random				
RandomBoolean1	OPC	□	Boolean	
RandomBoolean2	OPC	☑	Boolean	
RandomDouble1	OPC	22.02	Double	
RandomDouble2	OPC	41.71	Double	
RandomInteger1	OPC	23	Integer	
RandomInteger2	OPC	57	Integer	
RandomLong1	OPC	65	Long	
ReadOnly				
Realistic				
Sine				
Writeable				
Machine On	Memory	□	Boolean	
Poll Time	Memory		Boolean	
ScaledRef	Reference	119	Integer	±

Editing Tags

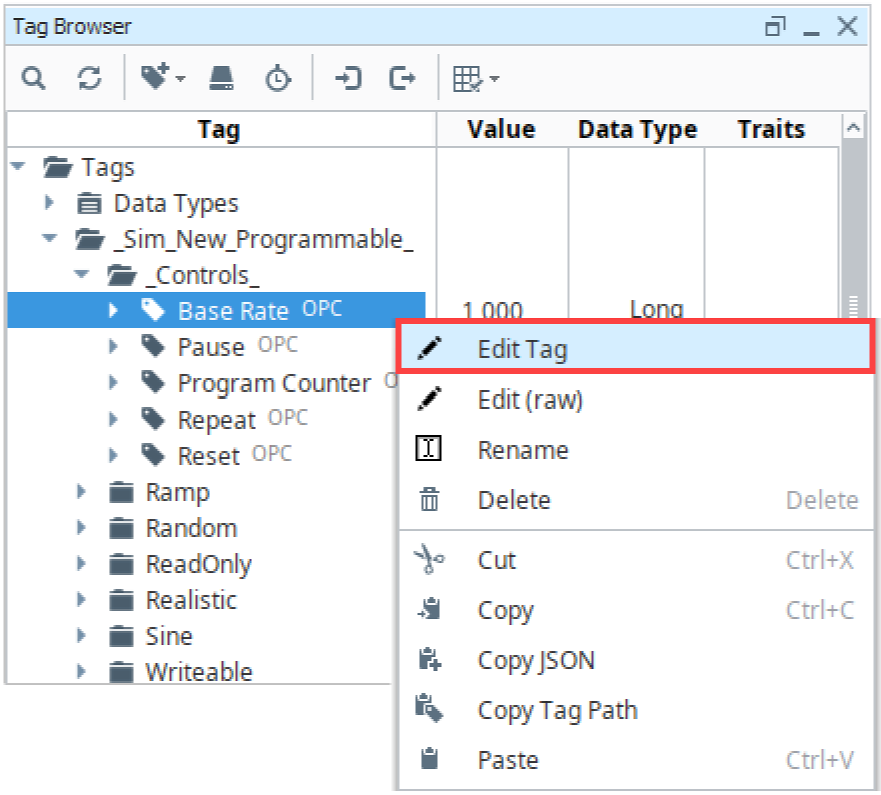
The Tag Editor is a powerful tool used when creating Tags and it can also be used for editing them. The properties displayed in the Tag Editor are custom to the type of Tag you've selected. You can find additional information on Understanding Tags and the Tag right click menu here.

The Tag Editor

Watch the Video

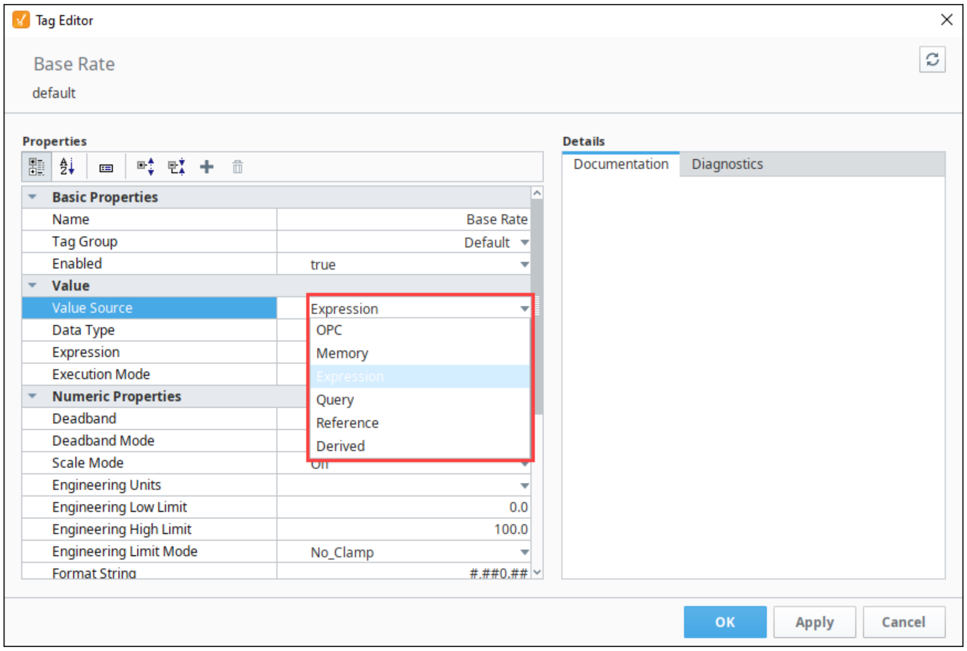
Edit a Tag

1. To edit an existing Tag, right-click on the Tag, and select the **Edit Tag** icon.



2. Once in the Tag Editor, you can change the properties as you wish.
   For example, if you want to change the Tag to a different type – such as from OPC to Expression – go to the Value Source property, click the **Expand** icon, and choose the type of Tag (Simulation, Memory, Expression, Query, Reference, or Derived) that you

want.

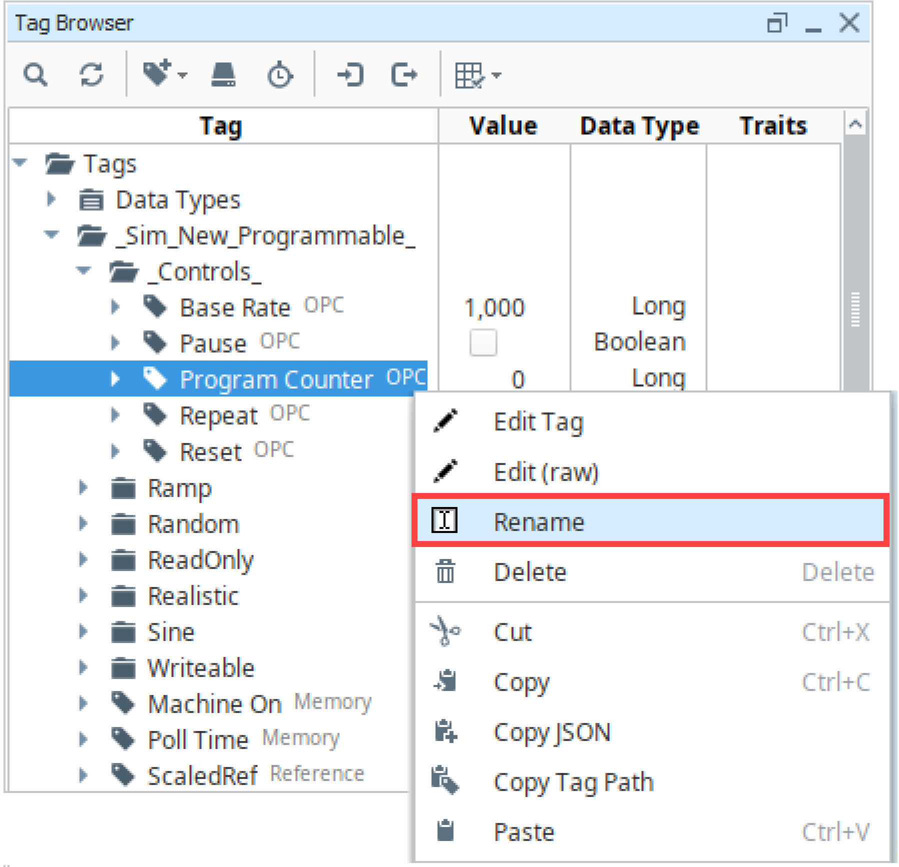


Basic Properties	
Name	Base Rate
Tag Group	Default
Enabled	true
Value	
Value Source	Expression
Data Type	
Expression	
Execution Mode	
Numeric Properties	
Deadband	
Deadband Mode	
Scale Mode	Off
Engineering Units	
Engineering Low Limit	0.0
Engineering High Limit	100.0
Engineering Limit Mode	No_Clamp
Format String	#,##0.##

Rename a Tag

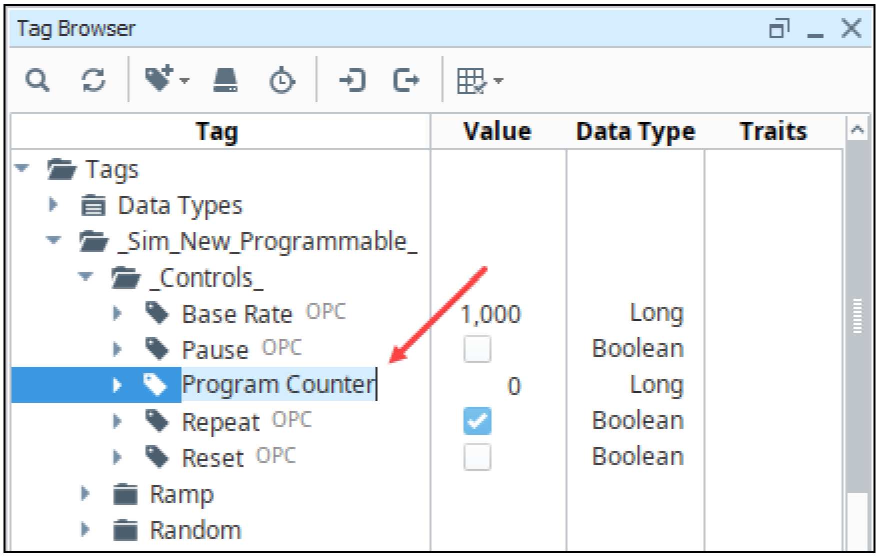
Tags names are flexible. For naming conventions, see Understanding Tags.

1. To rename an existing Tag, right click on the Tag in the Tag Browser and select the Rename option.



Tag	Value	Data Type	Traits
Tags			
Data Types			
_Sim_New_Programmable_			
Controls			
Base Rate OPC	1,000	Long	
Pause OPC	□	Boolean	
Program Counter OPC	0	Long	
Repeat OPC			
Reset OPC			
Ramp			
Random			
ReadOnly			
Realistic			
Sine			
Writeable			
Machine On Memory			
Poll Time Memory			
ScaledRef Reference			

2. The cursor will now blink inside the Tag name and you can type the new name.

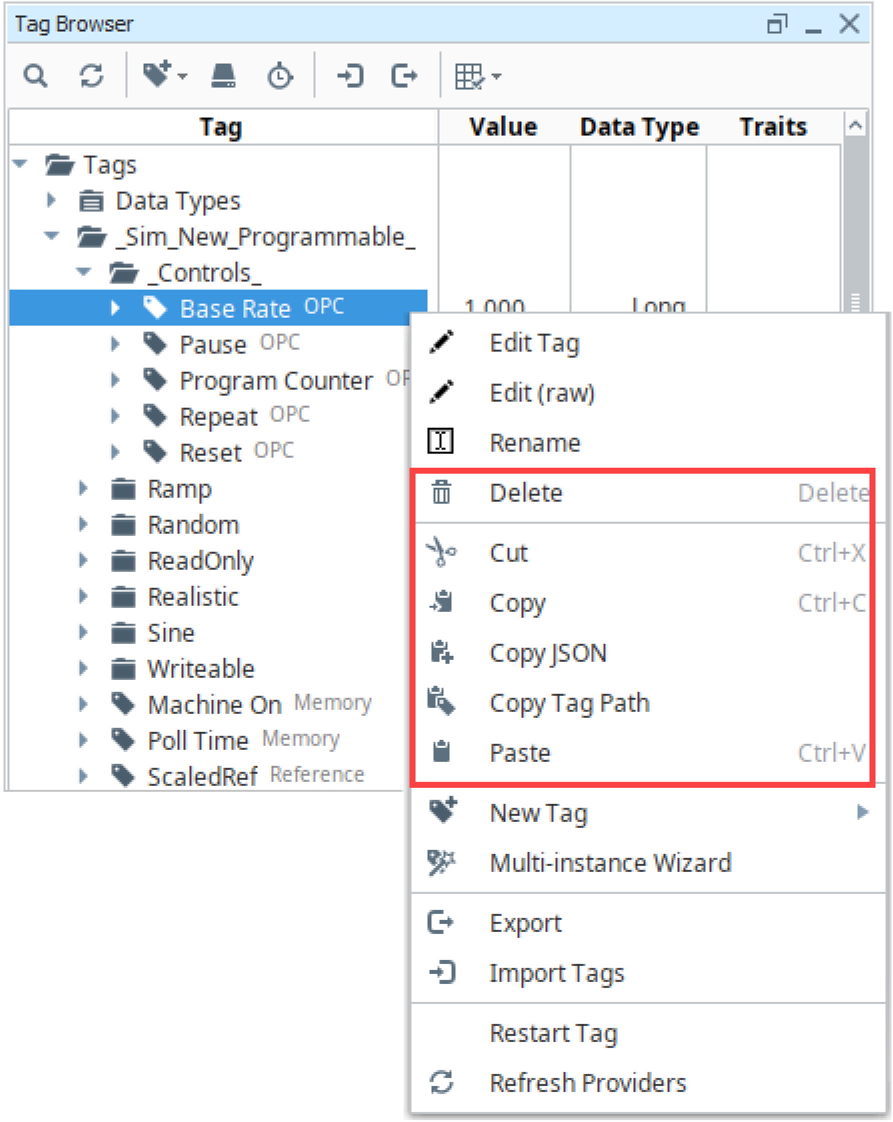


Tag	Value	Data Type	Traits
Tags			
Data Types			
_Sim_New_Programmable_			
Controls			
Base Rate OPC	1,000	Long	
Pause OPC	☐	Boolean	
Program Counter	0	Long	
Repeat OPC	☑	Boolean	
Reset OPC	☐	Boolean	
Ramp			
Random			

Cut, Paste, or Copy a Tag

You can also cut, paste, and copy tags within the Tag Browser. Right click on the Tag in the Tag Browser. Choose the command you want.

- **Delete:** Completely removes the Tag.
- **Cut**: Delete the Tag from the current location, but leave it in the clipboard to be pasted elsewhere in the browser.
- **Copy**: Make a copy of the Tag and leaves it in the clipboard to be pasted elsewhere in the browser.
- **Paste**: Pastes the Tag you've cut or copied into the currently selected location in the Tag Browser.



Addressing Bits

INDUCTIVE UNIVERSITY

Addressing Bits

Watch the Video

In order to address individual bits in Ignition, you must create a separate OPC Tag pointing directly to the specific bit in the PLC.

When the integer values that come from the OPC Tags are a series of binary bits, it is then possible to address each bit. For example, an integer value can have a 16-bit binary representation as shown here:

Integer	Bit level representation	How it works
4096	0,0,0,0,0,0,0,0,0,0,0,1,0,0,0	2^{12} = 4096
1025	1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0	2^{0} + 2^{10} = 1025

Address an Individual Bit

In this example, a MicroLogix PLC is connected to a Gateway. To address an individual bit, do the following:

1. From the **Tag Editor** window, create an OPC Tag to have:
   Data Type: **Integer** with a value of 1025

OPC Item Path: **`[MLX]B3:0`**

2. Then create a new OPC Tag with a Boolean value to the first bit of this Tag as follows:
   Data Type: **Boolean**
   OPC Item Path: **`[MLX]B3:0/0`** `(for Micrologic, you can specify the bit as: 0/0 or 0.0. That is, with a slash/ or a period.) [MLX]B3:0/0` has a value of "1" or a Boolean value of "True" because the integer value is odd.

3. You can create a tag for any of the other individual bits. For example, create a new OPC Tag with a Boolean value to the second bit of the original Tag as follows:

   Data Type: Boolean
   OPC Item Path: **`[MLX]B3:0/1`**
   [MLX]B3:0/1 has a value of "0" or a Boolean value of "False".

⚠ Addressing bits may work differently depending on the type of device you are addressing. Most commonly you will either use /<bit> like /0 or /1, or [<bit>] like [0] or [1].

User Defined Types - UDTs

What is a UDT?

On this page

...

UDTs (User Defined Types) are extremely important in Ignition. UDTs, also referred to as Complex Tags, offer the ability to leverage object-oriented data design principles in Ignition. Using UDTs, you can dramatically reduce the amount of work necessary to create robust systems by essentially creating parameterized "data templates".

By defining UDTs and using these essentially “data templates”, you can generate Tag instances to rapidly build complex screens. A change to the type definition is then inherited by all instances, drastically saving time when making routine changes.

The UDT data types are fully supported by Vision Templates, which means you can configure templates for your custom data types and take advantage of drag-and-drop binding to rapidly build complex screens.

Primary UDT Features

Object Oriented
Use small or large groups of Tags to create a single object. Create objects that match your real world devices or the existing structures in your PLCs.

Central Definition
Once you define your data type, you can then create instances of it. If at a later time you want to change some aspect of the type, you can simply edit the type definition, and all instances of it are automatically updated.

Parameterized Settings
Define custom parameters on your data type, and then reference them inside some or all of your member Tags. When it comes time to create instances, you can simply modify their parameter values in order to change where the underlying data comes from.

Extendable
Data types can inherit from other data types in order to add additional members or override settings. Instances can also override settings, allowing for flexibility when dealing with irregularities and corner cases.

INDUCTIVE UNIVERSITY

Understanding UDTs

Watch the Video

UDT Terminology

Many terms are frequently used when discussing complex Tags:

UDT
User Defined Type - the definition of the data type is its structure, Tags, attributes, and settings.

Instances
Instances are running copies of a data type with specific data for all members. Instances are linked to their parent types and are reloaded when their parent type changes. Besides specifically overridden settings, all settings are inherited from the type definition.

Parameters
Parameters are custom properties on data types that you can use inside the type or instance definition to create parameterized data templates. For example, if a data type consists of three OPC Tags that only differ by a number in the path, you can use a parameter for the "base address", allowing instances to be created with only one setting.

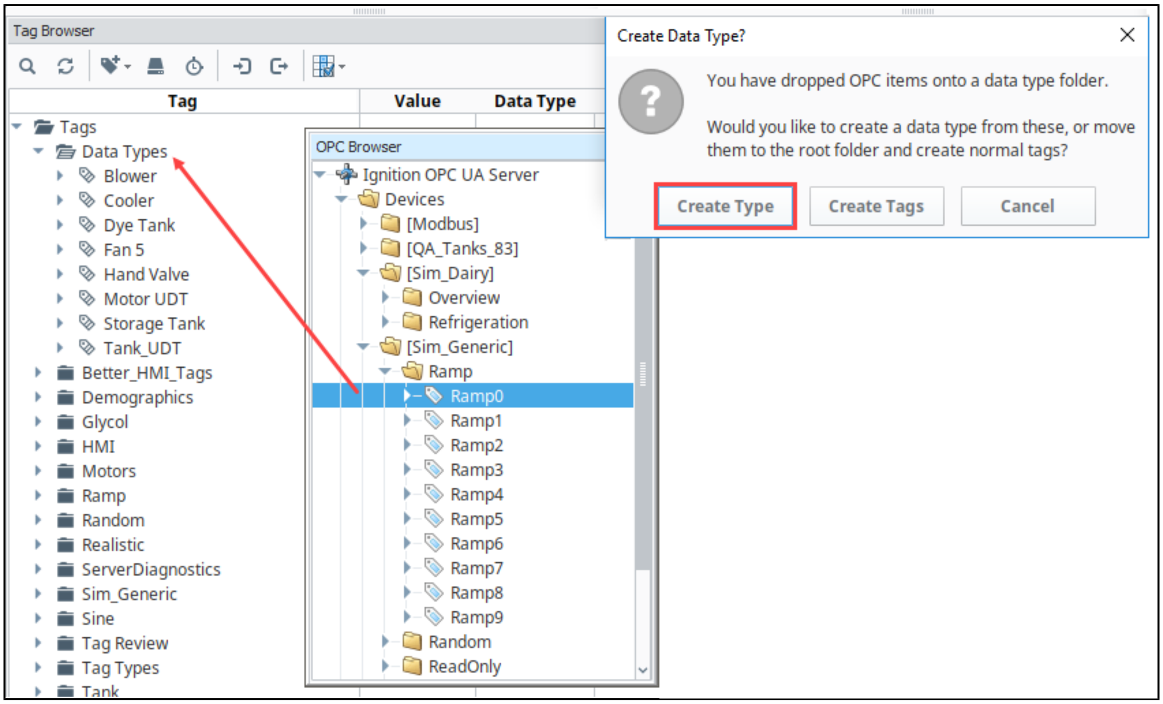
Members
Members are the Tags inside of a data type or instance.

Defining UDTs

You can save time and quickly generate data types from existing structures. This is particularly useful, for example, when data types are already defined in the PLC. Creating a new data type is very similar to creating standard Tags. You can open the Tag Browser and drag multiple Tags or folders into the Data Types folder, or you can create them manually by right-clicking and selecting New Tag.

Creating from OPC

If you have data types defined in your PLC (or OPC server), you can take a short-cut to the conversion step outlined above by dragging the Tags (or the folder for the type) from the **OPC Browser** directly onto the **Data Types** folder, and selecting **Create Type** from the subsequent dialog.



Create Data Type?

You have dropped OPC items onto a data type folder.

Would you like to create a data type from these, or move them to the root folder and create normal tags?

Creating Instances of UDTs

Creating instances of complex types is virtually identical to creating other types of Tags using the New Tag menu. Unlike standard Tags, it is likely that you'll have to modify attribute values or override certain member properties in order to make the instance unique. The process for doing this is the same to that used when creating data types. Once created, instances run very much like standard Tags. If the parent data type is updated, the instance will automatically receive the updates and refresh.

INDUCTIVE UNIVERSITY

Creating UDT Definitions

Watch the Video

Folders for UDT Instances

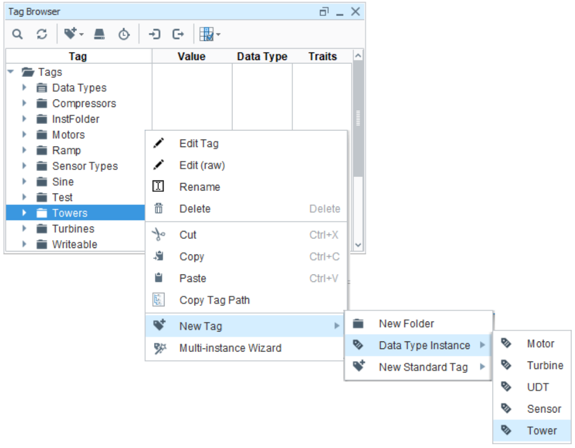
It's always a good idea to create a folders to keep your Tags organized. So you may want to create a folder for your UDT instances whether you create UDT instances manually or use the Multi-Instance Wizard.

Creating Instances Manually

INDUCTIVE UNIVERSITY

Manually Creating UDT Instances

You create individual instances of a UDT in exactly the same way as a basic Tag. Right-click and select **New Tag**, then the type of UDT you want to create.



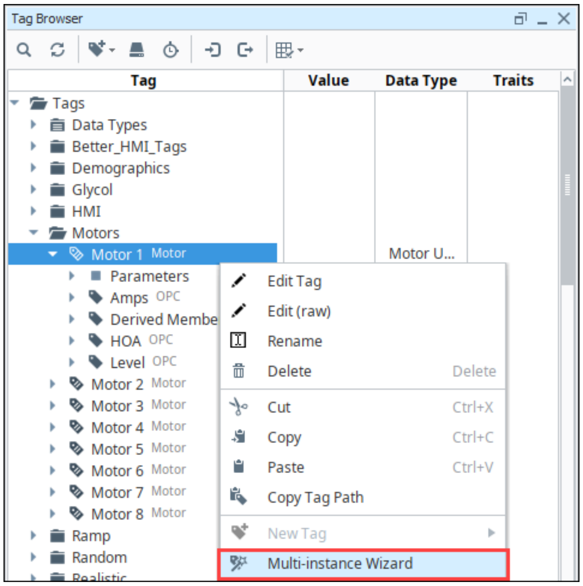
Watch the Video

UDT Multi-Instance Wizard

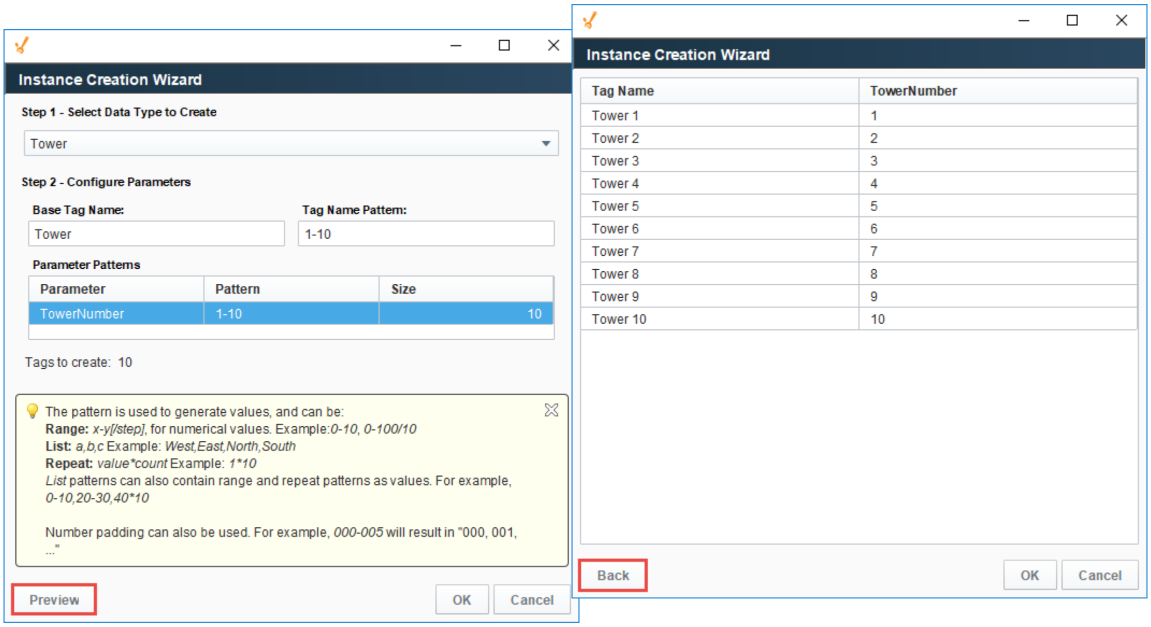
Watch the Video

Using the Multi-Instance Wizard

The Multi-Instance Wizard provides a powerful but simple mechanism for rapidly generating many instances of a data type, by specifying patterns for parameters. To get started, simply right-click a Tag in the **Tag Browser**, then select **Multi-instance Wizard** from the menu.



The **Instance Creation Wizard** window will be displayed. Once you select a data type to create from the dropdown list, you will see the parameters populate in the table, where you can specify values for them. The size of the parameter patterns dictate how many Tags get created, or if only single values are specified, you can choose to create multiple copies of the same configuration. You can even Preview the pattern to verify your values.



Instance Creation Wizard

Step 1 - Select Data Type to Create

Tower

Step 2 - Configure Parameters

Base Tag Name: Tower

Tag Name Pattern: 1-10

Parameter Patterns

Parameter	Pattern	Size
TowerNumber	1-10	10

Tags to create: 10

The pattern is used to generate values, and can be:
Range: *x-y[/step]*, for numerical values. Example:*0-10, 0-100/10*
List: *a,b,c* Example: *West,East,North,South*
Repeat: *value*count* Example: *1*10*
List patterns can also contain range and repeat patterns as values. For example, *0-10,20-30,40*10*

Number padding can also be used. For example, *000-005* will result in "000, 001, ..."

Instance Creation Wizard

Tag Name	TowerNumber
Tower 1	1
Tower 2	2
Tower 3	3
Tower 4	4
Tower 5	5
Tower 6	6
Tower 7	7
Tower 8	8
Tower 9	9
Tower 10	10

Related Topics ...

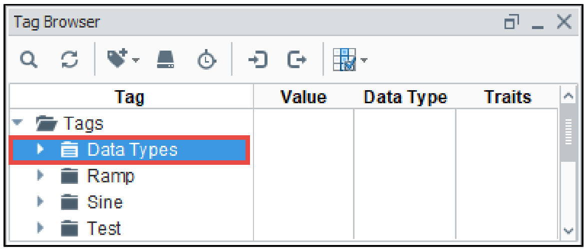
- Creating Tags
- System Tags

In This Section ...

UDT Definitions

So far, we have only dealt with simple Tags. In Ignition, you can also create Complex Tags or User Defined Types (UDTs). These types can model UDTs in certain PLCs (such as a ControlLogix) or can be completely new.

You create the UDTs in a special folder in the **Tag Browser** called **Data Types**.



UDTs (User Defined Types) can be created in the following different ways:

- Browsing OPC servers via OPC Browser
- Creating UDTs from existing Tags
- Creating UDTs manually

If your PLC supports UDTs, the easiest way to create a user defined type is from OPC.

The Data Type Folder

When creating a new data type, the new definition is **always** placed in the Data Types folder, regardless of which folder was selected before creating the new definition.

Additionally, the Data Types Folder in the Tag Browser can only ever contain UDT definitions, and UDT definitions can only ever exist in the Data Types folder.

On this page ...

INDUCTIVE UNIVERSITY

Creating UDT Definitions

Watch the Video

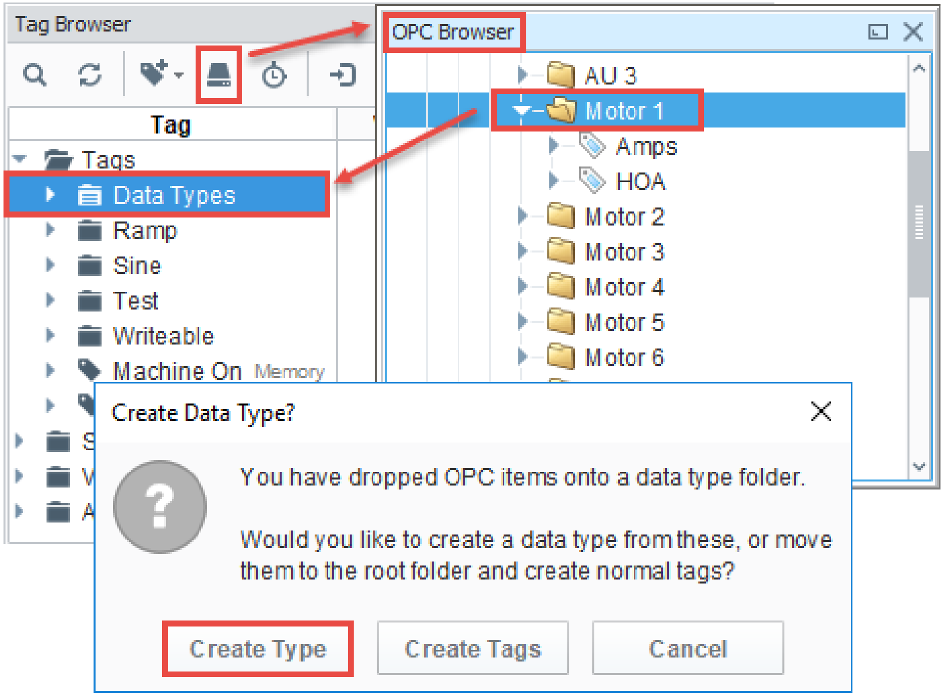
Creating the UDT from OPC Browser

In the following example, we used a Dairy Simulator which supports UDTs. We will use the second method from above, that is, creating the UDT from OPC. In our Dairy PLC, let's say we already have a motor UDT setup, and now we will create the UDT in Ignition.

1. From **Tag Browser**, click the **OPC Browse** icon.

2. Under the **Dairy** folder, go to the **Overview** folder and find the **Motor 1** folder.
   Motor 1 has two Tags: AMPS and HOA.

3. Drag the **Motor 1** folder from the OPC Browser to the **Data Types** folder in **Tag Browser**.
   A window prompts you asking if you want to Create Type or Create Tags. Click on **Create Type**.

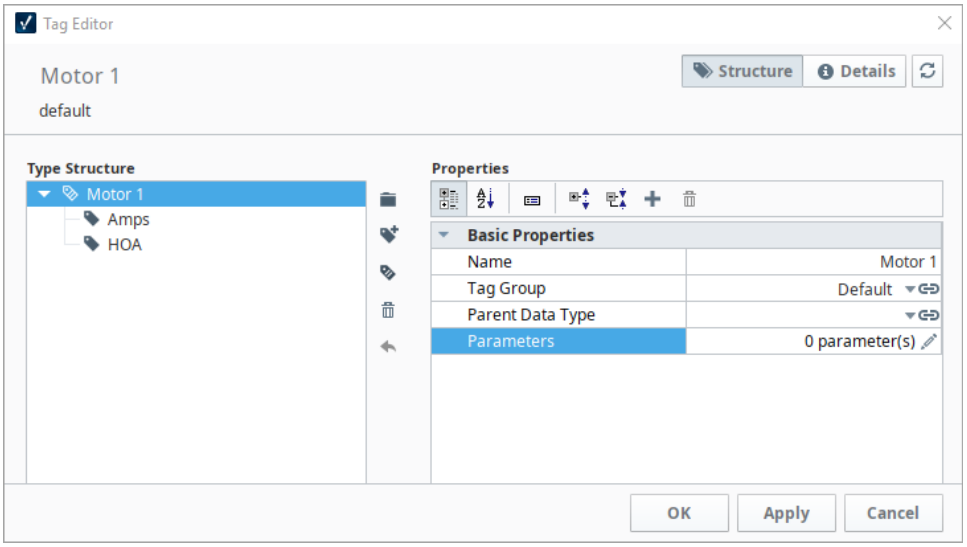


You have dropped OPC items onto a data type folder.

Would you like to create a data type from these, or move them to the root folder and create normal tags?

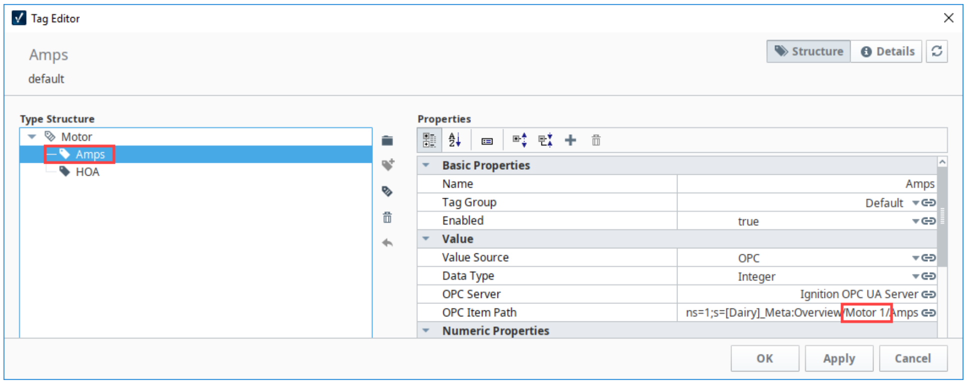
4. The **Tag Editor** window will open showing you the Tag structure of your UDT. Notice the two Tags; Amps and HOA are automatically part of the UDT structure.

> **Tag Editor Structure and Details Icons**
>
> There are two icons on the top right side of the Tag Editor. The **Tag** icon displays the Tag structure of the UDT. The **Inf ormation** icon contains an area to add notes and check diagnostics. You can toggle these icons to switch back and forth between the Tag Structure and the Details.



Basic Properties	
Name	Motor 1
Tag Group	Default
Parent Data Type	
Parameters	0 parameter(s)

5. In the **Tag Editor**, change the Name from **Motor 1** to **Motor** to make the type name more of a generic name.
6. Right now, each Tag is pointing to a specific address in the PLC. Select the **Amps** Tag to see that it is pointing to **Motor 1** in the PLC.



Basic Properties	
Name	Amps
Tag Group	Default
Enabled	true
Value	
Value Source	OPC
Data Type	Integer
OPC Server	Ignition OPC UA Server
OPC Item Path	ns=1;s=[Dairy]_Meta:Overview/Motor 1/Amps
Numeric Properties	

Because we are creating a UDT, we don't want to point to one specific set of Tags. We want each instance of the UDT to reference a different set of Tags. To do that, we need to add a parameter to the UDT. Parameters are custom properties on data types. You can use the parameters inside the type or instance definition to create parameterized data templates.

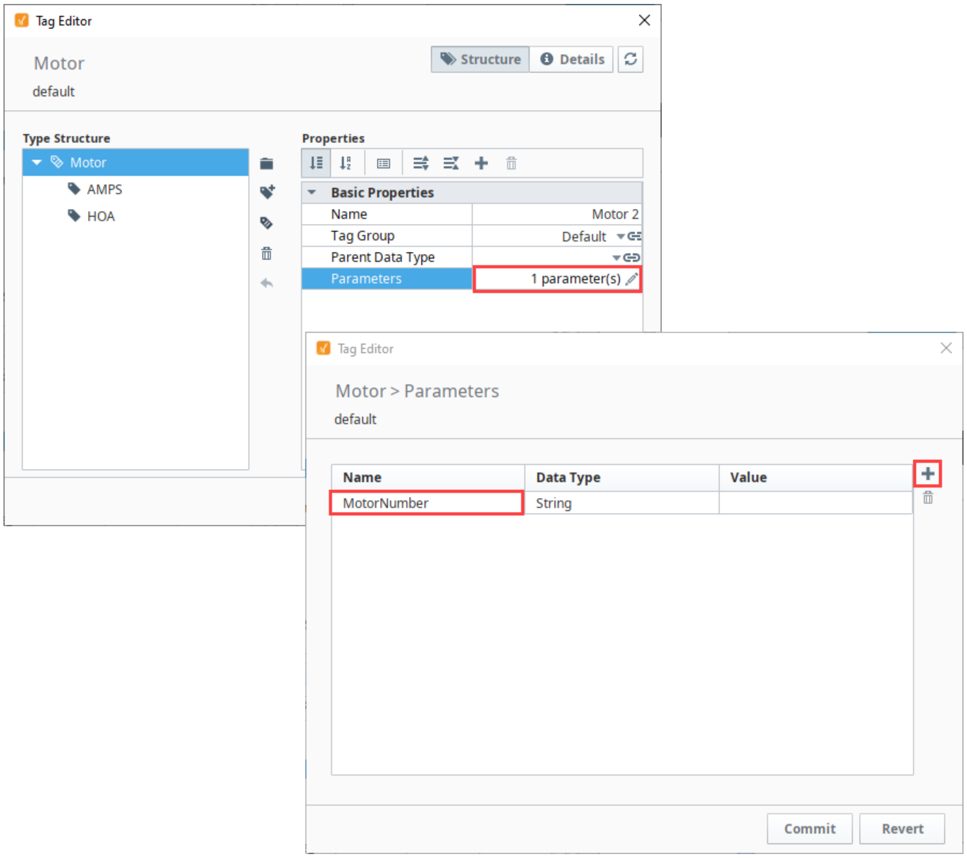
For example, if a data type consists of two OPC Tags that only differ by a number in the path, you can use a parameter for the "base address," allowing instances to be created with only one setting.

7. In the **Tag Editor**, with the **Motor** data type selected, click the **Edit** icon next to Parameters property to add a new parameter.
8. Click the **Add** icon, and enter the name for your new parameter (i.e., 'MotorNumber').

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

As of release 8.0.15, you can also specify a data type for the parameter.

9. Click **Commit** to save the parameter. To cancel the entry and go back to the previous window, click **Revert.**



Tag Editor

Motor
default

Type Structure
- Motor
  - AMPS
  - HOA

Properties

Basic Properties

Name	Motor 2
Tag Group	Default
Parent Data Type	
Parameters	1 parameter(s)

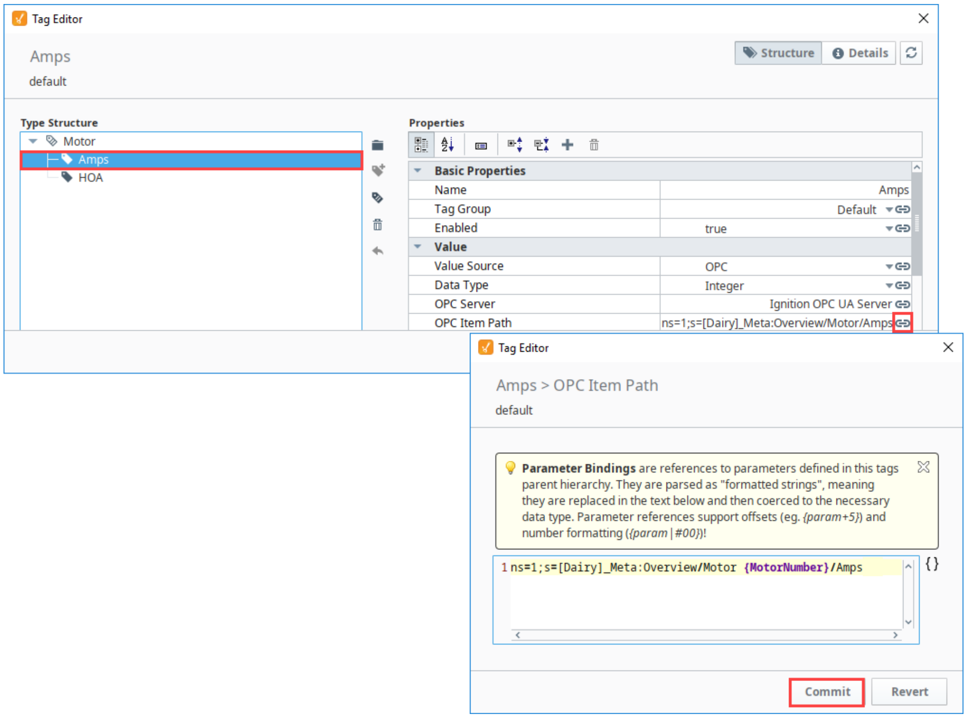
Tag Editor

Motor > Parameters
default

Name	Data Type	Value
MotorNumber	String	

10. For each Tag, we can substitute the Motor number with the **'MotorNumber'** parameter we just created. With the **Tag Editor** still open , select the **Amps** Tag.

   a. In the **OPC Item Path** field, click the **binding** icon, select **Edit**, and the **Amps > OPC Item Path** window will open.
   b. Place your cursor at the end of **'Motor1',** delete the **'1',** add a space, and enter **'{MotorNumber}'**. Don't forget the curly braces. In the OPC Item Path it will look like the following, then click **Commit** to save your updates and go back to the previous window.



Tag Editor

Amps
default

Type Structure
- Motor
  - Amps
  - HOA

Properties

Basic Properties	
Name	Amps
Tag Group	Default
Enabled	true
Value	
Value Source	OPC
Data Type	Integer
OPC Server	Ignition OPC UA Server
OPC Item Path	ns=1;s=[Dairy]_Meta:Overview/Motor/Amps

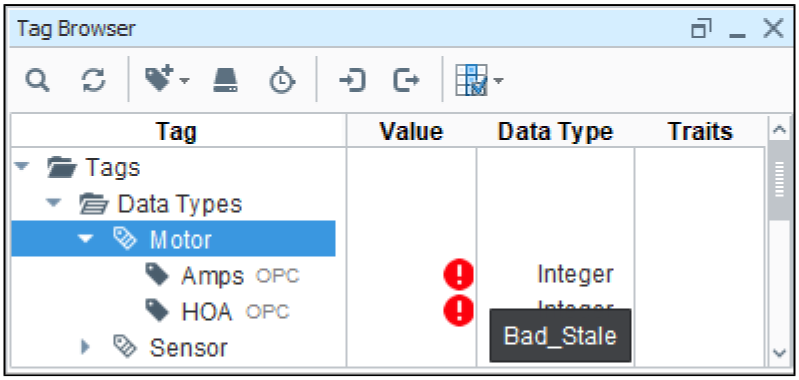
Tag Editor

Amps > OPC Item Path
default

Parameter Bindings are references to parameters defined in this tags parent hierarchy. They are parsed as "formatted strings", meaning they are replaced in the text below and then coerced to the necessary data type. Parameter references support offsets (eg. *{param+5}*) and number formatting (*{param|#00}*)!

```
ns=1;s=[Dairy]_Meta:Overview/Motor {MotorNumber}/Amps
```

11. Repeat Step 8 for the **HOA** Tag. The **OPC Item Path** will then show the **'MotorNumber'** parameter for each of the Tags.
12. Click **Apply**.
13. Click **OK** to save the UDT. You will then see the **Motor** UDT in the **Data Types** folder. If you expand the Motor UDT, you will also see the Amps and HOA Tags. Don't panic if you see the exclamation point ❗ next to any Tags. This is normal! The Tags are defined, but they don't have any values which is the reason for the Bad_Stale warning. They are not suppose to have values. The UDT instances you create from the UDTs will contain the values.



Creating a New Data Type Manually

When creating a new data type manually, you have to specify the name of the data type, parameters, and members or Tags that are going to be part of the structure of your new data type.

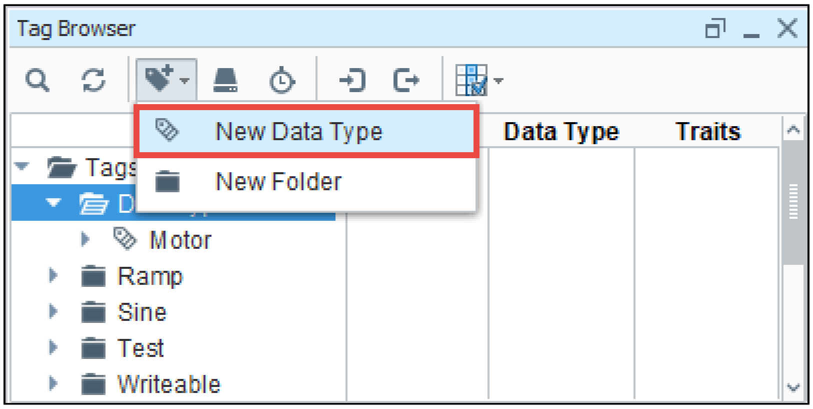
INDUCTIVE UNIVERSITY

Manually Creating UDT Instances

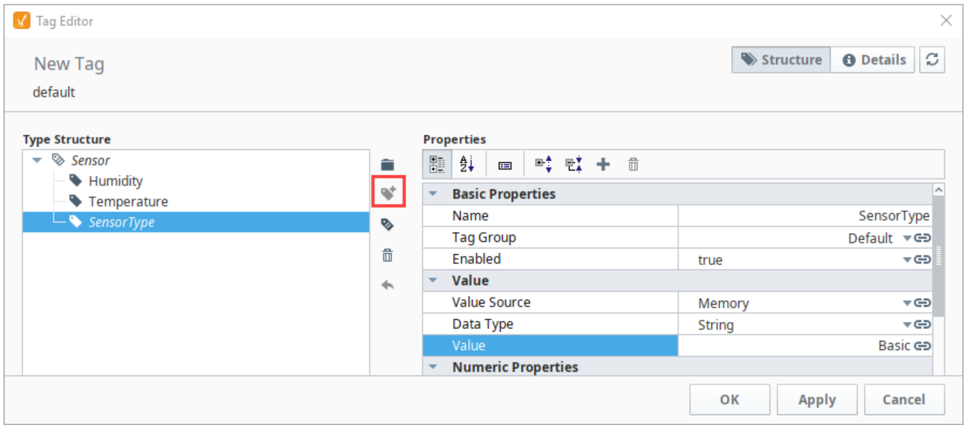
Watch the Video

1. Select the **Tags** folder in the **Tag Browser**.

2. Click the **Tag** icon on the **Tag Browser** toolbar, and select Select **New Data Type**.



3. When editing complex Tags, the **Tag Editor** window appears a bit differently then the Tag Editor for a standard Tag. The member tree structure is presented on the left. By selecting a member (i.e., Sensor) and clicking the Tag icon, you can add different Tag types, and the properties for the selected Tag appears on the right.



- Sensor
  - Humidity
  - Temperature
  - SensorType

Basic Properties	
Name	SensorType
Tag Group	Default
Enabled	true
Value	
Value Source	Memory
Data Type	String
Value	Basic
Numeric Properties	

UDT Features

There are many features of UDTs that allow more control over the definition and instances

Extending Other Types

User Defined Types can extend other UDTs to add additional members, or override default values. The **Parent Type** property (with the main /top UDT member selected) can be modified by selecting another UDT from the dropdown. This will add all members of the selected UDT to this UDT, and you can add additional Tags or change default properties.

> ⚠ The Parent Type can only be selected when the Tag is first created. After that, it is not possible to modify the Parent Type property.

Adding Members to a Data Type

Simply click the **Tag** icon on right side of the Type Structure. Data types can contain standard Tags like OPC and DB Tags, as well as folders and instances of other complex types.

Adding Parameters

Parameters, which can be used for property expansion in member Tags, can be added by selecting the data type in the member tree. If a data type contains other complex types in it, there may be various points in the tree with custom parameters. While a data type can override the parameter values inherited from a parent, new parameters can only be added to the root node of the new data type.

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

As of release 8.0.15, UDT parameters can be saved as specific data types.

Configuring Member Properties

The Tags inside of data types are configured much like normal Tags. However, in this case, the values can be thought of more as "default values", which will be used unless other values are specified when the instance is created. Most of the values configured in the data type can be modified later in subtypes or instances. Furthermore, unlike normal Tags, in the context of a data type many properties (generaly the string based properties) can reference the custom attributes of the type in order to build parameterized Tags.

Attribute Referencing and Parameterized Types

As mentioned above, many properties in the member Tag configuration can reference the parameters available in the data type. When instances are created, these references are replaced with the values defined for the type. Parameter references also support basic offsets and numerical formatting, providing a great deal of flexibility. To reference a parameter, use the syntax `{ParameterName}.`

To offset a value, use the form `{ParameterName+offset}`.
To format a value, use the form `{ParameterName|format}`. The format pattern is the same as that used for the `numberFormat` expression function. In short, "0" can be used to require a digit, and "#" can be used for optional digits. ie: **##0**

Example:

For this example, we'll assume that we're parameterizing the **OPC Item Path**, and that the data type has an integer attribute named `BaseAddress` defined. We'll pretend the **OPC Server** provides Tags named like `DataPoint1`.

Standard referencing

OPC Item Path: `DataPoint{BaseAddress}`

Offset

Imagine that our data type had three fields, and these were laid out sequentially in the device.
Instead of specifying each address for each Tag, we can simply offset from the base address:

Member 1: `DataPoint{BaseAddress+0}`
Member 2: `DataPoint{BaseAddress+1}`
Member 3: `DataPoint{BaseAddress+2}`

Formatting (with offset)

Continuing from the example above, imagine that our OPC server actually provided addresses in the form `DataPoint001`, in order to stay consistent up to "DataPoint999". This can be accommodated
using number formatting in the reference:

Member 1: `DataPoint{BaseAddress+0|000}`
Member 2: `DataPoint{BaseAddress+1|000}`
Member 3: `DataPoint{BaseAddress+2|000}`

This format of three zeros means "three required digits". If our instance has a base address of 98, the resulting paths will be `DataPoint098, DataPoint099, DataPoint100`.

The following feature is new in Ignition version **8.0.3**
Click here to check out the other new features

Parameters support more mathematical operators in addition to offsets and formatting. There is a simple expression language available that can be used in conjunction with formatting. The following table shows all available operators in their order of operations (they are evaluated starting at the top of the table).

Operator	Description	Example
()	Parenthesis. These operators are used for grouping any number of values. Also used to change the order of operations.	{Baseaddress*(2+3)}
^	Power. This operator is used to raise a number to a power.	{BaseAddress^2}
-	Negative. Used to create a negative value from one number.	{BaseAddress*-2}
*	Multiplication. Multiply two numbers.	{BaseAddress*2}
/	Division. Dividing the first number by the second number.	{BaseAddress/2}
%	Modulus. This operator returns the remainder of a division operation. IE: 7/3 = 2 with a remainder of 1, so 7%3 = 1	{BaseAddress%2}
+	Addition. Add two numbers.	{BaseAddress+2}
-	Subtraction. Subtract two numbers	{BaseAddress-2}

Example

```
# This dynamic OPC Item path takes in three parameters to determine the tag path
ns=1;s=[DeviceName]Path/to/tag{BaseAddress+(ParamNum*Multiplier)|0000}
```

```
# The OPC Item path resolves to the following assuming the following values:
# BaseAddress = 5
# ParamNum = 8
# Multiplier = 2
ns=1;s=[DeviceName]Path/to/tag0021
```

Calculations and numerical parameter names

If the parameter names are purely numerical values (we don't recommend this: it gets confusing), then quotation marks must encase the parameter to run any sort of calculations on the value of the parameter.

For example, if a UDT contains a parameter named 0, and its value is 10:

```
// This will evaluate to 0, because it thinks you mean the integer 0, not the parameter named "0"
{0 * 1000}

// This will evaluate to 10000, because the quotation marks denote a parameter named "0"
{"0" * 1000}
```

Overriding Properties

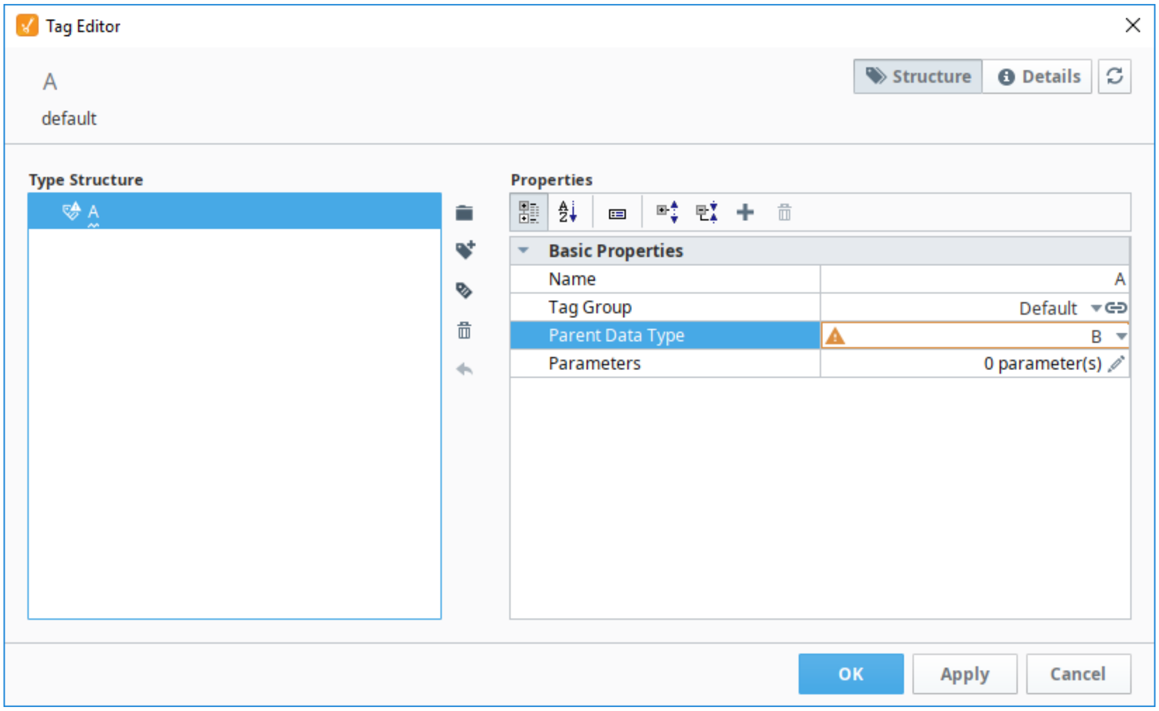
Subtypes and instances can override the properties defined in parent types. To do this, simply select the *override control,* the small grey ball (●) next to the property to override in the member editor. Conversely, to remove the override, simply unselect the control.

Custom parameters can be overridden as well, but it is not required to specify that the value is an override. Simply provide a new value for the property. For inherited parameters, the **delete** button next to the parameter table will simply remove the override. The parameter can only truly be delete from the type that defines it.

The following feature is new in Ignition version **8.0.5**
Click here to check out the other new features

UDT Error Indication

The Tag Editor will indicate if there is an error in the UDT. For example, if you configure circular inheritance (A inherits from B, and B inherits from A), then the Tag Editor will show a **UDT Error** ⚠ icon as well as squiggle line under the UDT name. In addition, the Parent Data Type property will show a warning icon in this state.

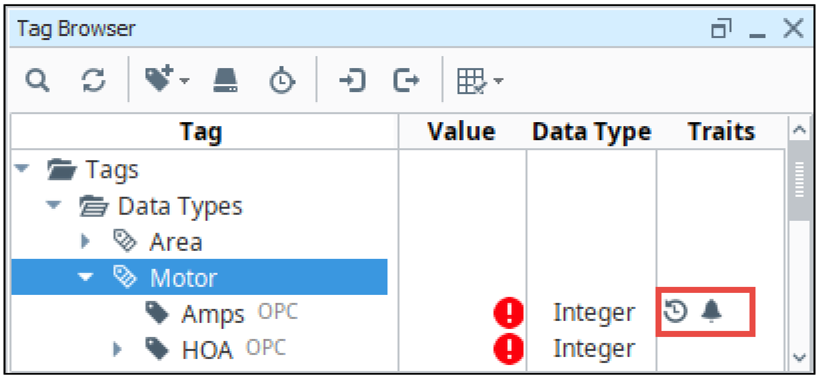


Property	Value
Name	A
Tag Group	Default
Parent Data Type	B
Parameters	0 parameter(s)

The following feature is new in Ignition version **8.0.3**
Click here to check out the other new features

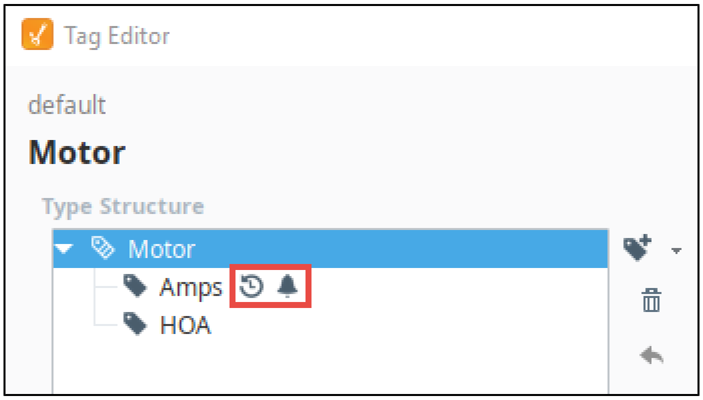
Trait Badges

Trait badges show the presence of various configurations such as Alarms and Tag History in the Traits column of the Data Types folder in the Tag Browser. Trait badges also appear in each instance of a UDT in the Traits column of the Tag Browser.



Tag	Value	Data Type	Traits
Tags			
Data Types			
Area			
Motor			
Amps OPC	!	Integer	
HOA OPC	!	Integer	

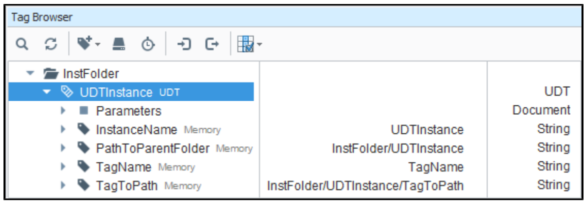
Trait badges are also displayed within a UDT in the Tag Editor.



Pre-Defined Parameters

UDTs have a few parameters already defined to make things easier for you. They give you access to the name and various paths associated with a UDT member Tag. These parameters can be accessed from anywhere in a Tag that a normal parameter can be used. Each of these parameters uses that Tag it is in as a starting point for its path.

Parameter Name	Description
{InstanceName}	The name of the UDT Instance that this Tag is inside.
{PathToParentFolder}	The full path to the folder that this Tag is in.
{TagName}	The name of the Tag that is using this parameter.
{PathToTag}	The full path to the Tag using this parameter.

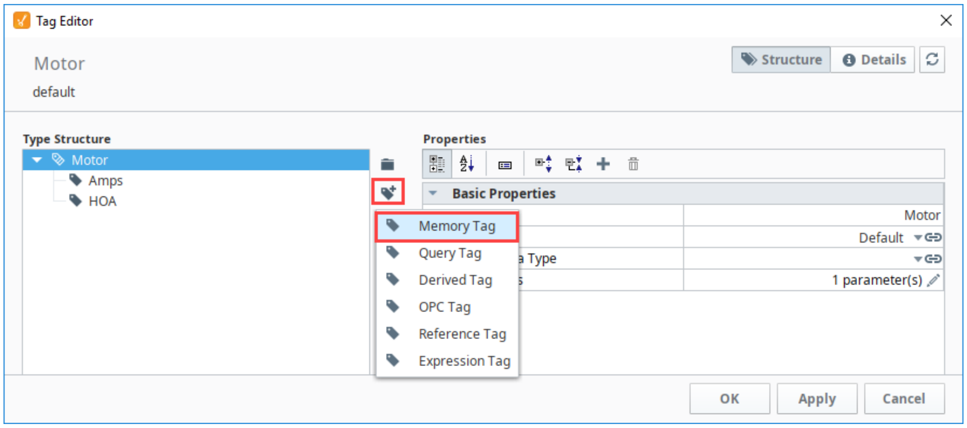


Tag	Value	Data Type
InstanceName Memory	UDTInstance	String
PathToParentFolder Memory	InstFolder/UDTInstance	String
TagName Memory	TagName	String
TagToPath Memory	InstFolder/UDTInstance/TagToPath	String

How to Edit an Existing UDT

If you make a change to a UDT, all your instances will automatically get updated because they are all of the same UDT type. You can make rapid work of making changes to instances using a UDT.

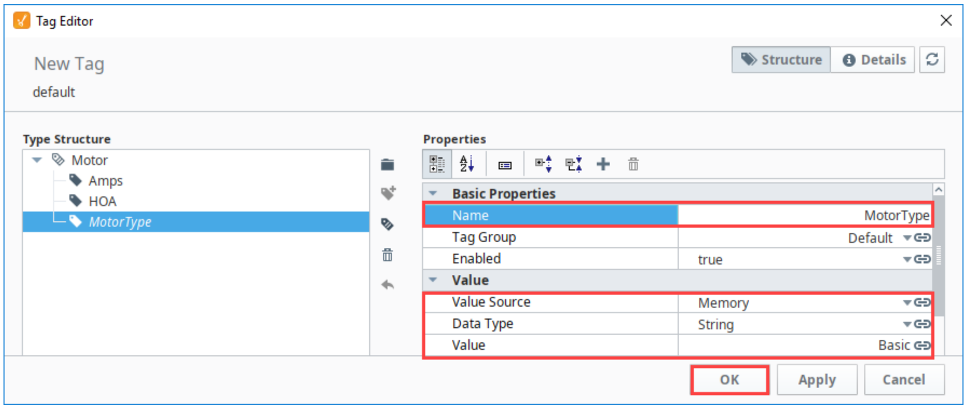
1. To make a change to your Motor UDT, go to the **Tag Browser** and double click on **Motor**. This example adds a new Memory Tag.
2. The **Tag Editor** will open. Click on the **Tag** icon on the top-right of the Type Structure, and select **New Memory Tag**.



3. While still in the Tag Editor, configure the following properties:

Name: **MotorType**
Data Type: **String**
Value: **Basic**

4. Click **OK**.



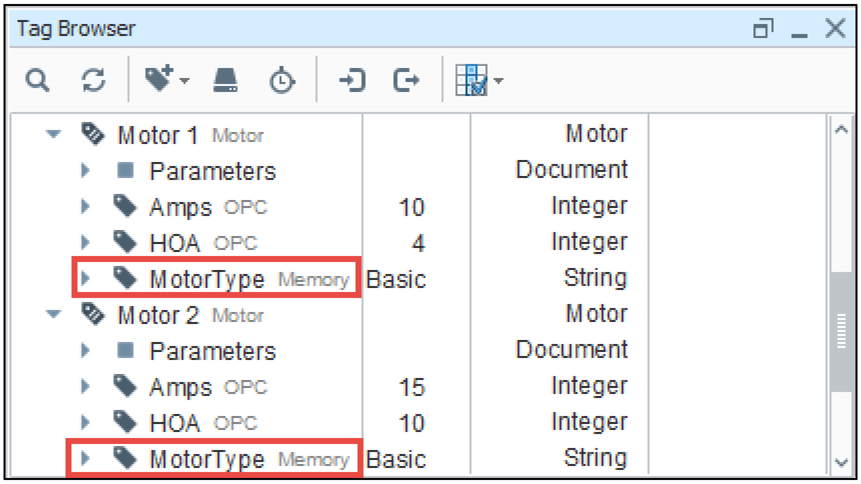
New Tag
default

Type Structure
- Motor
  - Amps
  - HOA
  - MotorType

Properties

Basic Properties	
Name	MotorType
Tag Group	Default
Enabled	true
Value	
Value Source	Memory
Data Type	String
Value	Basic

5. Now, each instance automatically gets the new Memory Tag. Expand each of your Motor Tag instances to see the new MotorType.

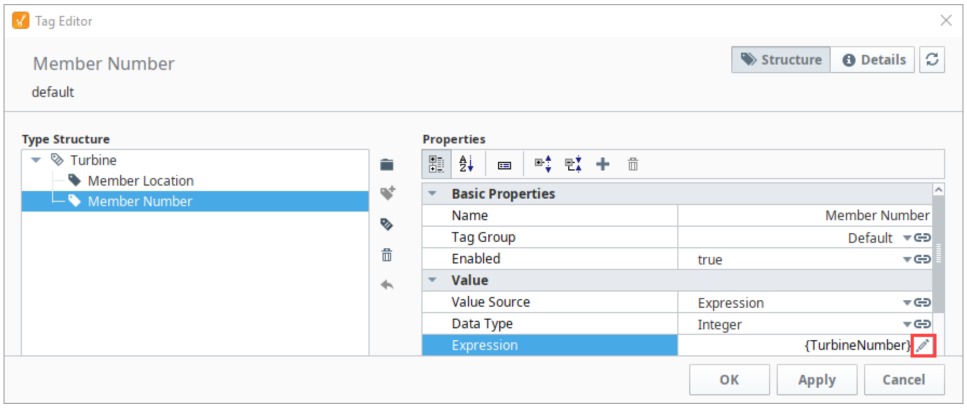


Motor 1 Motor		Motor
Parameters		Document
Amps OPC	10	Integer
HOA OPC	4	Integer
MotorType Memory	Basic	String
Motor 2 Motor		Motor
Parameters		Document
Amps OPC	15	Integer
HOA OPC	10	Integer
MotorType Memory	Basic	String

Data Type Parameters in Expressions

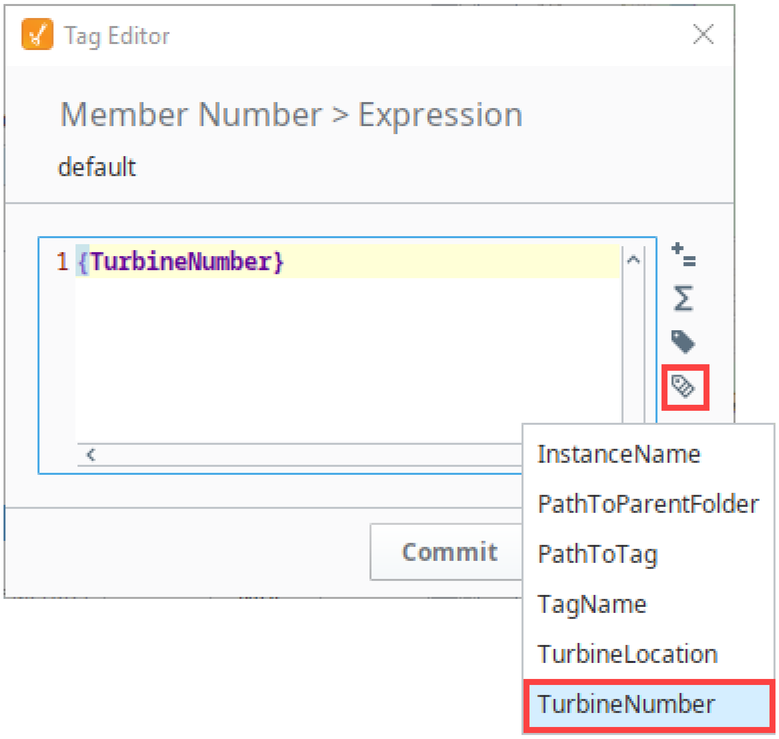
It is possible to use the value of data type parameters directly in expression bindings within a UDT. Parameter references can be quickly inserted into an expression.

1. Open the UDT, and select the **Edit** icon next to one of the members.



Name	Member Number
Tag Group	Default
Enabled	true
Value Source	Expression
Data Type	Integer
Expression	{TurbineNumber}

2. This opens the Expression window. Click on the **UDT Parameters** icon on the right of the expression area, select a parameter, then click **Commit**.



```
{TurbineNumber}
```

> **Using Parameters in UDTS**
>
> If you're familiar with using parameters in UDTs, you might want a quick refresher and look at the Quick Reference section at the end of this page.

Syntax

Quotation marks are not required when referencing string parameters. The syntax for data type parameter expressions does not differ from Tag and property references when using string values. If you use quotation marks referencing string values, the parameter values will not display correctly. It will display the parameter rather than the parameter's value.

If the data type of the parameter is not a string, then quotation marks are not required, but can be used.

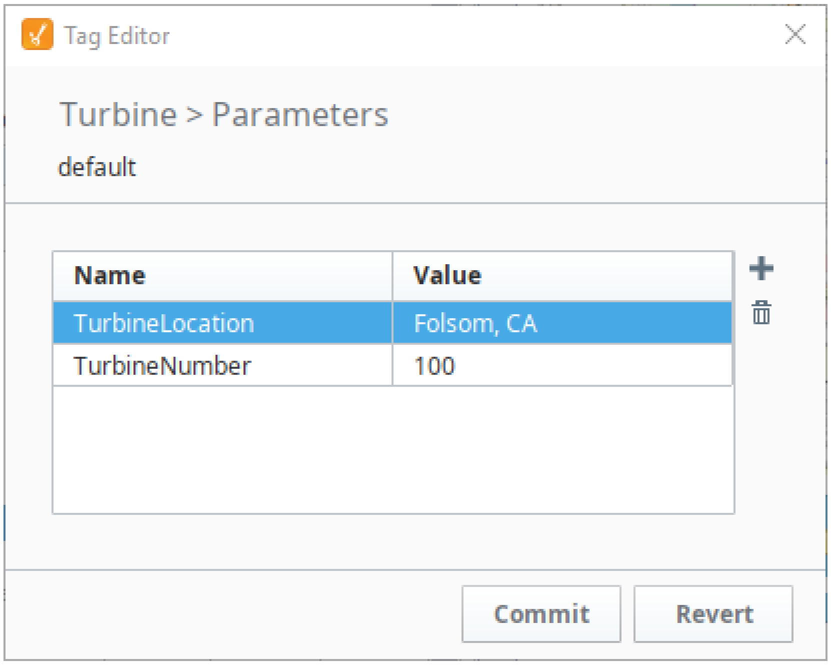
It does not matter if double quotation marks or single quotation marks (" versus ') are used as long as a matching closing quotation mark is present.

Syntax Code Examples

```
{string parameter} //This expression will evaluate successfully.
```

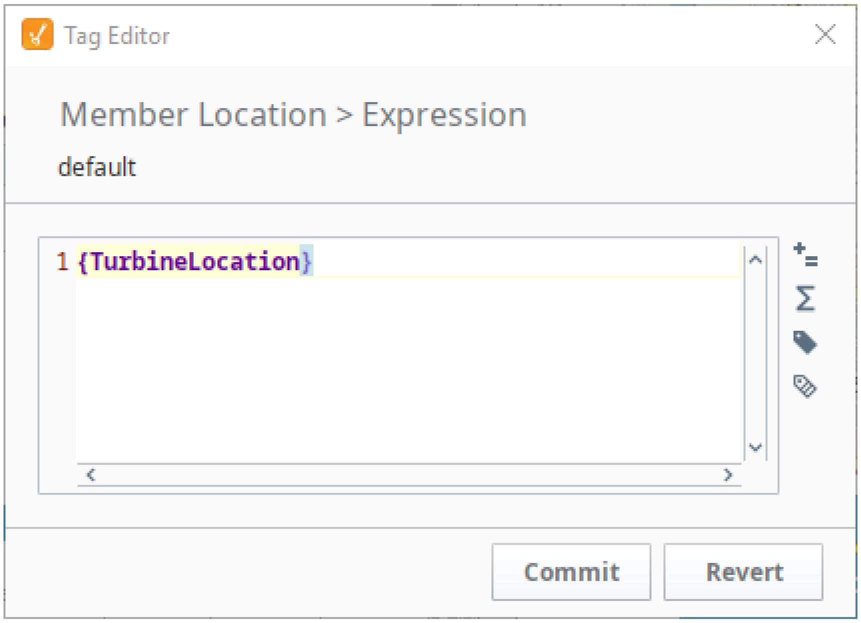
Examples

Below is an instance of a Turbine UDT. It contains a string parameter named **TurbineLocation**.



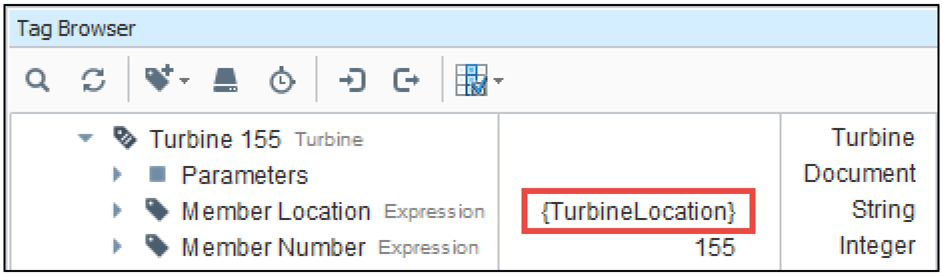
Name	Value
TurbineLocation	Folsom, CA
TurbineNumber	100

Inside the Turbine UDT is an expression Tag named **Member Location**. To show the value of the **TurbineLocation** parameter on Member Location, the following expression would be used on the UDT definition.



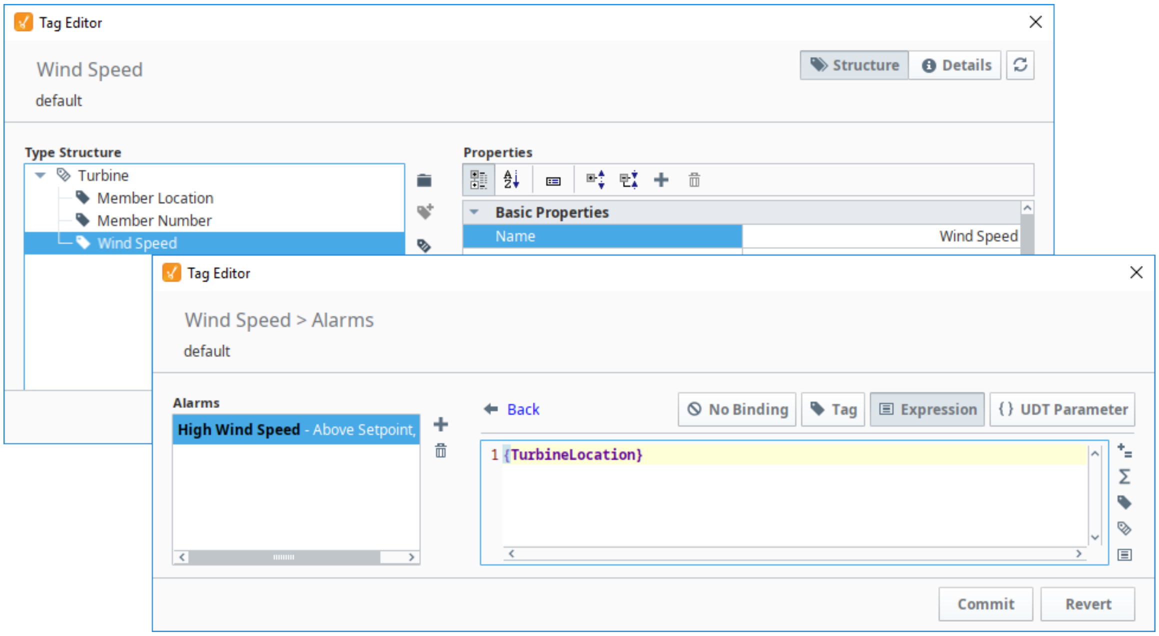
```
{TurbineLocation}
```

If you use quotation marks referencing string values, the parameter values will not display correctly. The Tag displays the parameter rather than the parameter's value.



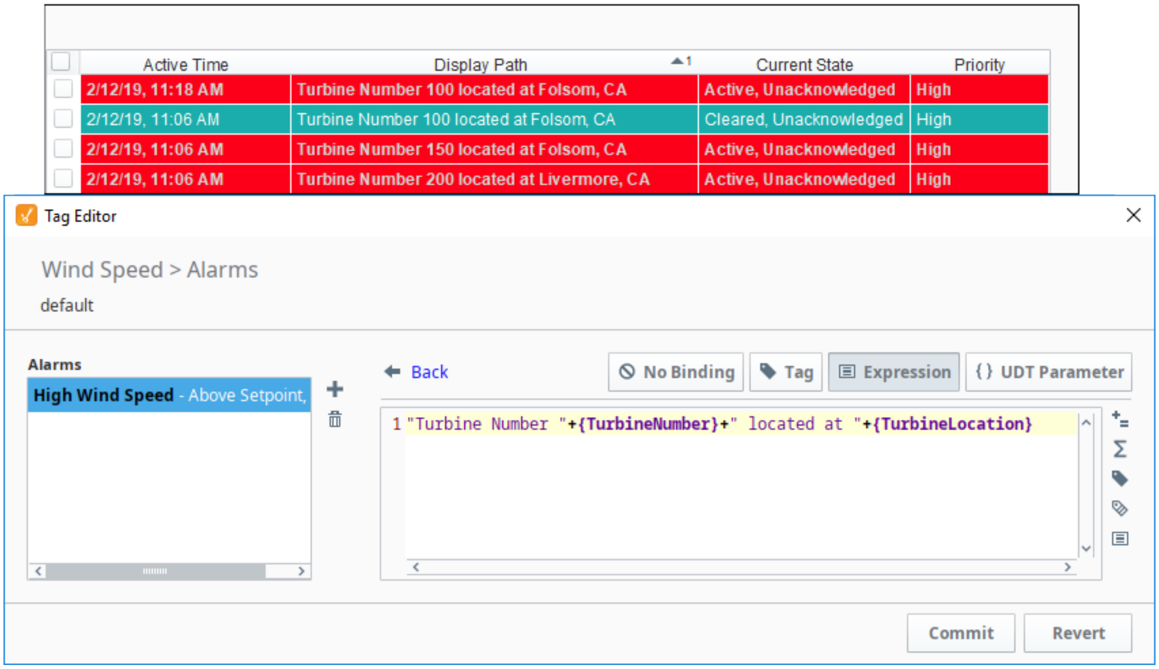
Turbine 155 Turbine		Turbine
Parameters		Document
Member Location Expression	{TurbineLocation}	String
Member Number Expression	155	Integer

The same syntax should be applied to parameters in bindings on alarm properties. Below is an example using the same UDT, but the expression is instead located on the Display Path property of an alarm.



```
{TurbineLocation}
```

Notice, that static values are included in quotation marks, and non-string parameters are outside of the the quotations marks.



□	Active Time	Display Path	Current State	Priority
□	2/12/19, 11:18 AM	Turbine Number 100 located at Folsom, CA	Active, Unacknowledged	High
□	2/12/19, 11:06 AM	Turbine Number 100 located at Folsom, CA	Cleared, Unacknowledged	High
□	2/12/19, 11:06 AM	Turbine Number 150 located at Folsom, CA	Active, Unacknowledged	High
□	2/12/19, 11:06 AM	Turbine Number 200 located at Livermore, CA	Active, Unacknowledged	High

Tag Editor

Wind Speed > Alarms

default

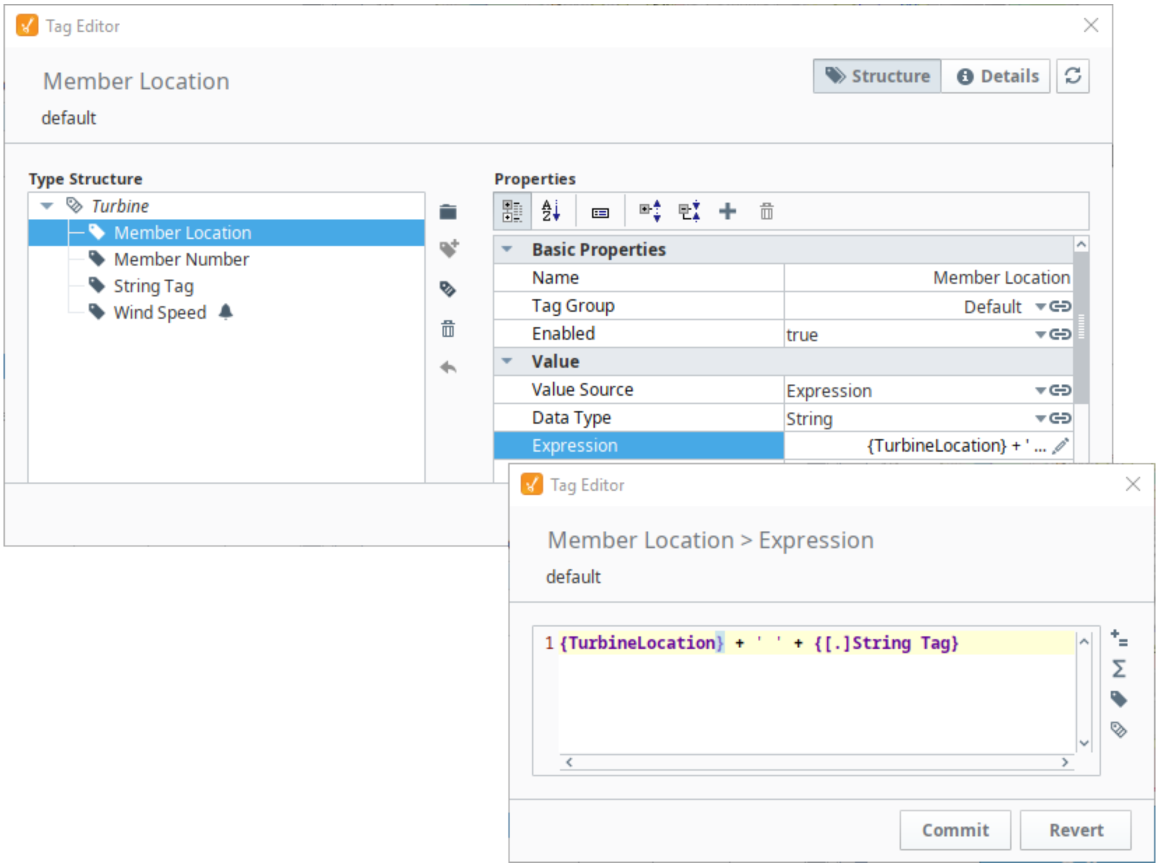
Alarms

High Wind Speed - Above Setpoint,

```
"Turbine Number "+{TurbineNumber}+" located at "+{TurbineLocation}
```

Combining Parameters and Tag References

Because parameter and Tag references differ in syntax, some consideration must be made when attempting to use both in the same expression. Tag references must not be placed inside of quotes. After adding a string Tag to the Turbine UDT, a reference to the Tag can be added to Member Location's expression. Single quotes were added to create a space between the Member's Location and the string value.



Tag Editor

Member Location

default

Type Structure

- Turbine
  - Member Location
  - Member Number
  - String Tag
  - Wind Speed

Properties

Basic Properties	
Name	Member Location
Tag Group	Default
Enabled	true
Value	
Value Source	Expression
Data Type	String
Expression	{TurbineLocation} + ' ...

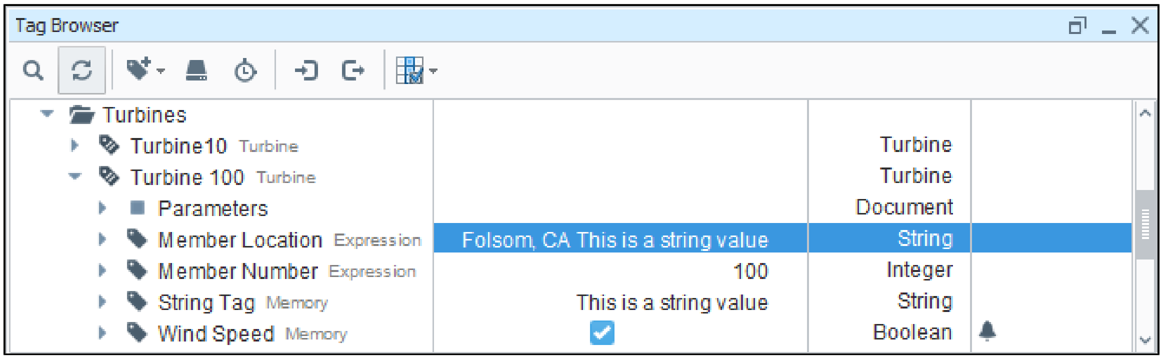
Tag Editor

Member Location > Expression

default

```
{TurbineLocation} + ' ' + {[.]String Tag}
```

Here's what it looks like in the Tag Browser.



Turbines			
Turbine10 Turbine		Turbine	
Turbine 100 Turbine		Turbine	
Parameters		Document	
Member Location Expression	Folsom, CA This is a string value	String	
Member Number Expression	100	Integer	
String Tag Memory	This is a string value	String	
Wind Speed Memory	✓	Boolean	🔔

Quick Reference

Reference Type	Require Quotes?	Expression Example
Static String	Requires Quotes	`"This is a string"`
String UDT Parameter	No Quotes	`{turbine_location}`
String Tag Reference	No Quotes	`{[.]String Tag}`
String UDT Parameter with Static String	Requires Partial Quotes	`"The turbine is located at " + {turbine_location}`
String UDT Parameter with String Tag Reference	Partial Quotes	`{turbine_location} + " " + {[.]String Tag}`
String Parameter, Static String, and String Tag Reference	Partial Quotes	`"The Turbine " + {[.]String Tag} + " " + {turbine_location}`

Related Topics ...

- UDT Instances
- UDT Nesting
- Expression Overview and Syntax

UDT Instances

Creating instances of UDTs is virtually identical to creating other types of Tags using the **New Tag** menu. Unlike standard Tags, it is likely that you'll have to modify attribute values or override certain member properties in order to make the instance unique.

On this page ...

Create a UDT Instance

Once a UDT definition is created, you can create an instance of the UDT as an actual Tag in Ignition. Now that you have the Motor UDT created from the previous section, let's create a UDT instance.

When creating UDT instances, it's a good idea to create a folder to keep your Tags organized. In this example, we will be creating a bunch of motors, so create a Motors folder in your Tag Browser to keep all your Motor Tags organized.

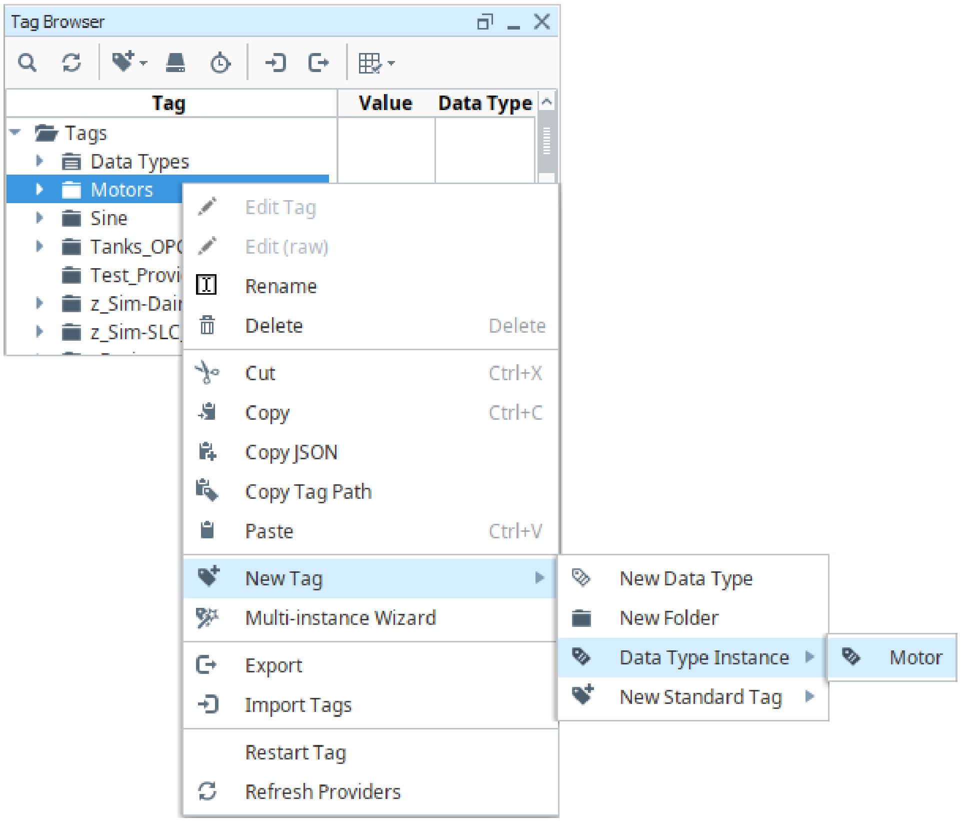
INDUCTIVE UNIVERSITY

Manually Creating UDT Instances

Watch the Video

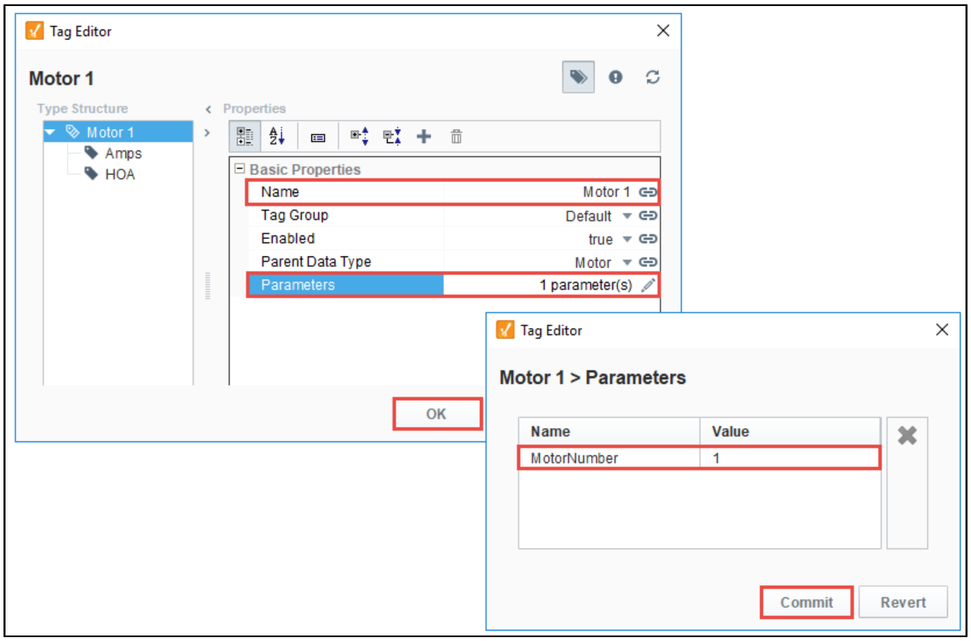
To create a UDT instance, do the following:

1. Right click on the **Tags folder > New Tag > New Folder**. Enter a name (i.e., Motors) for the folder and click **OK**.
2. In the Tag Browser, right-click on your **Motors** folder and select **New Tag > Data Type Instance > Motor** to create a new instance.



3. The Tag Editor window will open and you'll see a **New Instance** in the Type Structure area. Next to the Name property, assign the new instance a name, **Motor 1.**

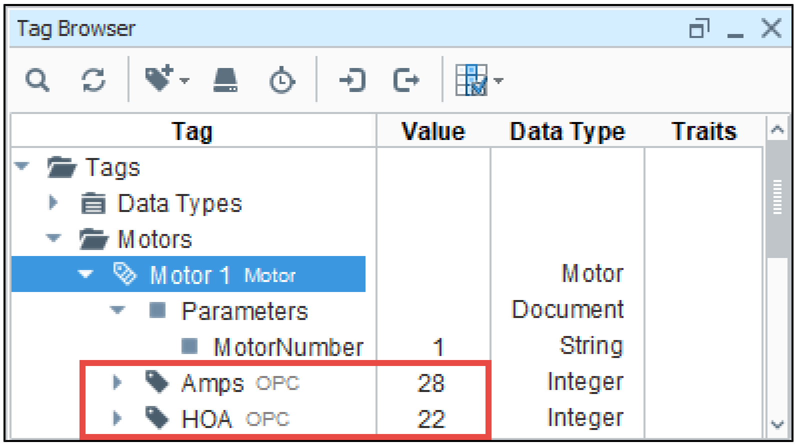
4. Enter the MotorNumber value for the **Parameters** property by clicking on the **Edit** ✎ icon. The Parameters window will open.
5. Enter a value for the **MotorNumber** that is used in the configuration of the UDT Motor. In this example, enter '**1**' and click **Commit**. (**Note:** Id you want to cancel the entry and go back to the previous window, click **Revert**.)
6. Click **OK** to create the Motor 1 instance.



Name	Motor 1
Tag Group	Default
Enabled	true
Parent Data Type	Motor
Parameters	1 parameter(s)

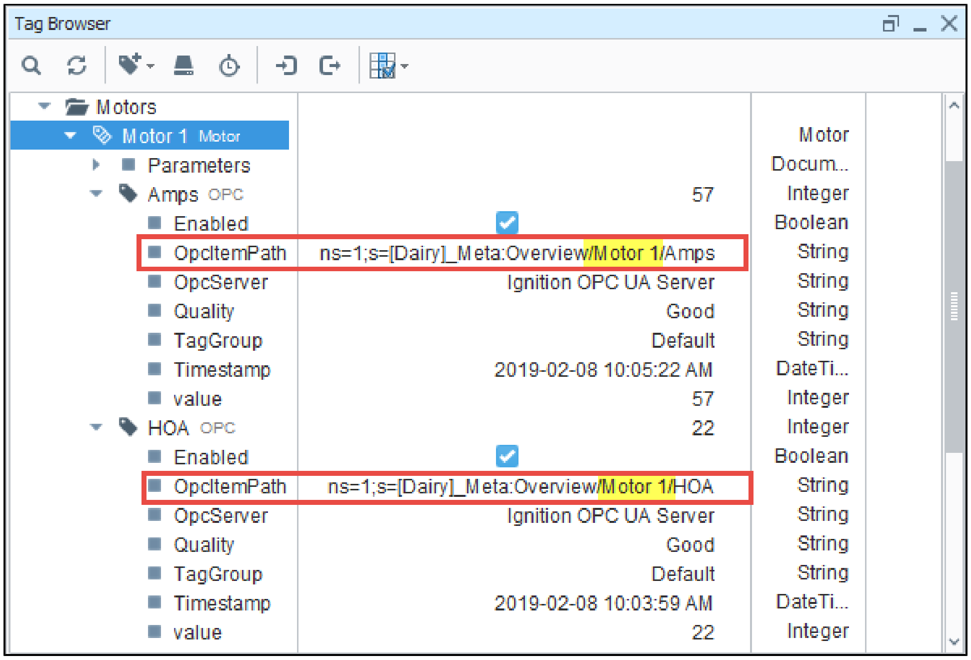
Name	Value
MotorNumber	1

7. From the **Tag Browser**, expand **Motor 1** to verify that the Tags are working correctly.



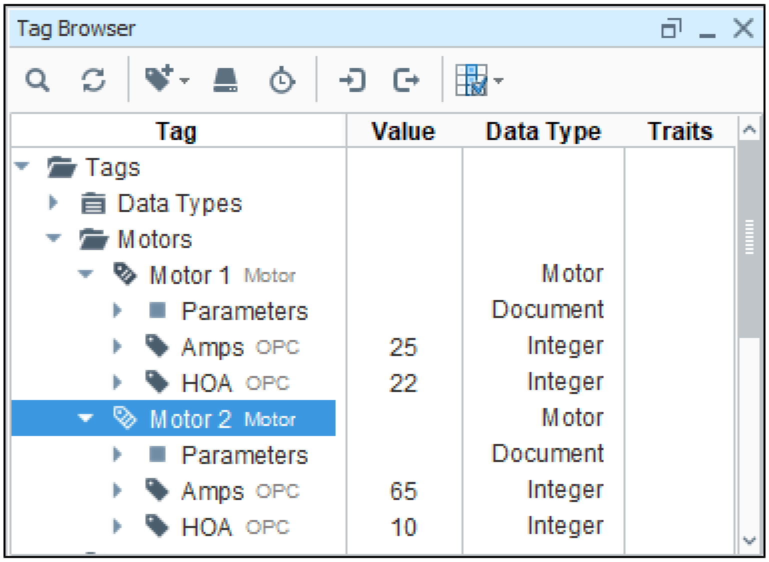
Tag	Value	Data Type	Traits
Tags			
Data Types			
Motors			
Motor 1 Motor		Motor	
Parameters		Document	
MotorNumber	1	String	
Amps OPC	28	Integer	
HOA OPC	22	Integer	

8. To verify that the OPC Item Path for the MotorNumber parameter was translated correctly, expand each of the Tags: **Amps** and **HOA** . You can see that MotorNumber (i.e., Motor 1) is displayed correctly.



Motors		
Motor 1 Motor		Motor
Parameters		Docum...
Amps OPC	57	Integer
Enabled	✓	Boolean
OpcItemPath	ns=1;s=[Dairy]_Meta:Overview/Motor 1/Amps	String
OpcServer	Ignition OPC UA Server	String
Quality	Good	String
TagGroup	Default	String
Timestamp	2019-02-08 10:05:22 AM	DateTi...
value	57	Integer
HOA OPC	22	Integer
Enabled	✓	Boolean
OpcItemPath	ns=1;s=[Dairy]_Meta:Overview/Motor 1/HOA	String
OpcServer	Ignition OPC UA Server	String
Quality	Good	String
TagGroup	Default	String
Timestamp	2019-02-08 10:03:59 AM	DateTi...
value	22	Integer

9. Repeat Steps 2 through 6 for each Motor you want to create. Each instance name must be unique. (Your next motor could be Motor 2.) Expand the Motors folder to see if Motor 2 was created and the Amps and HOA Tags are working.



Tag	Value	Data Type	Traits
Tags			
Data Types			
Motors			
Motor 1 Motor		Motor	
Parameters		Document	
Amps OPC	25	Integer	
HOA OPC	22	Integer	
Motor 2 Motor		Motor	
Parameters		Document	
Amps OPC	65	Integer	
HOA OPC	10	Integer	

Override Properties in UDT Instances

It is possible to override properties of a UDT instance to create a different base structure as compared to other instances of the same data type. This is an important feature in certain scenarios where you have to be able to override parts of the configuration of that instance. For example, you may have to create an instance of a UDT that uses a completely different OPC address scheme from the definition.

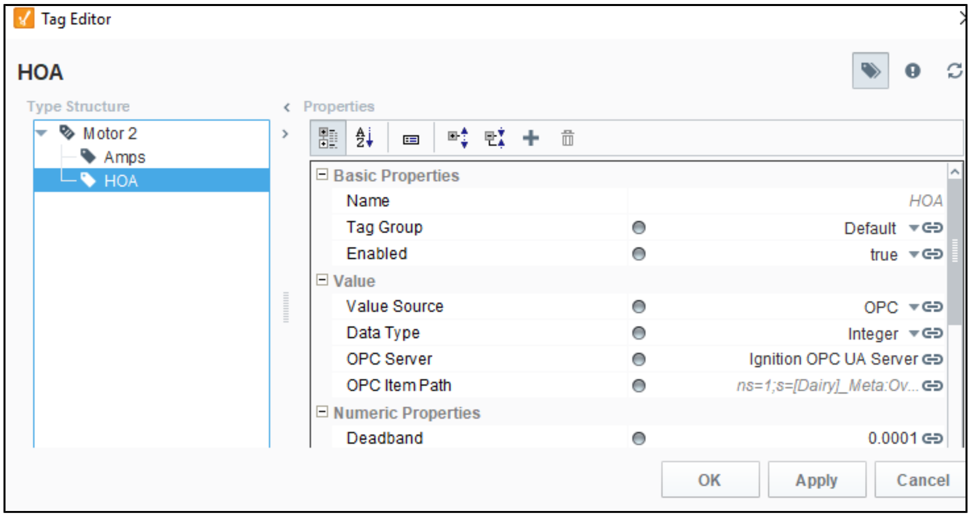
All the instances of the same data type definition have the same configuration, but when there is an instance which needs to have a different value from the rest of them, you can override that value. Let's use the Motor example from above and override the **OPC Item Path** property of the **HOA** Tag for the **Motor 2** instance.

INDUCTIVE UNIVERSITY

Overriding Properties in UDT Instances

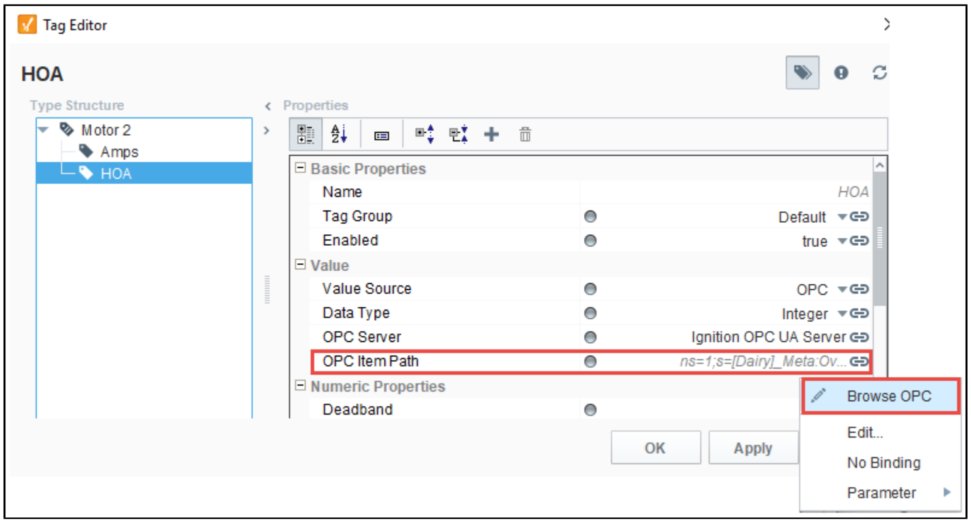
Watch the Video

1. From the **Tag Browser**, you would go to **Tags > Data Types Motors > Motor 2.**
2. Right-click on **Motor 2** and select **Edit tag** to open the Tag Editor.
3. Under **Type Structure**, select the **HOA** Tag. The circles to the riht of the property names are the override buttons. You'll notice that none of the properties have been overriden, otherwise they would be green.



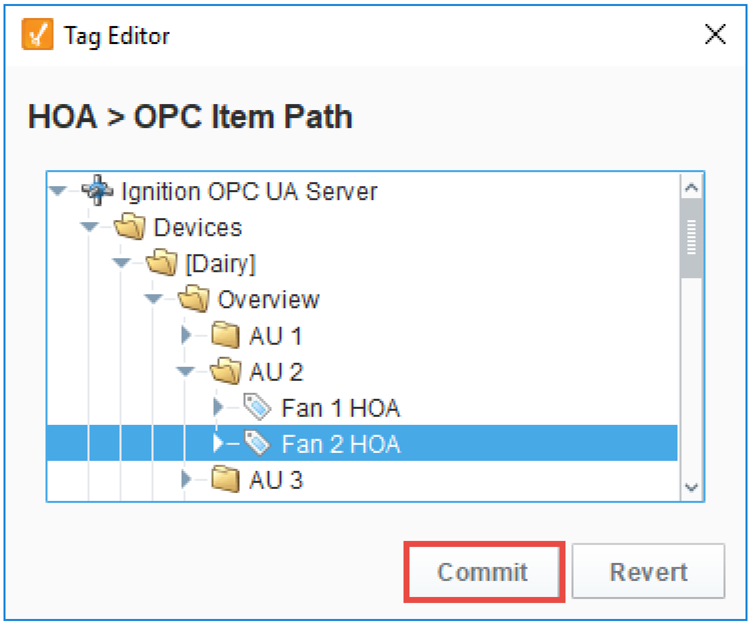
Basic Properties		
Name		HOA
Tag Group	●	Default
Enabled	●	true
Value		
Value Source	●	OPC
Data Type	●	Integer
OPC Server	●	Ignition OPC UA Server
OPC Item Path	●	ns=1;s=[Dairy]_Meta:Ov...
Numeric Properties		
Deadband	●	0.0001

4. Let's override the **OPC Item Path** for the **HOA.** Click the **binding** 🔗 icon and select **Browse OPC.**

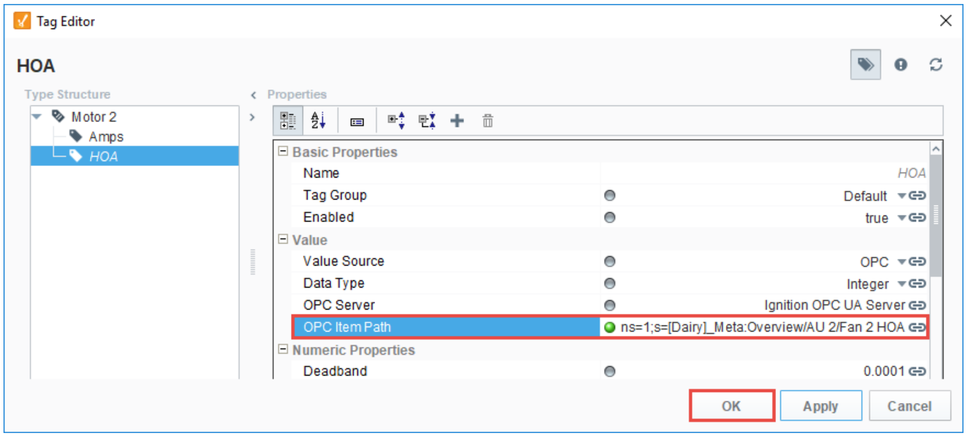


Basic Properties	
Name	HOA
Tag Group	Default
Enabled	true
Value	
Value Source	OPC
Data Type	Integer
OPC Server	Ignition OPC UA Server
OPC Item Path	ns=1;s=[Dairy]_Meta:Ov...
Numeric Properties	
Deadband	

5. In this example, browse the OPC and select another HOA Tag, and click **Commit**.

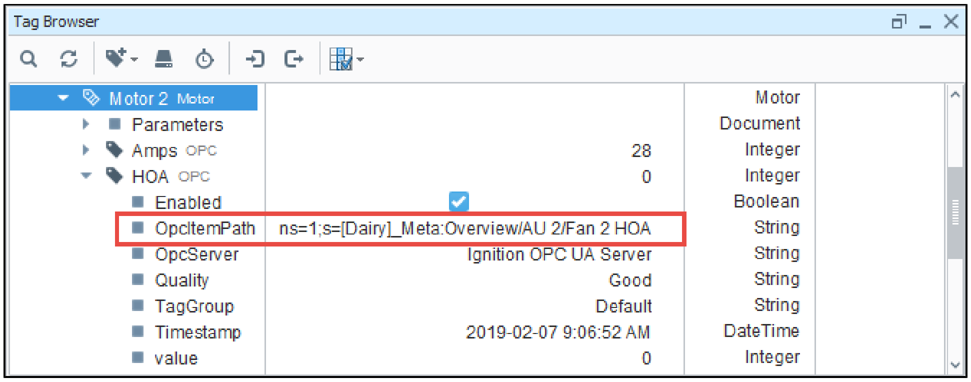


6. You'll notice the circle changed to green thus overriding the OPC Item Path for the HOA Tag. Click **OK**.



Basic Properties	
Name	HOA
Tag Group	Default
Enabled	true
Value	
Value Source	OPC
Data Type	Integer
OPC Server	Ignition OPC UA Server
OPC Item Path	ns=1;s=[Dairy]_Meta:Overview/AU 2/Fan 2 HOA
Numeric Properties	
Deadband	0.0001

7. Go to the **Tag Browser**, expand the HOA Tag for Motor 2, and see the updated OPC Item Path.



Motor 2 Motor		Motor
Parameters		Document
Amps OPC	28	Integer
HOA OPC	0	Integer
Enabled	☑	Boolean
OpcItemPath	ns=1;s=[Dairy]_Meta:Overview/AU 2/Fan 2 HOA	String
OpcServer	Ignition OPC UA Server	String
Quality	Good	String
TagGroup	Default	String
Timestamp	2019-02-07 9:06:52 AM	DateTime
value	0	Integer

8. To go back to the original configuration, unclick the green circle next to the **OPC Item Path**.

If you Change the UDT definition on a Field that was Overwritten

It's important to note that if you make a change to the definition on a field that was overwritten, it will not update on that particular instance. That instance is locked in at that override, unless you go back to your instance and remove the override by clicking the green circle.

Related Topics ...

- UDT Definitions
- UDT Multi-Instance Wizard
- UDT Inheritance

UDT Multi-Instance Wizard

The Multi-Instance Wizard provides a powerful, but simple mechanism for rapidly generating many instances of a UDT at the same time by specifying patterns for UDT parameters.

On this page ...

INDUCTIVE UNIVERSITY

UDT Multi-Instance Wizard

Watch the Video

Value Patterns and Tag Names

Value Patterns

In order to define values for parameters (and the Tag names), you can use several different types of patterns (and combinations of patterns):

Range `number1-number2[/step]`
A numeric range of values, such as `1-10`. Optionally, a `step` parameter can be included, in order to only generate numbers at certain multiples. For example, `0-100/10` would generate `0,10,20`, and so on.

Repeat `value*count`
A value (numerical or string), and the number of times to use it. For example, `North Area*10` would use the parameter `North Area` for `10` items.

List `value1, value2, value3`
A comma separated list of values (or other patterns) to use.

Examples:

1-10,21-30,31-40 Results in `30` Tags being created, with the specified value ranges (so, for example, there would be no parameter `15`).
A,B,C Results in `3` Tags, with each of the values.
0-100/5 Results in `21` Tags (because range is inclusive), with values `0, 5, 10... 100`.

As mentioned, the size of the pattern will dictate how many Tags will be created. If some patterns are smaller than others, the last value will be repeated for the other Tags.

Tag Names

The names of the generated instances can be specified using a system similar to that of the parameter patterns. If you just want to use sequential names, you don't need to specify a pattern, as values will be generated automatically starting at one. You can also set the pattern to simply be the starting number to generate sequential names from there.

Base Name
A string base for the Tag name. This can also be a list of names, in which case the names will be used directly, and the name pattern won't be used.

Name Pattern
A pattern that will be used to generate values that will be appended to the base name.

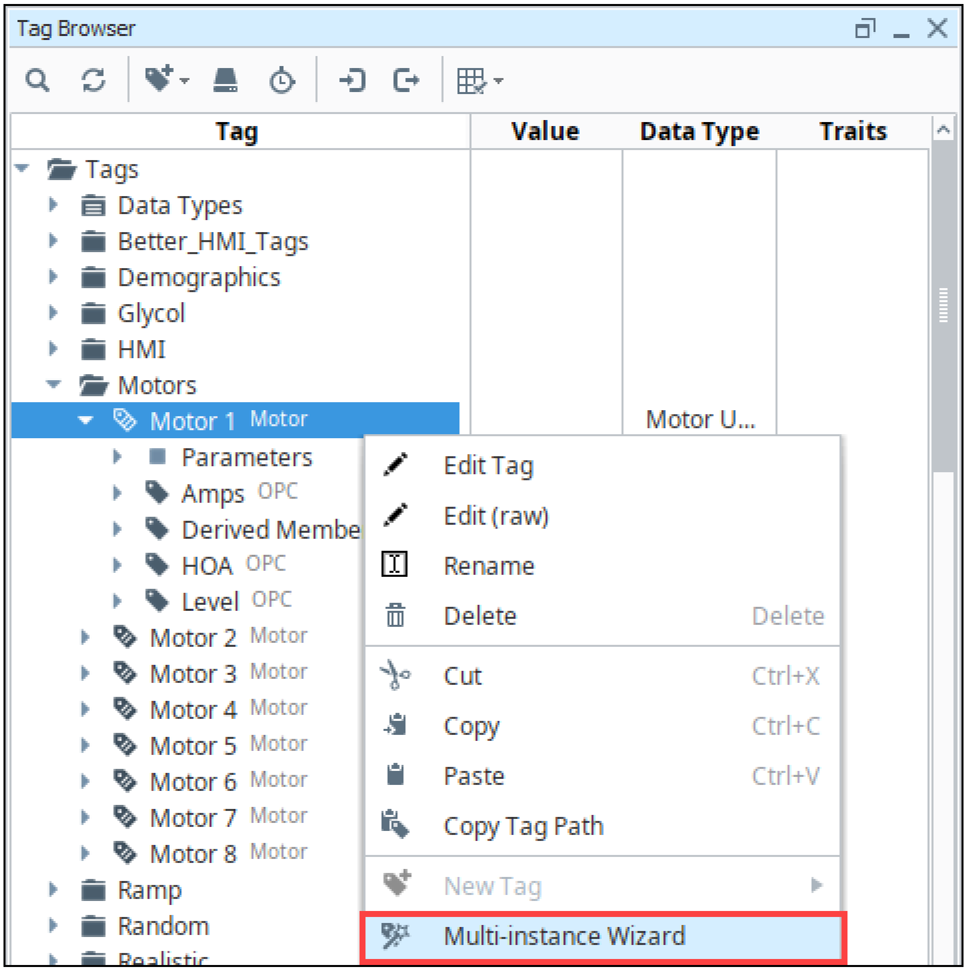
At any time, you can use the **Preview** button to view the Tag names and parameters that will be created. Once you are satisfied, click **OK** to generate the Tags under the selected folder in the Tag provider.

How to Make New Instances of a UDT

Now that we have a Motor UDT that we created on the UDT Definitions page, we can make instances of it. Instances are running copies of a data type with concrete data for all members.

1. In the **Tags Browser**, right-click on the **Motors** folder if you have one, or create a **New Folder** called **Motors**.
   You can create a single instance of the motor or use the Multi-instance Wizard to rapidly create many instances at the same time. In this example, we will use the Multi-Instance Wizard.

2. Right-click on the **Motors** folder, and select **Multi-instance Wizard** to open the Instance Creation Wizard window.

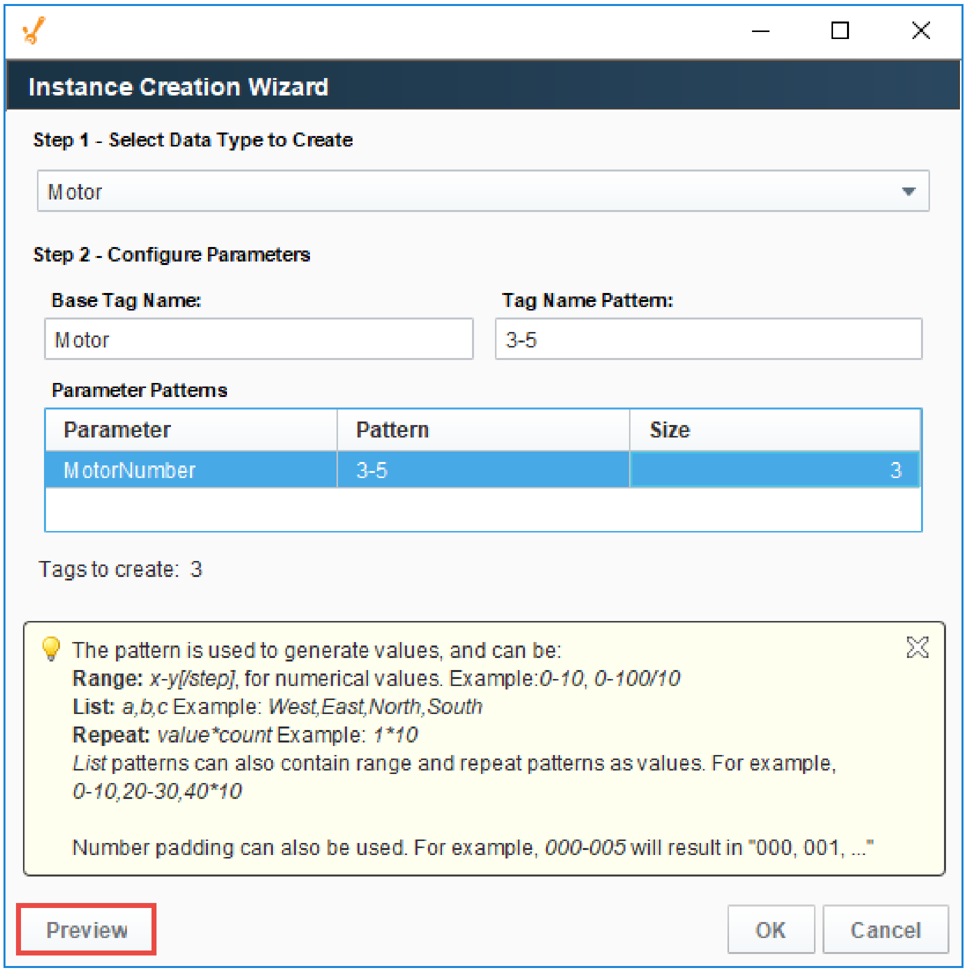


3. In **Step 1 - Select Data Type to Create**, select a UDT (i.e., **Motor**) from the dropdown.

4. In **Step 2 - Configure the Parameters**, enter the following for your Motor:

    - Base Tag Name: "**Motor** ". Note the space at the end. Without this space your Tag names will look like Motor1, Motor2, etc.
    - Tag Name Pattern: **3-5** This creates three Tags **Motor 3**, **Motor 4**, and **Motor 5**. (We already have Motor 1 and Motor 2 that we created in UDT Instances).
    - Parameter Patterns: the **MotorNumber** parameter is entered by default when we selected the data type to create in Step 1.
    - Pattern: **3-5** is the pattern of the parameter so the Motor 3 Tag will have a parameter of 3, Motor 4 will have a parameter of 4, and Motor 5 will have a parameter of 5.

    You'll notice that after you enter the Pattern, the number of Tags to create is updated. In this example, three Motor Tags will be created. Click **Preview.**



Instance Creation Wizard

Step 1 - Select Data Type to Create

Motor

Step 2 - Configure Parameters

Base Tag Name: Motor

Tag Name Pattern: 3-5

Parameter Patterns

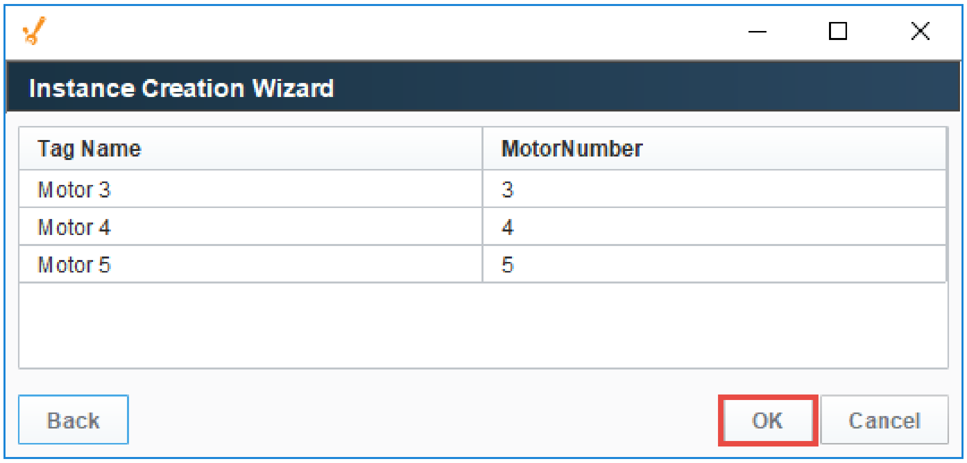
Parameter	Pattern	Size
MotorNumber	3-5	3

Tags to create: 3

The pattern is used to generate values, and can be:
Range: *x-y[/step]*, for numerical values. Example:*0-10, 0-100/10*
List: *a,b,c* Example: *West,East,North,South*
Repeat: *value*count* Example: *1*10*
List patterns can also contain range and repeat patterns as values. For example, *0-10,20-30,40*10*

Number padding can also be used. For example, *000-005* will result in "000, 001, ..."

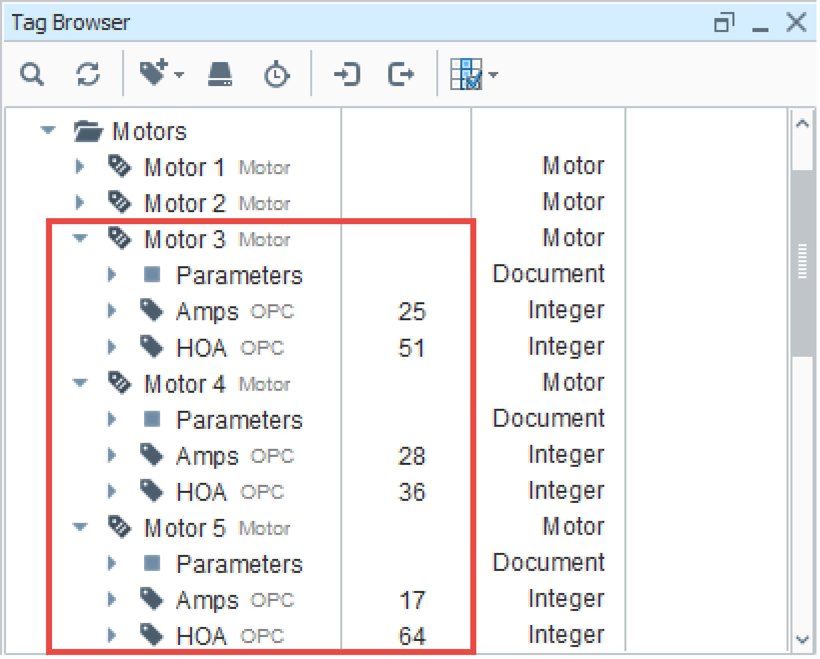
5. In Preview, you will see how the Base Tag Names and Parameter Values get created. Click **Back** to go back to the Instance Creation Wizard window if you want to make an update. If you like what you see on the Preview window, click **OK**.



Instance Creation Wizard

Tag Name	MotorNumber
Motor 3	3
Motor 4	4
Motor 5	5

6. In the Tag Browser, expand Motor Tags 3-5 to see if all the members of the UDT were created and are running.



Motors		
Motor 1 Motor		Motor
Motor 2 Motor		Motor
Motor 3 Motor		Motor
Parameters		Document
Amps OPC	25	Integer
HOA OPC	51	Integer
Motor 4 Motor		Motor
Parameters		Document
Amps OPC	28	Integer
HOA OPC	36	Integer
Motor 5 Motor		Motor
Parameters		Document
Amps OPC	17	Integer
HOA OPC	64	Integer

Cannot Edit Existing Instances using the Multi-Instance Wizard

You cannot edit existing instances using the Mult-Instance Wizard. The Mult-Instance Wizard is only used for quickly creating many instances of a UDT at the same time. If you want to make a change to all your instances, refer to How to Edit an Existing UDT.

Related Topics ...

- Overriding Properties in UDT Instances
- UDT Instances
- UDT Definitions

UDT Inheritance

Once you have a single data type created, it is possible to set up UDT inheritance where data types extend to other data types, to add additional members, or override default values. For example, you can create a new data type and using the inheritance feature it will inherit all Tags from the parent data type including the parameters. Then you can add additional Tags and/or override any settings in your new data type. UDT Inheritance is a way to extend to a class of data types to add more functionality to that class.

For example, you may have a simple motor and a complex motor. The Complex motor can inherit from the simple motor, which means all simple motor values will be in the complex motor and you can add more.

Nesting (using one or more UDTs to make up a larger UDT) is different from inheritance and can be found under UDT Nesting.

On this page ...

INDUCTIVE UNIVERSITY

UDT Inheritance

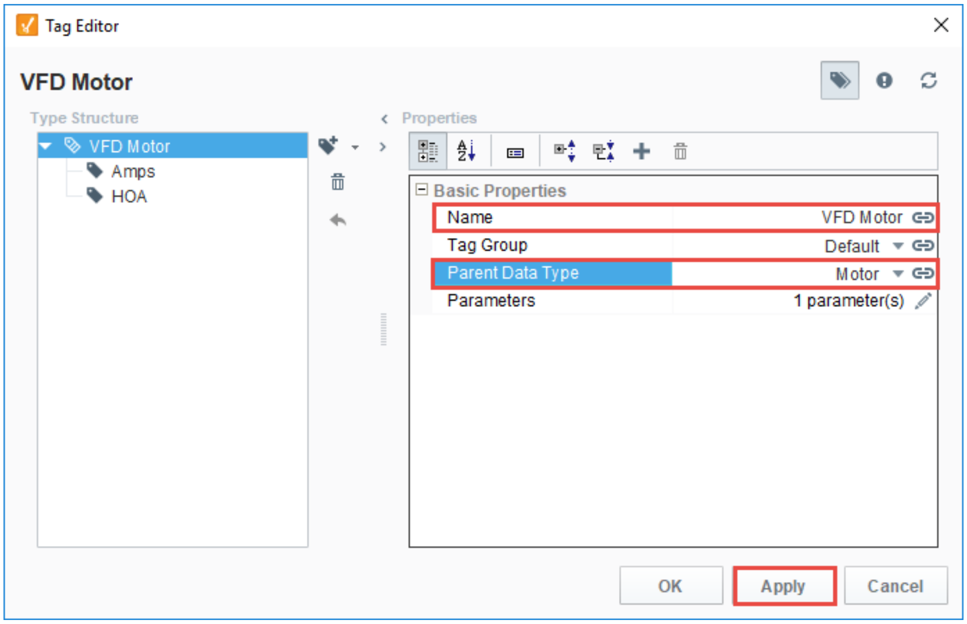
Watch the Video

To Inherit Property Values from an Existing UDT

Let's use our data type Motor from the previous sections to create another data type. We'll set the parent to Motor so our new data type automatically inherits all the properties of Motor.

1. From **Tag Browser**, right-click on **Data Types**, and choose **New Tag >New Data Type**.
   The Tag Editor window will open.

2. In the **Name** field, enter a new name (i.e. VFD Motor), and from the **Parent Data Type** dropdown, choose an existing UDT (i.e. Motor). Click **Apply** and the new UDT will automatically inherit all the properties of the parent UDT Motor: Amps and HOA.



Basic Properties

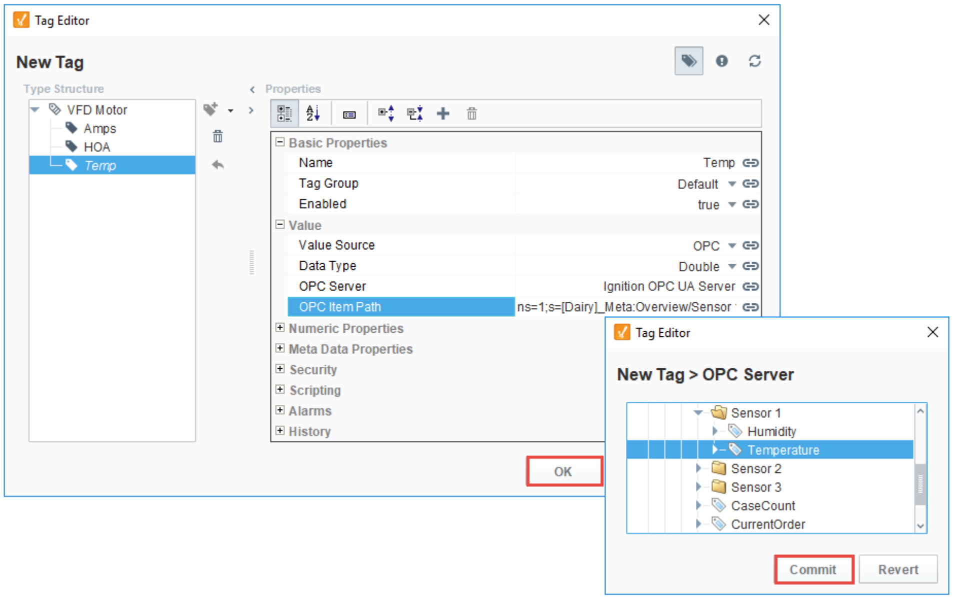
Name	VFD Motor
Tag Group	Default
Parent Data Type	Motor
Parameters	1 parameter(s)

3. With the Tag Editor still open, let's add a new OPC Tag to this new UDT. Click on the **Add Tag** icon and select **New Standard Tag > OPC Tag**.

4. In the Type Structure, click on New Tag, and enter the properties for your new Tag:

   Name: **Temp**
   Data Type: **Double**
   OPC Server: Click on the binding ( ) icon and select **Browse OPC Server**.
   OPC Item Path: Browse the **OPC** and find the Tag you want to use. This example uses a **Temperature** Tag from a Sensor in the Dairy program. Click **Commit**.

   Click **OK** to save your new Tag **(i.e., VDF Motor)**.



Basic Properties

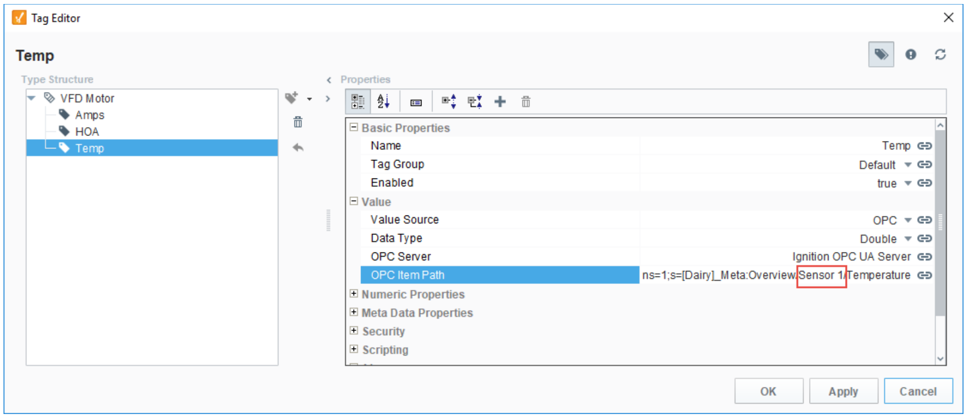
Name	Temp
Tag Group	Default
Enabled	true

Value

Value Source	OPC
Data Type	Double
OPC Server	Ignition OPC UA Server
OPC Item Path	ns=1;s=[Dairy]_Meta:Overview/Sensor

Numeric Properties
Meta Data Properties
Security
Scripting
Alarms
History

5. You can see in the image below that the Temp Tag is pointing to a specific address in the PLC. Because we're creating a new Tag in our UDT, we don't want to point to one specific set of **'Temp'** Tags. We want each instance of the VFD Motor UDT to reference a different set of **'Temp'** Tags. To do that, we need to add a parameter to the VFD Motor data type that we will call **'SensorNumber'.**



Tag Editor

Temp

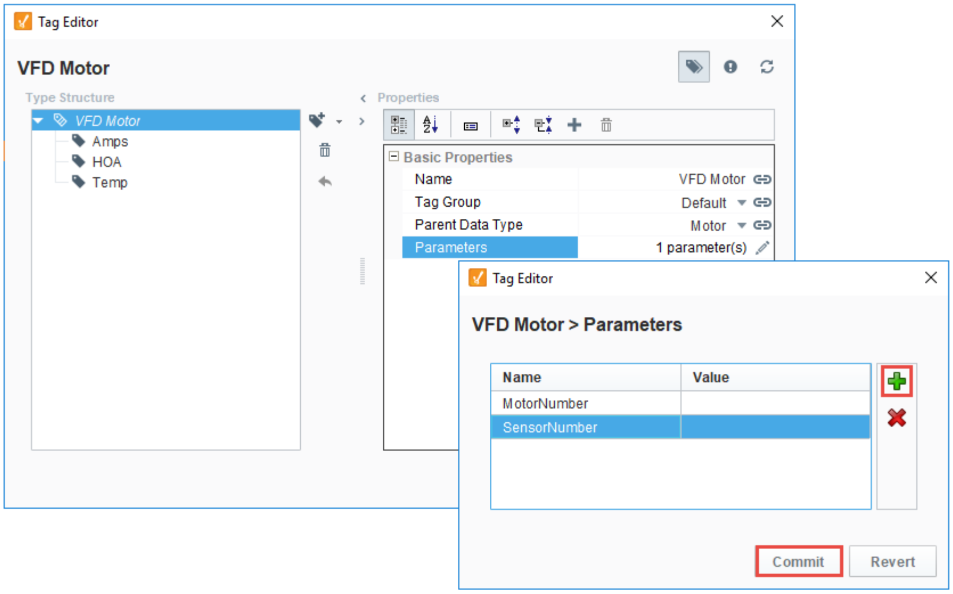
Type Structure

- VFD Motor
  - Amps
  - HOA
  - Temp

Properties

Basic Properties	
Name	Temp
Tag Group	Default
Enabled	true
Value	
Value Source	OPC
Data Type	Double
OPC Server	Ignition OPC UA Server
OPC Item Path	ns=1;s=[Dairy]_Meta:Overview/Sensor 1/Temperature
Numeric Properties	
Meta Data Properties	
Security	
Scripting	

6. To create a new UDT parameter, select the **VFD Motor**, and click the **Edit** ✎ icon next to the **Parameters** property. The Parameters window will open.
7. Click the Add ✚ icon and add the new parameter **'SensorNumber'**. Click **Commit**.



Tag Editor

VFD Motor

Type Structure

- VFD Motor
  - Amps
  - HOA
  - Temp

Properties

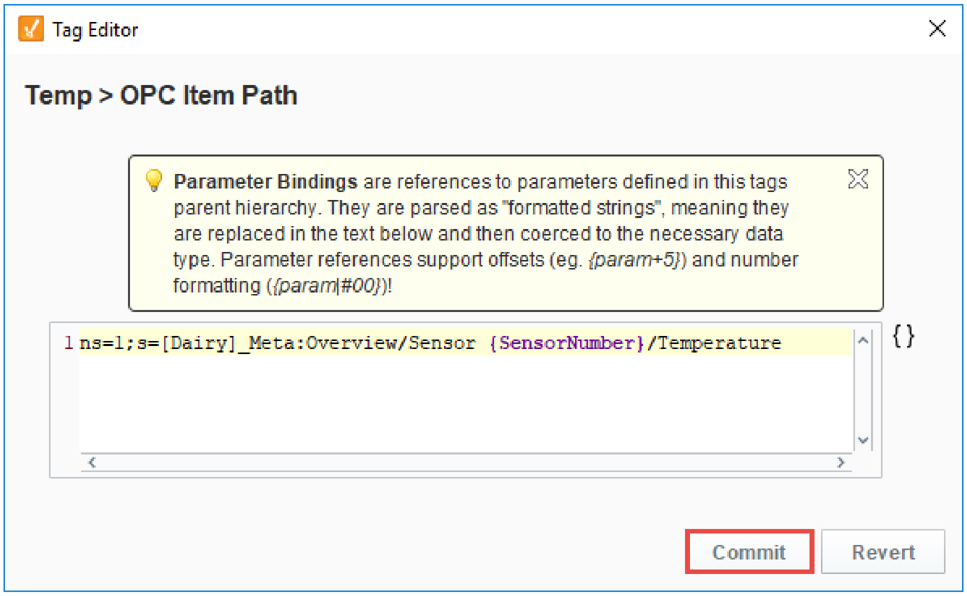
Basic Properties	
Name	VFD Motor
Tag Group	Default
Parent Data Type	Motor
Parameters	1 parameter(s)

Tag Editor

VFD Motor > Parameters

Name	Value
MotorNumber	
SensorNumber	

8. With the Tag Editor still open, select the **"Temp'** Tag. In the **OPC Item Path** field, click the **binding** 🔗 icon, select **Edit**, and the **Temp > OPC Item Path** window will open. Place your cursor at the end of **'Sensor1',** delete the **'1',** add a space, and enter **'{SensorNumber}'**. Don't forget the curly braces. In the OPC Item Path it will look like the following, then click **Commit** to save your updates and go back to the previous window.

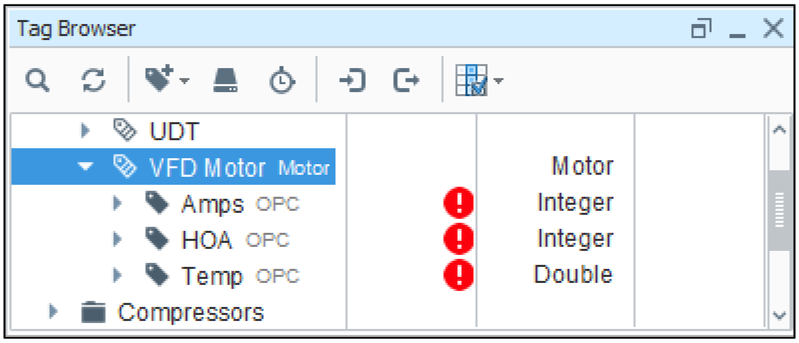


Temp > OPC Item Path

Parameter Bindings are references to parameters defined in this tags parent hierarchy. They are parsed as "formatted strings", meaning they are replaced in the text below and then coerced to the necessary data type. Parameter references support offsets (eg. *{param+5}*) and number formatting (*{param|#00}*)!

```
ns=1;s=[Dairy]_Meta:Overview/Sensor {SensorNumber}/Temperature
```

9. Click **OK**. The new data type called VFD Motor inherited all the Tags from the Motor data type including the new Temp Tag which are now visible in the Tag Browser. Don't panic if you see the exclamation point ❗ next to any Tags. This is normal! The Tags are defined, by they don't have any values which is the reason for the Bad_Stale warning when you hover of the icon.

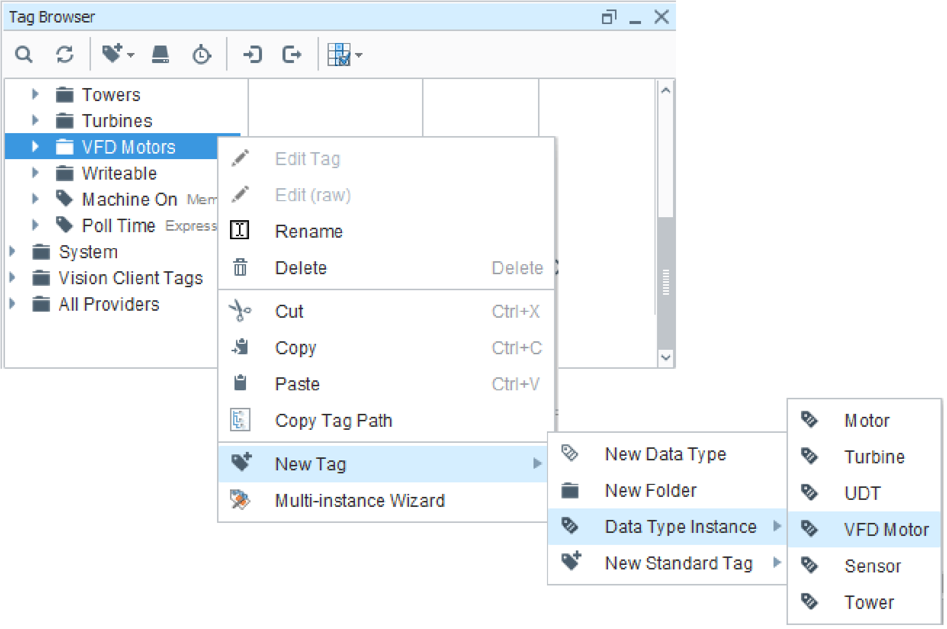


UDT		
VFD Motor Motor		Motor
Amps OPC	❗	Integer
HOA OPC	❗	Integer
Temp OPC	❗	Double
Compressors		

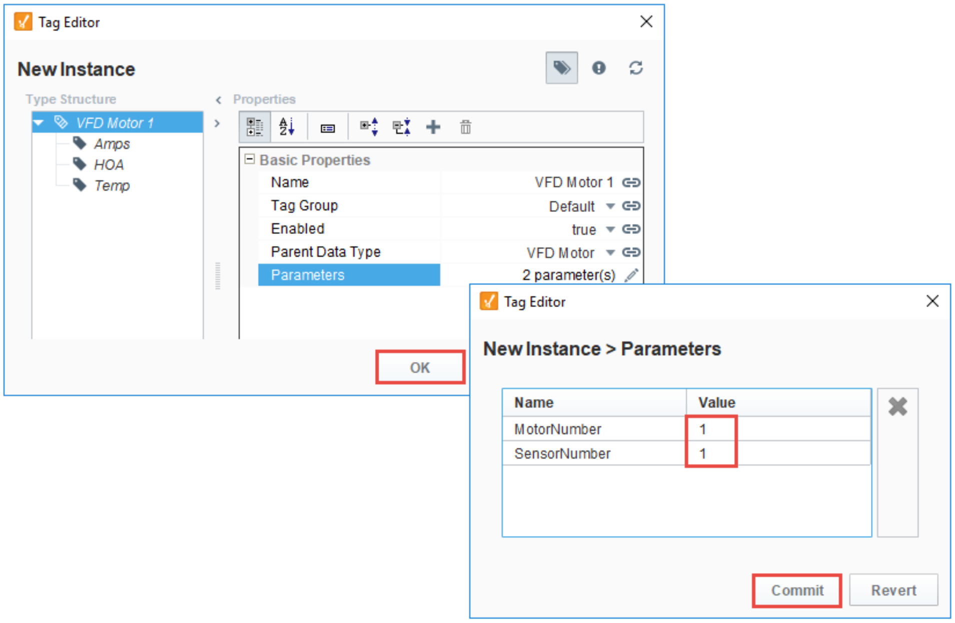
Creating the Data Type Instance

Now that our UDT is set up using inherited data types, a new Tag and one new parameter, let's create a data type instance of the VFD Motor

1. Go directly to the **Tag Browser,** and select **Tags > New Tag > Data Type Instance > VFD Motor.** This example we created a VFD Motors folder so all the instances were placed together in one folder.



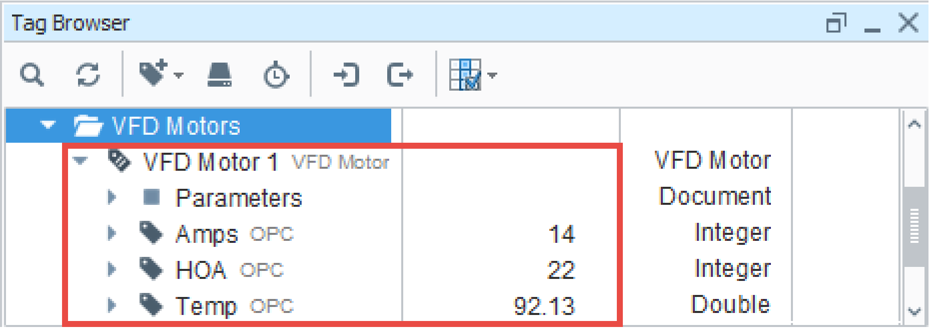
2. The **Tag Editor** window will open. Enter the **Name** for the instance (i.e., VFD Motor 1). Click the **Edit** icon next to the **Parameters** property and enter the parameter values of **'1'** for **MotorNumber** and **SensorNumber,** and press **Commit.** Then click **OK** to create the instance.



Basic Properties	
Name	VFD Motor 1
Tag Group	Default
Enabled	true
Parent Data Type	VFD Motor
Parameters	2 parameter(s)

Name	Value
MotorNumber	1
SensorNumber	1

3. Now, you'll be able to see all the values for VFD Motor 1 including the Temp Tag that was added.



Tag Browser

VFD Motors		
VFD Motor 1 VFD Motor		VFD Motor
Parameters		Document
Amps OPC	14	Integer
HOA OPC	22	Integer
Temp OPC	92.13	Double

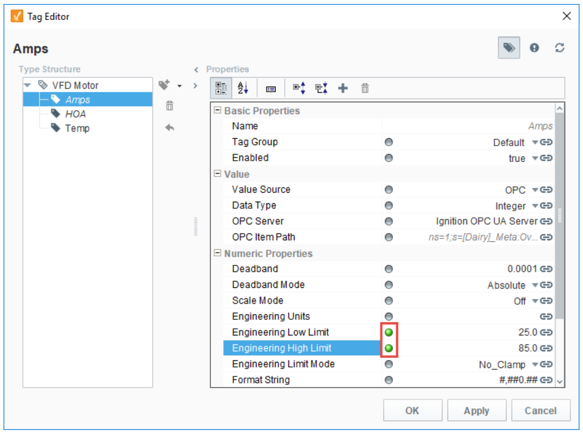
Overriding Properties of the Parent UDT

Another nice benefit of the UDT inheritance feature is it allows you to override some of the properties of the parent. For example, since the VFD Motor has Motor as the parent, you can go to any of the Tags and override any of the settings of that data type. Click the circle to the right of the property and enter a new value, or change a property's value and the green circle changes to green automatically. This overrides the property inherited from the parent. To learn more, go to Overriding Properties in UDT Instances.

INDUCTIVE UNIVERSITY

Overriding Properties in UDT Instances

Watch the Video



Tag Editor

Amps

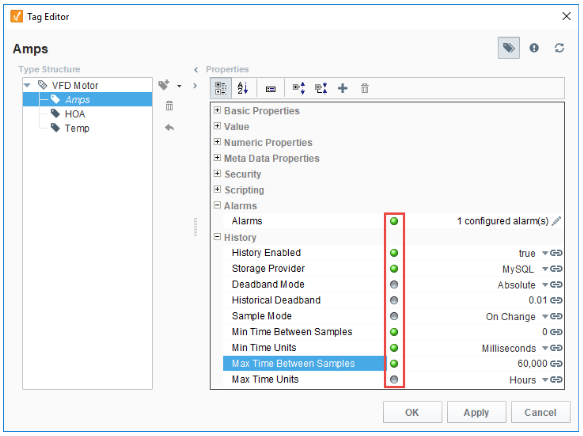
Type Structure

- VFD Motor
  - Amps
  - HOA
  - Temp

Properties

Basic Properties		
Name		Amps
Tag Group	●	Default
Enabled	●	true
Value		
Value Source	●	OPC
Data Type	●	Integer
OPC Server	●	Ignition OPC UA Server
OPC Item Path	●	ns=1;s=[Dairy]_Meta:Ov...
Numeric Properties		
Deadband	●	0.0001
Deadband Mode	●	Absolute
Scale Mode	●	Off
Engineering Units	●	
Engineering Low Limit	🟢	25.0
Engineering High Limit	🟢	85.0
Engineering Limit Mode	●	No_Clamp
Format String	●	#,##0.##

You can also turn on Alarming and History that wasn't initially turned on in the parent UDT by simply using the override feature. Next to the Alarm property, click the green circle to change it to green, and click the pencil (✎) icon to configure the alarm if it is not already not configured. If you want to turn on History, click the green circle or change any of the History properties which will cause any of the green circles to change from gray to green.



Tag Editor

Amps

Type Structure

- VFD Motor
  - Amps
  - HOA
  - Temp

Properties

⊞ Basic Properties		
⊞ Value		
⊞ Numeric Properties		
⊞ Meta Data Properties		
⊞ Security		
⊞ Scripting		
⊟ Alarms		
Alarms	🟢	1 configured alarm(s)
⊟ History		
History Enabled	🟢	true
Storage Provider	🟢	MySQL
Deadband Mode	●	Absolute
Historical Deadband	●	0.01
Sample Mode	●	On Change
Min Time Between Samples	🟢	0
Min Time Units	🟢	Milliseconds
Max Time Between Samples	🟢	60,000
Max Time Units	●	Hours

Related Topics ...

- UDT Definitions
- UDT Nesting

UDT Nesting

It's possible to set up UDT nesting in Ignition where you are putting one UDT inside of another UDT. The UDT is nested as an instance within another UDT. It facilitates quicker development of projects since you're able to piece together multiple UDT definitions as needed without having to build everything from the ground up as with each UDT definition. This is particularly useful because it promotes rapid development if you are expanding a plant or facility where all you have to do is make a few Tag changes to existing parameters and property settings.

For example, you may have a production line that is built out of several different machines. You don't need to re-create a motor for each line, instead you can create it once and use it in every line.

Inheritance (having simple and complex version of similar objects) is different from nesting UDTs, and can be found under UDT Inheritance.

On this page ...

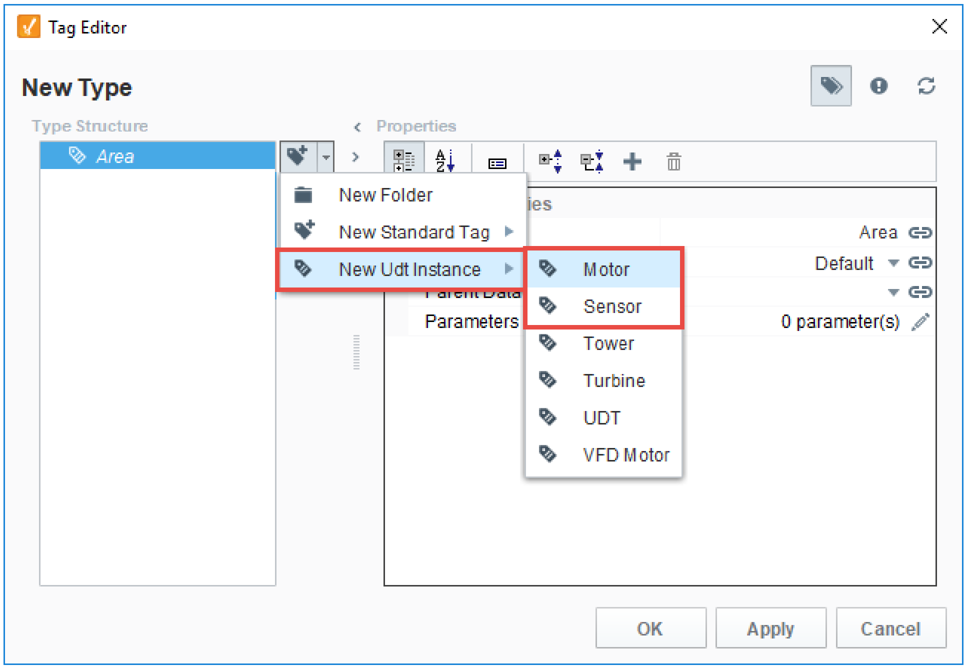
Nested UDTs

Watch the Video

Set Up UDT Nesting

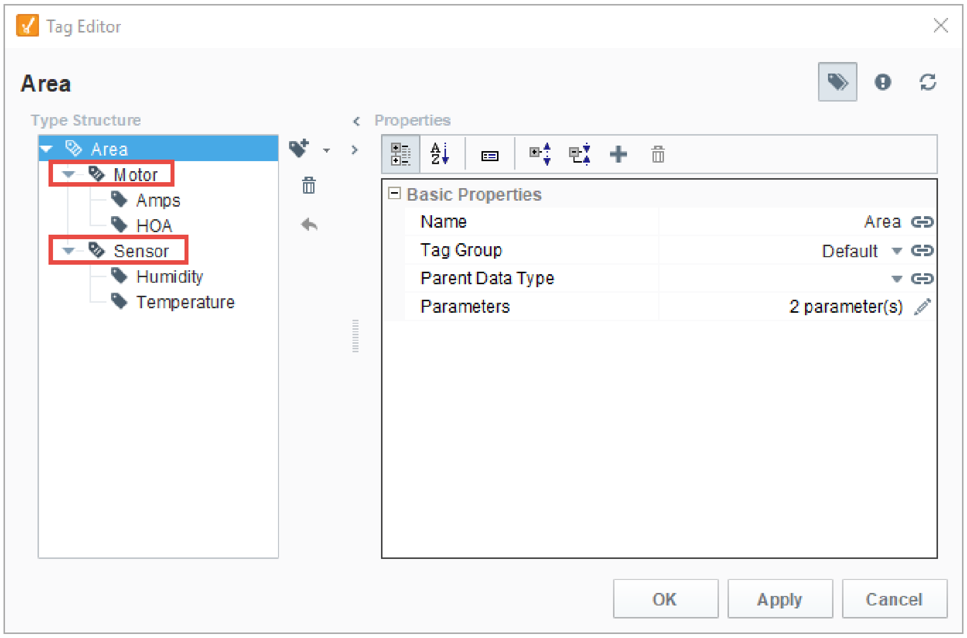
In this example, let's use our a Motor and Sensor data types that were created and used in previous sections of this manual. We are going to create a third UDT called Area that will contain the Motor and Sensor data types inside of it.

1. In the Tag Browser, right click on **Data Types** and select **New Tag > New Data Type**. The Tag Editor window will open. Assign the new data type a Name called **Area**.
2. Inside of the **Area** data type, create two data type instances, one for Motor and the other for Sensor.
   Click on the **Tag** 🏷 icon, select **New UDT Instance > New Motor**. Rename the new data type to **'Motor'**. Click **Apply**.
   Click on the **Tag** 🏷 icon, select **New UDT Instance > New Sensor**. Rename the new data type to **'Sensor'**. Click **Apply**.



Area
Default
Parameters 0 parameter(s)

3. With the Tag Editor still open, you'll notice that both the Motor and Sensor UDTs were added. For every UDT that you add inside another UDT, those UDT instances have parameters that need to be specified. In this example, the Motor UDT has the **'MotorNumber'** parameter, and the Sensor UDT has the **'SensorNumber'** parameter. You must pass a value into the each of these UDTs (Motor and Sensor) from the parent UDT (Area). To view the parameters for each UDT instance, select each UDT instance (Motor and Sensor) and click the **Edit** ✎ icon.

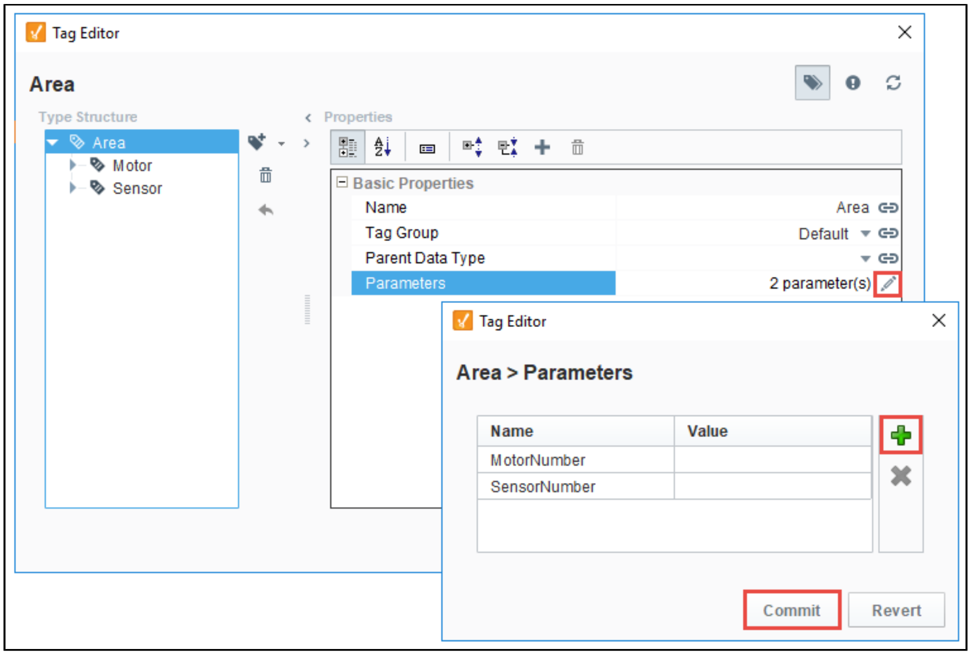


Basic Properties

Name	Area
Tag Group	Default
Parent Data Type	
Parameters	2 parameter(s)

4. Now that you know what parameters are in each UDT instance, go to the Area UDT, and click the **Edit** ✎ icon next to the **Parameters** property. The Parameters window will open.

5. Click on the **Add** ✚ icon to add the 'MotorNumber' and 'SensorNumber' parameters, then click **Commit**.

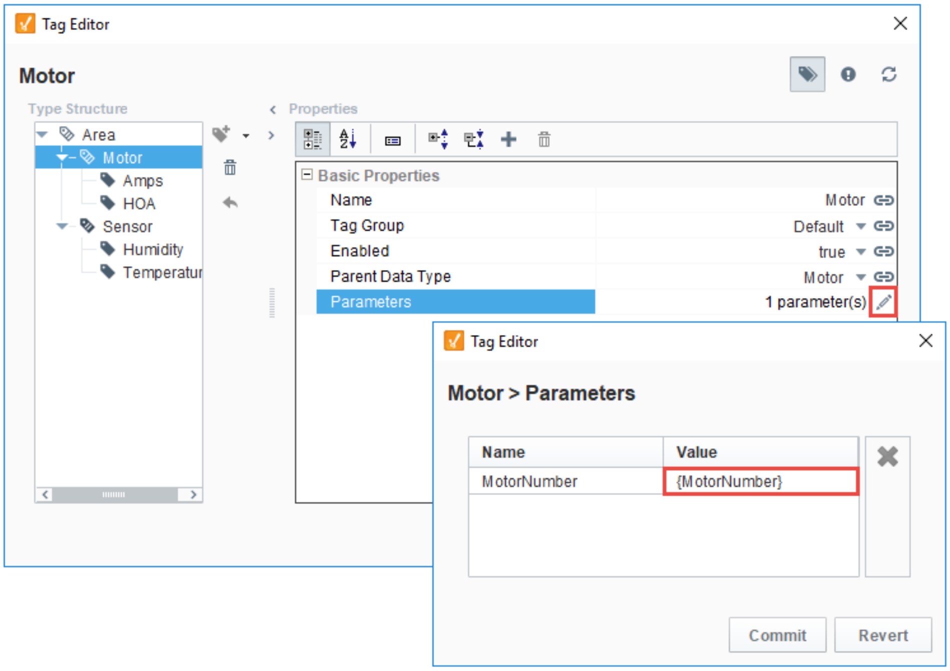


Name	Area
Tag Group	Default
Parent Data Type	
Parameters	2 parameter(s)

Name	Value
MotorNumber	
SensorNumber	

6. Now we need to pass these values into the UDT instances by adding a reference. Select the **Motor** UDT, and click the **Edit** ✎ icon next to the **Parameters** property.
7. Enter the reference for MotorNumber: **'{MotorNumber}'**. Click **Commit**.

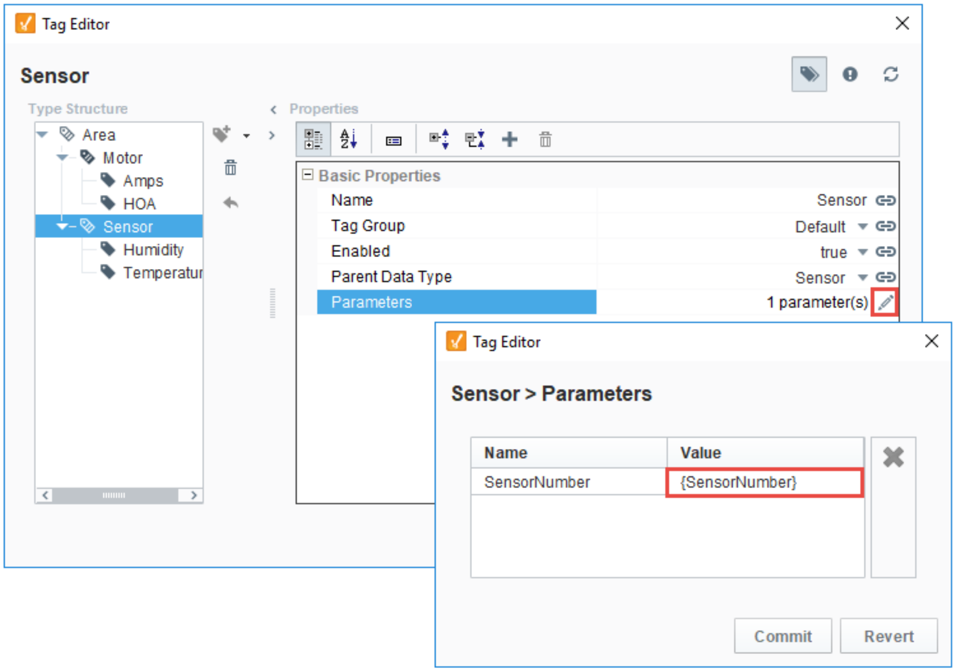
> **If Multiple Data Types use the same Parameter**
>
> In the event data types use the same parameter, you only need to enter it once in the new data type (i.e., Area).



Name	Motor
Tag Group	Default
Enabled	true
Parent Data Type	Motor
Parameters	1 parameter(s)

Name	Value
MotorNumber	{MotorNumber}

8. Select the **Sensor** UDT, and click the **Edit** ✎ icon next to the **Parameters** property. Enter the reference for SensorNumber: **'{SensorNumber}'**. Click **Commit**.



Basic Properties

Name	Sensor
Tag Group	Default
Enabled	true
Parent Data Type	Sensor
Parameters	1 parameter(s)

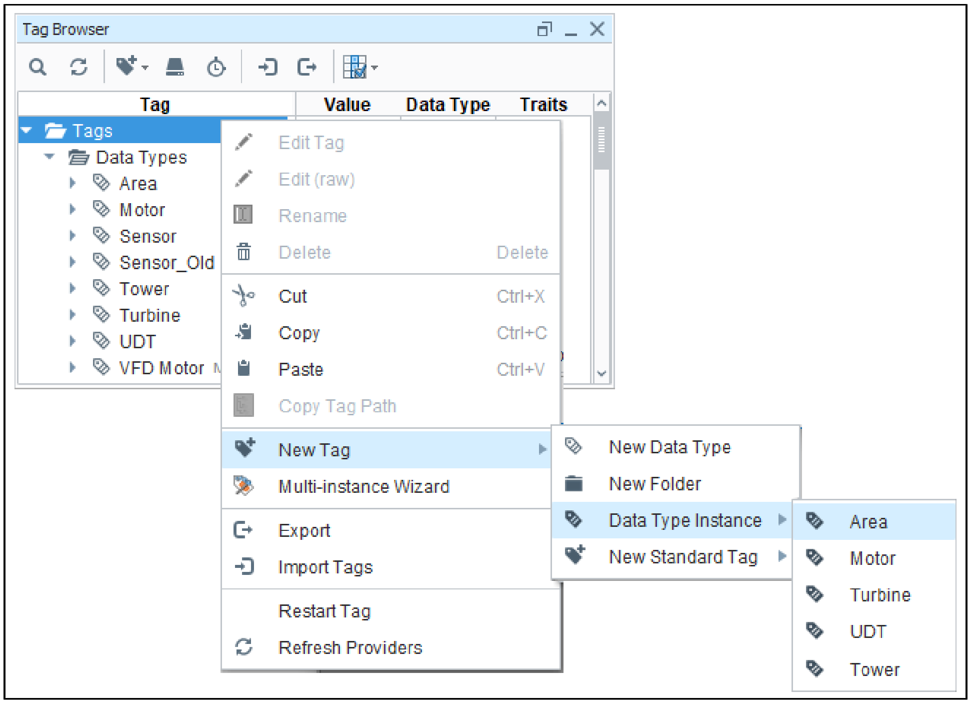
Name	Value
SensorNumber	{SensorNumber}

9. Click **OK**.

Create a New Instance of a Nested UDT

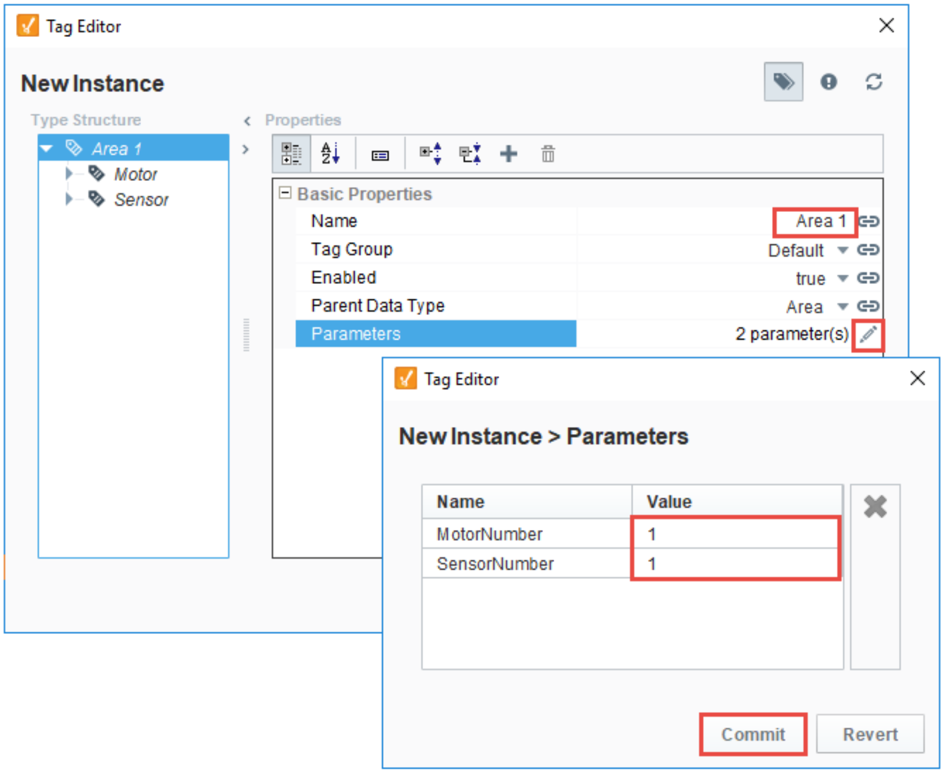
Now, that you added all your parameters, let's create a new instance of Area.

1. In the **Tag Browser**, right click on **Tags,** select **New Tag > New Data Type Instance > Area**.



2. In the Tag Editor, enter a **Name** for the new instance, (i.e., **Area 1**). Click on the **Edit** icon next to the **Parameters** property.
   Enter the values for each parameter:
   For MotorNumber, enter the value **1**.
   For SensorNumber, enter the value **1**.

Click **Commit**.
Click **OK**.



Tag Editor

New Instance

Basic Properties

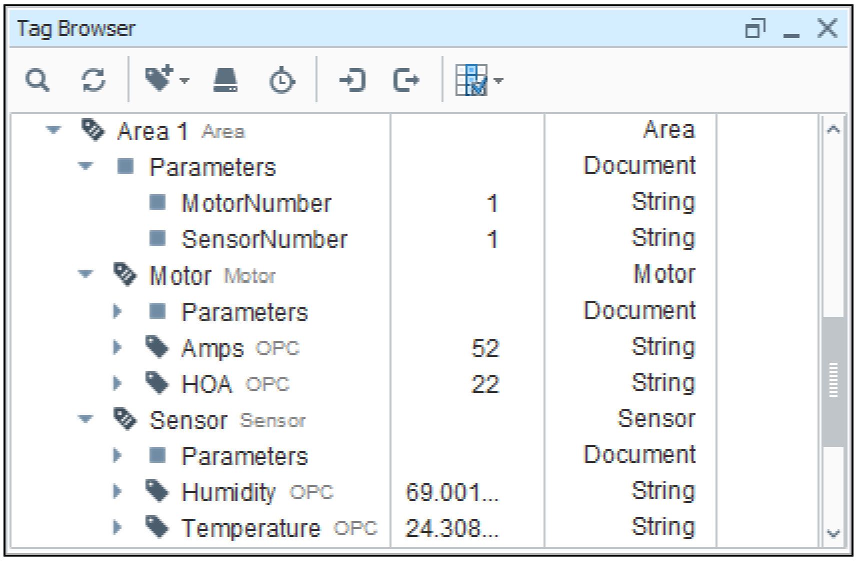
Name	Area 1
Tag Group	Default
Enabled	true
Parent Data Type	Area
Parameters	2 parameter(s)

Tag Editor

New Instance > Parameters

Name	Value
MotorNumber	1
SensorNumber	1

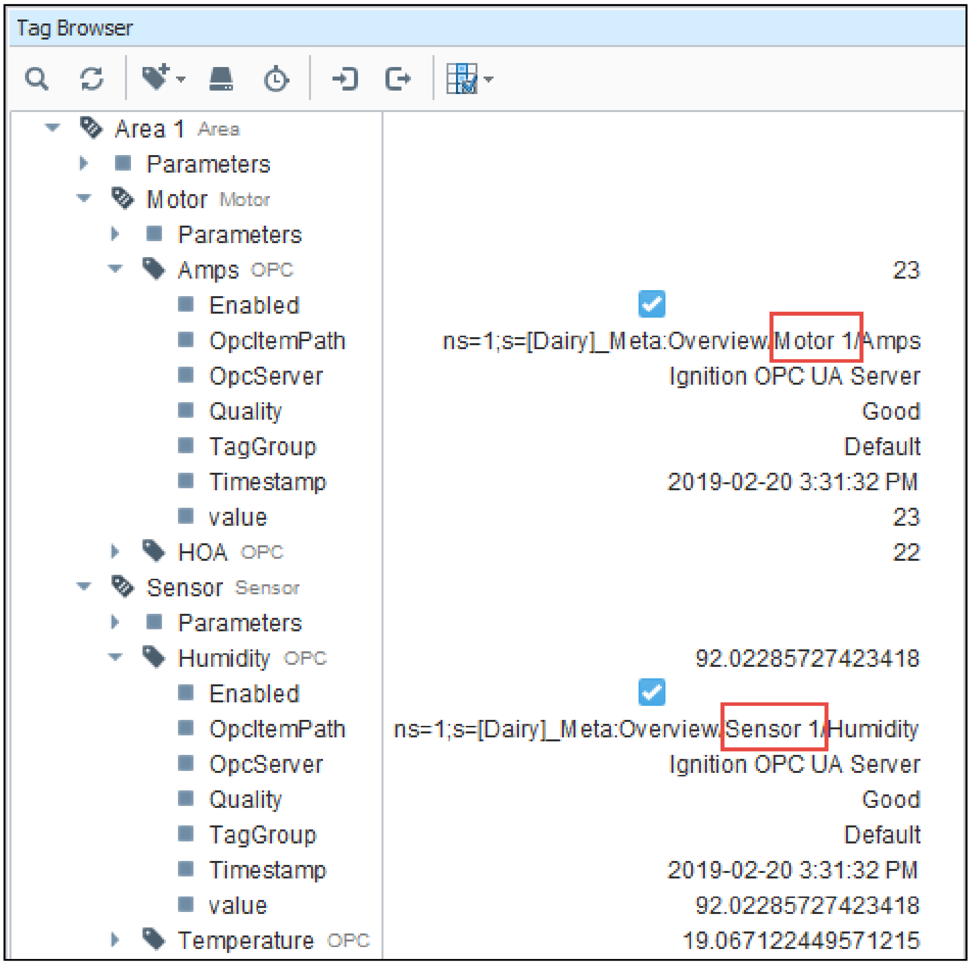
3. From the **Tag Browser**, you can see **Area 1** was created, and if expanded, it will show the two UDTs (Motor and Sensor) and that values are coming through.



Tag Browser

Area 1 Area		Area
Parameters		Document
MotorNumber	1	String
SensorNumber	1	String
Motor Motor		Motor
Parameters		Document
Amps OPC	52	String
HOA OPC	22	String
Sensor Sensor		Sensor
Parameters		Document
Humidity OPC	69.001...	String
Temperature OPC	24.308...	String

4. Check the **OPC Item Path** to verify that Motor is pointing to **Motor 1** and Sensor is pointing to **Sensor 1**.



Area 1 Area	
Parameters	
Motor Motor	
Parameters	
Amps OPC	23
Enabled	☑
OpcItemPath	ns=1;s=[Dairy]_Meta:Overview/Motor 1/Amps
OpcServer	Ignition OPC UA Server
Quality	Good
TagGroup	Default
Timestamp	2019-02-20 3:31:32 PM
value	23
HOA OPC	22
Sensor Sensor	
Parameters	
Humidity OPC	92.02285727423418
Enabled	☑
OpcItemPath	ns=1;s=[Dairy]_Meta:Overview/Sensor 1/Humidity
OpcServer	Ignition OPC UA Server
Quality	Good
TagGroup	Default
Timestamp	2019-02-20 3:31:32 PM
value	92.02285727423418
Temperature OPC	19.067122449571215

Related Topics ...

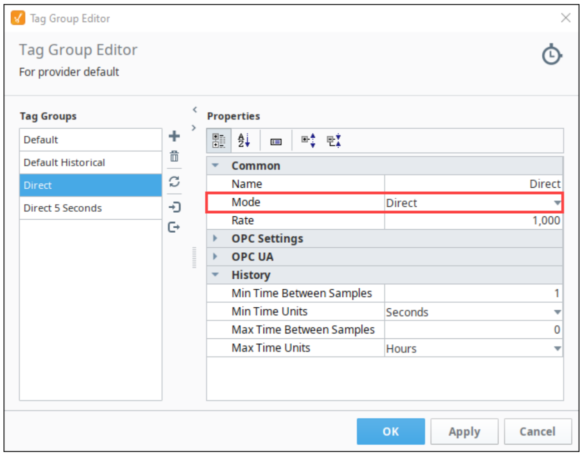
- UDT Definitions
- Data Type Parameters in Expressions

Tag Groups

What Is a Tag Group?

Tag Groups dictate the rate of execution of Tags, and therefore play a crucial role in the design of large and high-performance systems. It is beneficial to have multiple Tag Groups so you can set groups of Tags to run at different rates. Since it takes up computer resources to constantly fetch new Tag values, this frees up a lot of resources if many of the Tags are running slower. Some Tags you may wish to see updated at 250-500ms, while others may not be so crucial and may only need an update every 10-30 seconds.

Creating different Tag Groups allows you to organize your Tags into groups that subscribe (continually poll) at different rates. It is good practice to put some forethought and planning into the organization of your Tags and Tag Groups.



Common	
Name	Direct
Mode	Direct
Rate	1,000
OPC Settings	
OPC UA	
History	
Min Time Between Samples	1
Min Time Units	Seconds
Max Time Between Samples	0
Max Time Units	Hours

On this page

...

Tag Group Overview

Watch the Video

Tag Execution by Tag Group

Tags are executed by Tag Groups inside of a Tag Provider. In a typical system, there are a number of Tag Groups and one or more Tag Providers. Tag Providers can be either internal or external.

- **Internal provider** keeps the Tag configuration and execution in this Ignition Gateway.
- **Remote provider** allows this Ignition Gateway to view Tags from another Ignition Gateway.

Tags inside Internal Providers are available to other Ignition Gateways that use the **Remote** provider type.

> **The words Tag Group "types" and "modes" used interchangeability**
>
> When referring to Tag Groups, we often use the words "type" and "mode" interchangeably throughout this manual. Ignition has three Tag Group modes: Direct, Driven, and Leased. There can be different variations of these modes that can change their behavior. For example, the Driven Tag Group has "one-shot" and "any change" options that you can set, and we often refer to these as different "types" of Tag Groups as well. When you create a new Tag Group, you will select one of the three "modes" and any properties you need to build your Tag Group.

Types of Tag Groups

Tags are assigned to a Tag Group that determine how often they update. For example, how often an OPC value is polled from the PLC, how often an Expression Tag calculates its expression, and how often a Query Tag runs its query. It's easy to create Tag Groups in Ignition for just about any scenario you can think of. Tag Groups are extremely powerful and flexible, and you can create them based entirely on your individual business requirements.

There are three different Tag Group modes in Ignition that you can use to build your own, each with a different options:

- **Direct** - Executes at a fixed rate (10 second, 250ms, etc). Every Tag that uses a Direct Tag Group will poll the PLC at the **Rate** setting at all times (24x7x365). Ignition provides a Direct Tag Group to help you get started, but you can change them however you'd like. It is recommended to adjust the "Default" Tag Group to a slower rate as all new Tags are automatically added to this Tag Group.

- **Driven** - Executes fast or slow based on a condition. The rate of the Driven Tag Group is determined by a condition based on the value of the selected **Driving Expression** which can be a Tag or an Expression.This condition is a simple comparison between a driving Tag's value and a static number (equal to, greater than, etc.). While the condition is true, the Tag Group executes at the **Leased/Driven Rate** (fast rate). If false, it runs at the **Rate** (slow). (It's useful to keep in mind that the driving Tag can also be an Expression Tag that performs complex calculations and references other Tags. In this way, it's possible to create robust Tag Group triggering).

    - **Driven One-shot** - If you want to read Tag values only once on certain conditions (instead of polling), there are two properties that change the behavior of the Driven mode: the **One-shot** property, and the **Any Change** Operator option. Using either of these, the Tag Group does not run at either the fast or slow rate. Instead, it will be triggered once each time your condition is met. **Any Change** will execute each time the driving Tag value changes, and **one-shot** will execute once when the comparison condition is true and not again until the condition becomes false and subsequently true.

- **Leased** - Executes as on-demand polling. It works similar to the Driven Tag Group except there is no driving Tag. Instead, the driving mechanism is whether a Tag is being used on an open window or session. That is, if a user is looking at a Tag, it runs at the fast rate, and if the Tag is not being shown anywhere, it is moved to the slow rate.

Historical Tag Groups

Historical Tag Groups are simply standard Tag Groups used by Tags to store history. By using separate Tag Groups for status and history, it's possible to maintain a Tag's status at a fast rate, without storing large amounts of history unnecessarily.

Despite the fact that there is not a technical differentiation between standard and historical Tag Groups, it is recommended that you create separate Tag Groups for each purpose and name them in a manner that indicates their usage. It is common to modify Tag Groups in order to affect a large number of Tags, and without a consistent distinction it may be possible to affect Tag execution in unexpected ways.

> ⚠ When using Tag Groups with both the Rate (slow rate) and Leased/Drive (fast rate), a **0** poll rate can be specified. This means the Tags in this Tag Group will not poll at all. Do not set a **0** poll rate on Tags that are storing history or need other regular updates.

Driven Tag Group

Watch the Video

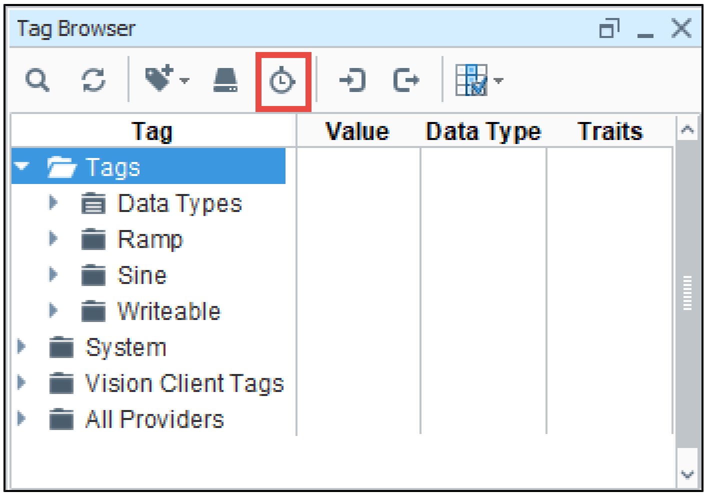
Leased Tag Group

Watch the Video

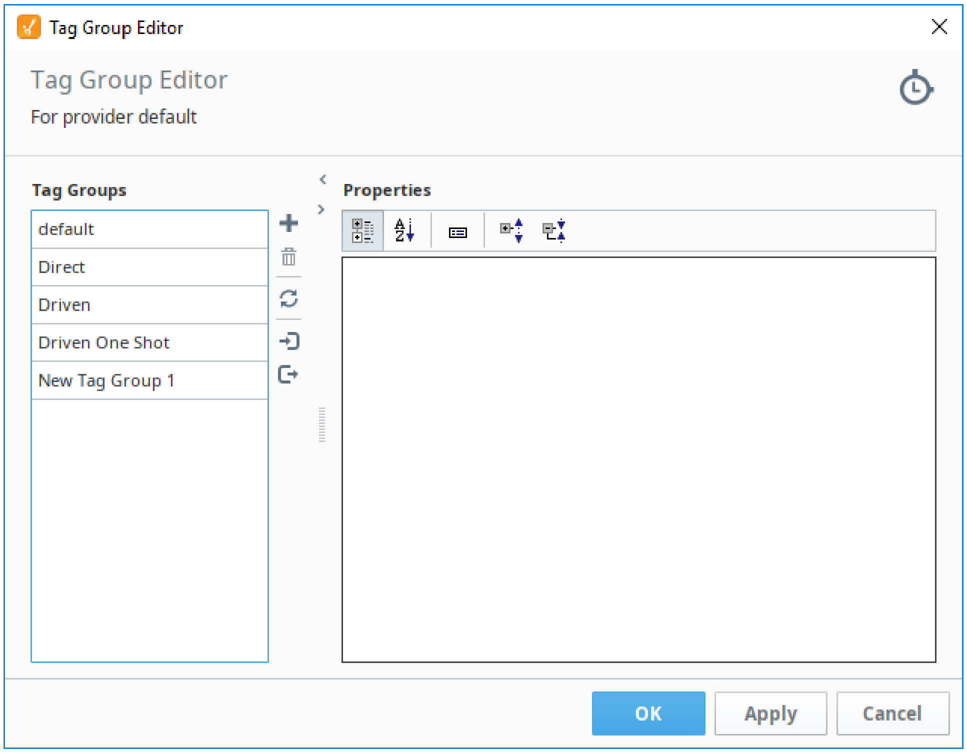
Adding and Editing Tag Groups

Adding and editing Tag Groups is easy in the Designer once you understand how the different Tag Group modes work. It's just a matter of choosing which Tag Group mode you want to use for your Tag, and entering the properties for your Tag Group.

1. In the Tag Browser, click on the **Tag Group** ⏱ icon to open the Tag Group Editor window.



2. A list of already configured Tag Groups appear on the left side of the window and configuration settings on the right. To add a Tag Group, click the **Add** ✚ icon. (Alternatively, you can click on an existing Tag Group to edit it.)



3. To create a new Tag Group, specify the **Name** of the Tag Group, and select a **Mode**: Direct, Driven, or Leased.

4. Each mode will have slightly different settings that will need to be configured. Enter the applicable settings to the type of Tag Group you're using.
To see the properties for each Tag Group mode and how to set each of them up, refer to Direct, Driven, Leased, and One-shot pages.

For descriptions of the Tag Group properties, refer to the properties table on each Tag Group page: Direct Mode, Driven Mode, Leased Mode, and One Shot Tag Group.

Setting a Tag's Tag Group

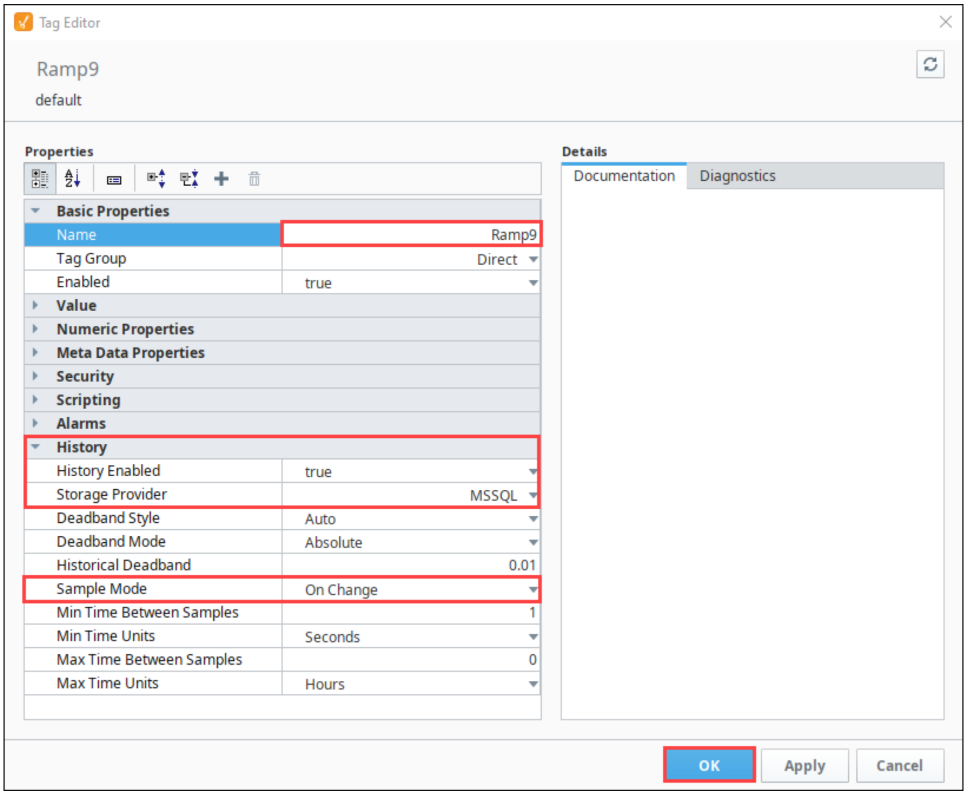
Each Tag in Ignition is assigned a Tag Group which dictates the polling rate and conditions on which the Tag will be evaluated. For example, the Tag Group will dictate how often that value is going to poll from the PLC if it's an OPC Tag, or how often the expression is going to run if it is an Expression Tag, or how often the value is going to query the database if it is a Query Tag. Whatever type of Tag you're using, you can set a Tag Group on that Tag. You can also specify both a Realtime Tag Group and a Historical Tag Group for each Tag.

1. In the **Tag Browser**, right-click on any Tag, and click the **Edit tag** ✎ icon. The **Tag Editor** window opens.

2. A list of Tag properties is displayed. Under **Basic Properties**, on the right side of the **Tag Group** property, click the dropdown list and choose **Direct** Mode.

3. The Tag also uses the Tag Group to determine how fast to log data for the Historian. Enabling History doesn't affect how fast the values get polled from the PLC, but affects how fast the data gets logged to a database. There are number of History properties so you might want to review them in Direct Mode Property table.

   a. Under History set **History Enabled** to 'true'.
   b. Select the **Storage Provider** from the dropdown list.
   c. Choose the **Sample Mode**.



Tag Editor

Ramp9

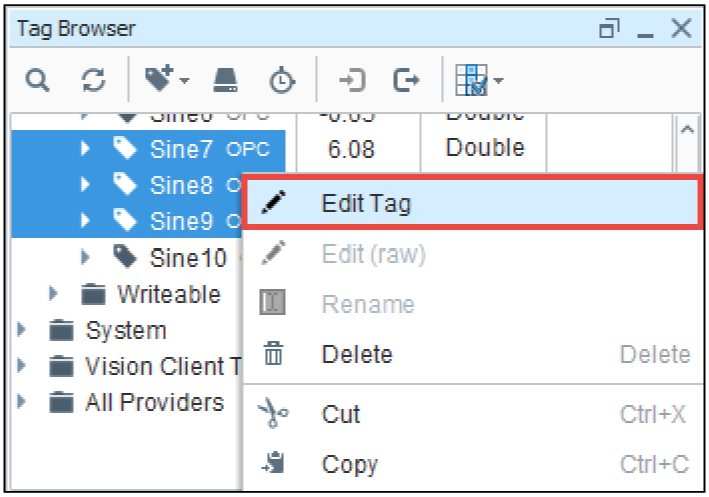
default

Properties

Basic Properties	
Name	Ramp9
Tag Group	Direct
Enabled	true
Value	
Numeric Properties	
Meta Data Properties	
Security	
Scripting	
Alarms	
History	
History Enabled	true
Storage Provider	MSSQL
Deadband Style	Auto
Deadband Mode	Absolute
Historical Deadband	0.01
Sample Mode	On Change
Min Time Between Samples	1
Min Time Units	Seconds
Max Time Between Samples	0
Max Time Units	Hours

4. Click **OK** to save it.
   You will immediately see your Tag updating at a different rate.

5. As you can see, setting Tag Groups on a Tag is easy. You can select multiple Tags in the Tag Browser by right clicking to edit the Tags. This opens the Tag Editor, and sets the Tag Group for all the selected Tags at the same time.



Tag Group Properties

The following table lists the properties for Tag Groups. Note that some properties are available only for specific modes.

Property	Description
Common	
Name	Unique name of the Tag Group.
Driven Mode	The rate of the Tag Group is based on the value of a driving Tag. The condition is a simple comparison between a Tag value and a number. If the condition is true, the Tag Group will execute at the fast rate. If false, it will run at the slow rate. There are two exceptions to this: the **Any Change** operator, and **One-shot** mode. Using either of these conditions will not run at a rate. Instead, it will be triggered by a change in the driving Tag's value. Keep in mind that the driving Tag can be an Expression Tag that performs complex calculations and references other Tags. In this way, it's possible to create robust Tag Group triggering.
Rate	Base update rate, specified in milliseconds, at which Tags will be executed. ⚠ If the rate is set to 0, the Tag Group will not be executed.
Leased/Driven Rate	Used by both the Leased and Driven Modes to determine when the Tag Group should run at the fast rate.
Driving Expression	The Tag Group executes based on the condition set on the Driving Expression: Tag or Expression. (Driven mode only.)
Driving Comparison	How the Comparison Value property should be compared to the Driving Tag's value. If the comparison is true, then the Fast Rate will be used by the Tag Group, otherwise, the Slow Rate will be used. The **Any Change** operator works differently than the other operators: The Tag Group will execute immediately whenever the driving Tag changes value. Using the Any Change operator means that the Tag Group no longer uses the Slow Rate or Fast Rate properties. (Driven mode only.)
Comparison Value	Used by the Driving Comparison property to determine if the Tag Group should execute at the slow or fast rate. (Driven mode only.)
One Shot	**One-shot** will execute once when the comparison condition is true, and not again until the condition becomes false, and subsequently true. (Driven mode only.)
OPC Settings	
Data Mode	This mode dictates how OPC values are obtained. The default mode, **Subscribed**, is preferred because it is more efficient than a read. **Subscribed** All OPC Tags in the Tag Group will be subscribed according to the Tag Group rate. Values will come in asynchronously as they change. **Polled** Tags will not be subscribed, but will instead be synchronously read each time the Tag Group executes. This operation is less efficient, but allows more precise control over when values are obtained. This mode is particularly useful when collecting data over a slow or expensive connection for display. When combined with the **one-shot** execution mode above, and a static Tag tied to a momentary button, it's easy to create a manual refresh button on a screen that pulls data on-demand.
Read After Write	When enabled, a read request will be sent immediately after a write request. This means that the value on the Tag will be updated much quicker to reflect the latest written value. Enabling this property is less efficient as a single write to a Tag becomes two separate requests. This is especially helpful with slower Tag Groups as the Tags will show the latest value quicker than the normal execution would allow.
Optimistic Writes	Optimistic Writes are only valid on OPC Tags. Optimistic Writes set a newly written Tag value in Ignition before receiving confirmation of the write from the PLC. This helps the operators see their newly entered value right away and is useful if you have slow a Tag Group rate. A faster rate (1 second or quicker) will have less need to turn on Optimistic Writes. If enabled, written values will be applied to the Tag in Ignition immediately. Normally, the system must receive confirmation that a write was successful from the device before the Tag in Ignition's value would change. The Optimistic Writes property changes

	the behavior by assuming the write went through until the next read value or subscription update proves otherwise. Enabling this will make writes appear to execute much quicker. Works in conjunction with the **OPC Optimistic Write Timeout** property below. If the Tag in Ignition does not receive confirmation that the new write was successful within the timeout, the Tag will change back to the last known value. While in an ambiguous state, the Tag with have a quality of "**Good (Provisional)**". This setting can be paired with the **OPC Read After Write**: the Ignition Tag will assume the newly written value, while an asynchronous read request is quickly sent out to confirm the write went through. While the write is pending, values received from subscription activities will override the current value. Assuming an initial value of 0, if a write of 10 is applied to the Ignition Tag, then the Tag will show a value of 10 until the system can confirm the new value. If a subscription update then returns a value of 5, the Ignition Tag will change to 5.
Optimistic Write Timeout (MS)	The timeout period for Optimistic Writes. A value of 0 effectively disables the fallback functionality: the new value is maintained on the Tag until the next read or subscription activity.
OPC UA	
Publishing Interval (ms)	The rate at which data is delivered to the OPC-UA client. A value of -1 means automatic, allowing the OPC-UA client to determine the rate.
Queue Size	The OPC-UA specifications states that in cases where the sampling interval (the rate as which the server checks the data source for changes) is faster than the publishing interval (rate at which the the data is delivered to the client), the samples may be queued or batched together before publishing. This setting determines the maximum size of that queue. When the maximum is reached and a publish has not yet occurred, oldest samples are dropped first. Currently, there are not many features in Ignition that utilize multiple entries in the queue, but 3rd party OPC-UA clients may be able to take advantage of this setting. Note that values on Ignition Tags will only ever show one value at a time, regardless of what this property is set to. (If the value is 0, nothing will be queued.) Support for this feature is dictated by the driver. • DNP3 - See the DNP3 page for details on how buffered data and Sequence of Events works
History	
Min Time Between Samples	Minimum time between samples (integer).
Min Time Units	Minimum time in units is defined as: Milliseconds, Seconds, Minutes, Hours, Days, Weeks, Months, and Years.
Max Time Between Samples	Maximum time between samples (integer).
Max Time Units	Maximum time in units is defined as: Milliseconds, Seconds, Minutes, Hours, Days, Weeks, Months, and Years.

Related Topics ...

- Creating Tags
- Direct Mode
- Driven Mode
- Leased Mode
- One Shot Tag Group

In This Section ...

Direct Mode

The Direct Mode executes at a fixed rate which is defined by the slow polling rate setting. Every Tag that uses the Direct Tag Group will poll the PLC at the Rate setting at all times (24x7x365).

For a complete list of Tag Group properties, see Tag Groups.

Direct Scan Class

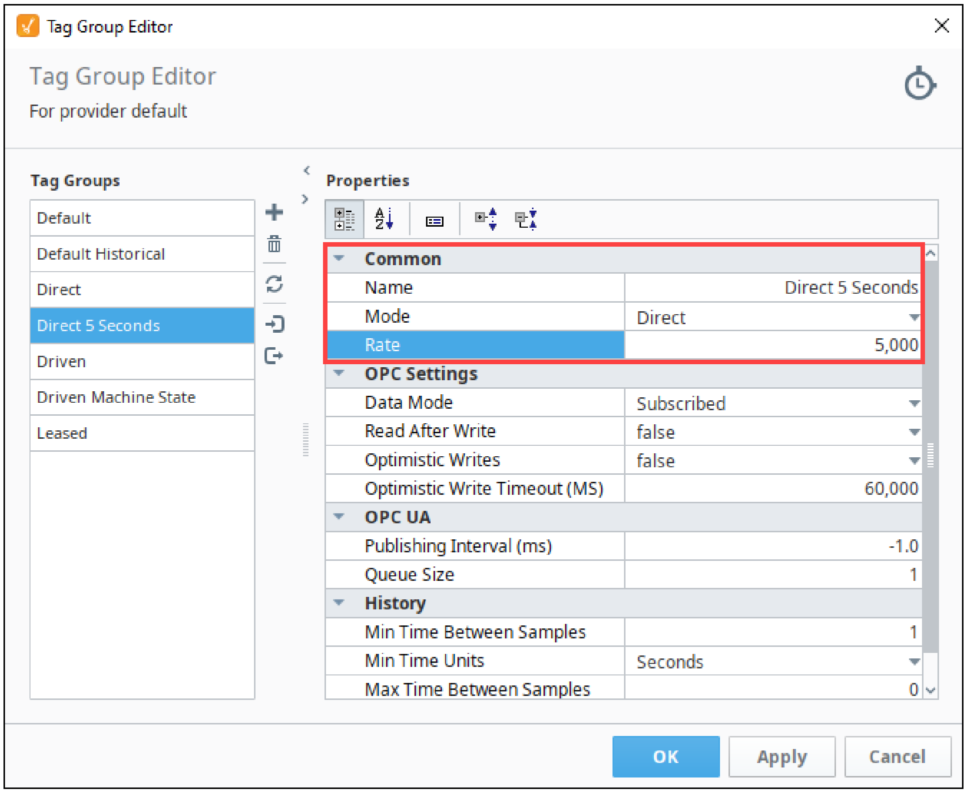
Watch the Video

Add a Direct Tag Group

1. In the Tag Browser, click on the **Tag Groups**⏱ icon to open the **Tag Group Editor**.

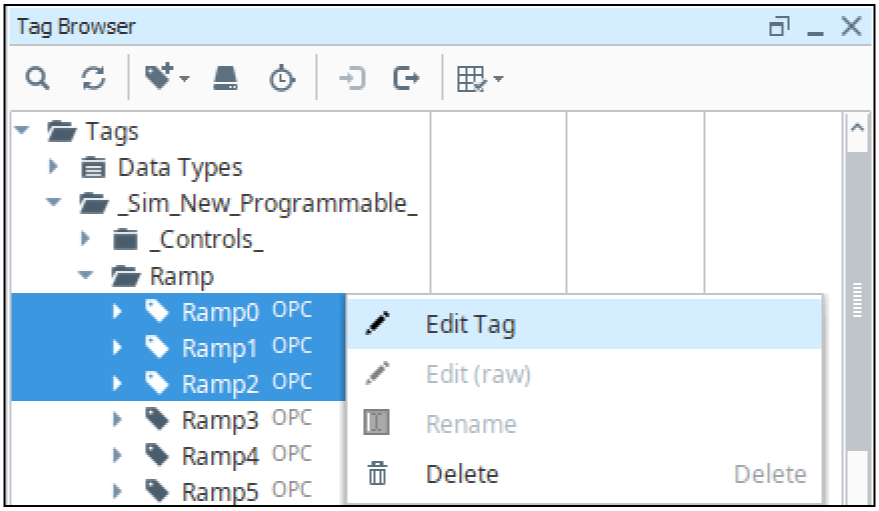
2. On the bottom left side, click the **Add** ✚ icon to create a new Tag Group, and enter the values for the following properties:
    a. Name - Enter a unique name for the Tag Group: **Direct 5 Seconds**
    b. Select the Mode: **Direct**
    c. Enter Rate: **5,000**

3. Click **OK** to save.

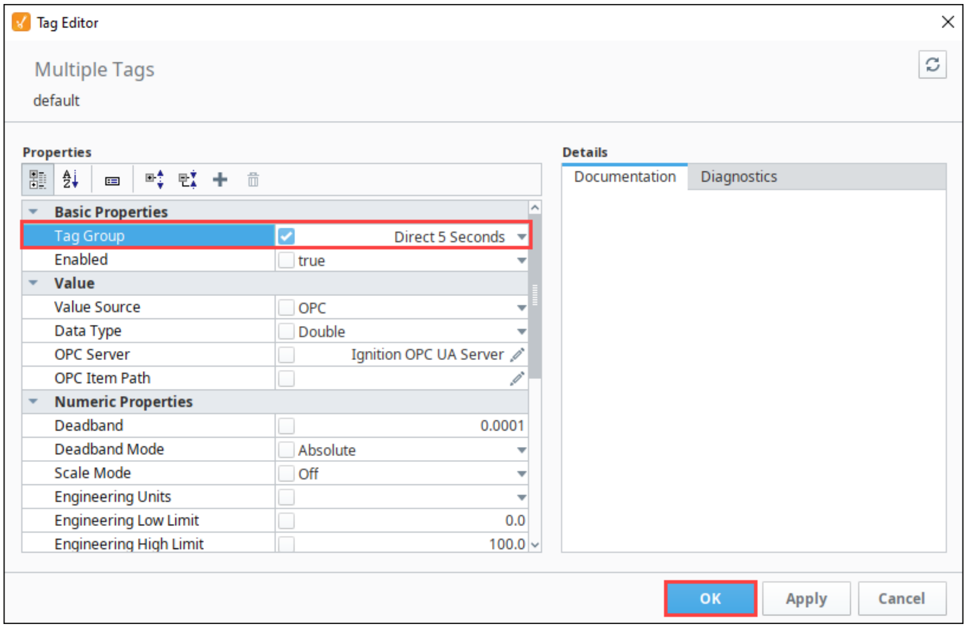


Common	
Name	Direct 5 Seconds
Mode	Direct
Rate	5,000
OPC Settings	
Data Mode	Subscribed
Read After Write	false
Optimistic Writes	false
Optimistic Write Timeout (MS)	60,000
OPC UA	
Publishing Interval (ms)	-1.0
Queue Size	1
History	
Min Time Between Samples	1
Min Time Units	Seconds
Max Time Between Samples	0

4. Now that you have your Tag Group created, let's add multiple Tags to the Tag Group. Go to your **Tag Browser**, find some Tags you want to add to the Tag Group. This example uses several Ramp Tags. Right click on the selected Tags, and click on **Edit Tag** icon.



5. This opens the **Tag Editor** window. It also shows you that you have multiple Tags selected. Select the **Direct 5 Seconds** Tag Group from the dropdown list.
6. Click **OK**.



Multiple Tags
default

Basic Properties

Tag Group	☑	Direct 5 Seconds
Enabled	□	true

Value

Value Source	□	OPC
Data Type	□	Double
OPC Server	□	Ignition OPC UA Server
OPC Item Path	□	

Numeric Properties

Deadband	□	0.0001
Deadband Mode	□	Absolute
Scale Mode	□	Off
Engineering Units	□	
Engineering Low Limit	□	0.0
Engineering High Limit	□	100.0

You created a new Direct 5 Second Tag Group and added your Tags. Just make sure you want to poll the 5 second values all the time (24/7) when you use the Direct 5 Seconds Tag Group.

Related Topics ...

- Driven Mode
- Tag Groups

Driven Mode

Driven Tag Groups

The Driven Tag Group is a very flexible and powerful Tag mode type in Ignition. It allows Tags to run at a fast or slow rate based on a Tag and some condition. This allows for better system performance, while still giving you high resolution data when needed.

Driven Tag Groups set the polling rate dynamically for Tags based on a comparison between the driving Tag's value and a value. In simple cases, when the logic comparison is true, the fast rate (Leased/Driven Rate) applies, otherwise, the slow rate (Rate) applies. There are two exceptions to this: the Any Change operator (Driving Comparison) and the one-shot mode. Using either of these exceptions, the Tag Group does not run at a rate. 'Any change' will execute once each time the value changes, and 'one-shot' will execute once when the One Shot comparison condition is true, and not again until the condition becomes false and subsequently true. This is not the same as setting the Rate to 0, which means the values should not be polled at all. Having a Rate of 0 can be useful (such as when a line is not running), but make sure your Tags are still polling if you are storing history or have alarms set up.

A driving Tag can be an OPC Tag, or an Expression that performs complex calculations and references other Tags. It's possible to create robust Tag Group triggering using an Expression.

> **Beware the Driving Tag's Tag Group**
>
> Make sure the driving Tag is never using the Tag Group it is driving. The driving Tag should be set to a separate Tag Group and almost always to a Direct type. Changes to the driving Tag will be recognized by the system faster, and the change in poll rate on the Driven Tag Group will occur faster.
>
> If the Driving Tag is using a Tag Group that has a 0 rate, it will never have it's value checked, and the Tag Group it is driving will never update.

Shown below are three common ways to run a Driven Tag Group: by **Machine State**, **Manual Trigger**, and **Time of Day.** Each has a description of the Tag Group, how to set it up, as well as how to setup the driving Tag.

For a complete list of Tag Group properties, see Tag Groups.

On this page ...

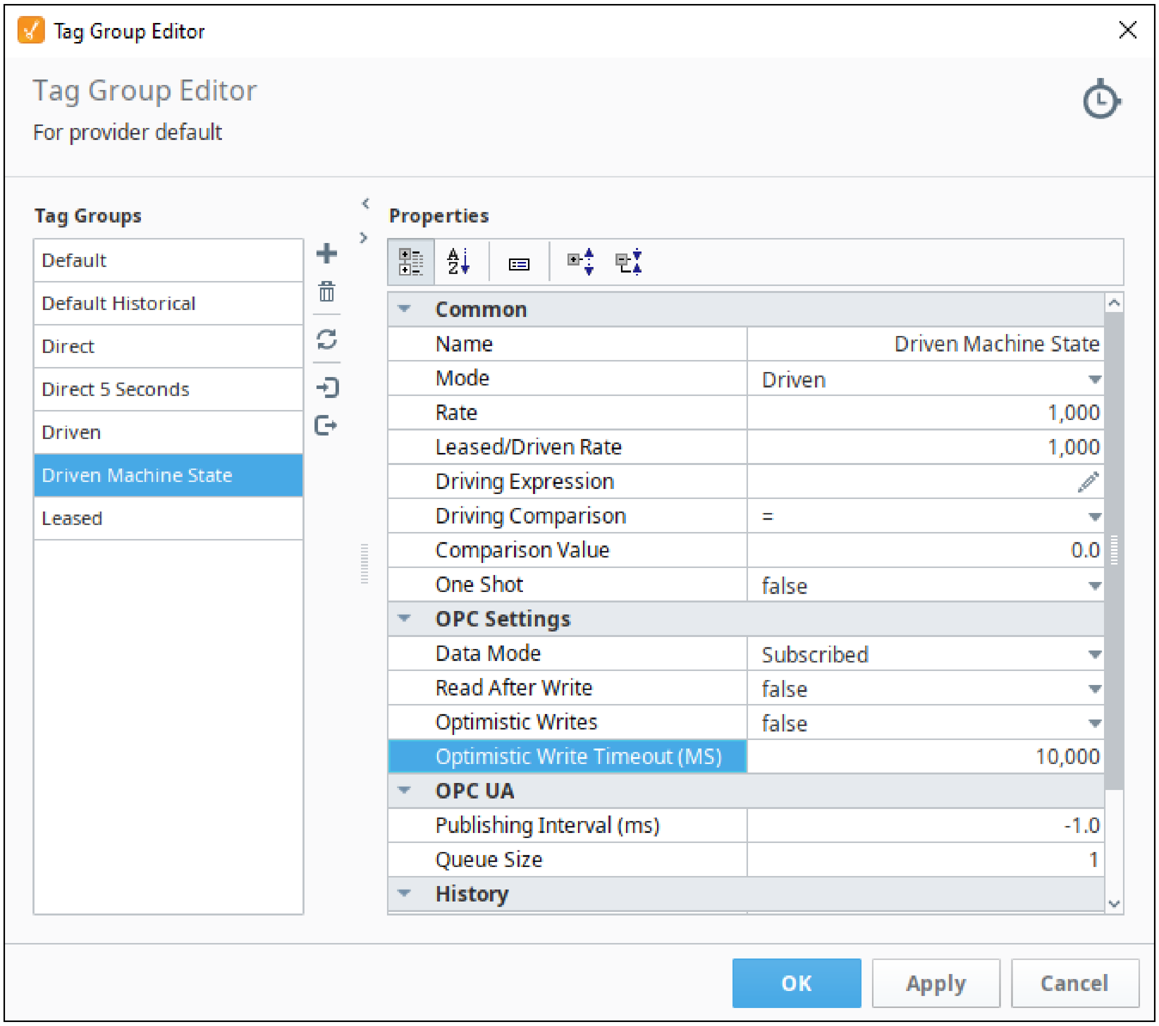
INDUCTIVE UNIVERSITY

Driven Tag Group

Watch the Video

Driven Tag Group - Machine State

Driven Tag Groups that are based on a machine can be very important when you only want to poll values differently based on machine state. A machine state Tag Group is when the Tags are polled at one rate with the machine ON, and a different rate when the machine is OFF. You can easily see the rate change in the the Tag Browser using this type of condition.



Tag Groups:

- Default
- Default Historical
- Direct
- Direct 5 Seconds
- Driven
- Driven Machine State
- Leased

Common	
Name	Driven Machine State
Mode	Driven
Rate	1,000
Leased/Driven Rate	1,000
Driving Expression	
Driving Comparison	=
Comparison Value	0.0
One Shot	false
OPC Settings	
Data Mode	Subscribed
Read After Write	false
Optimistic Writes	false
Optimistic Write Timeout (MS)	10,000
OPC UA	
Publishing Interval (ms)	-1.0
Queue Size	1
History	

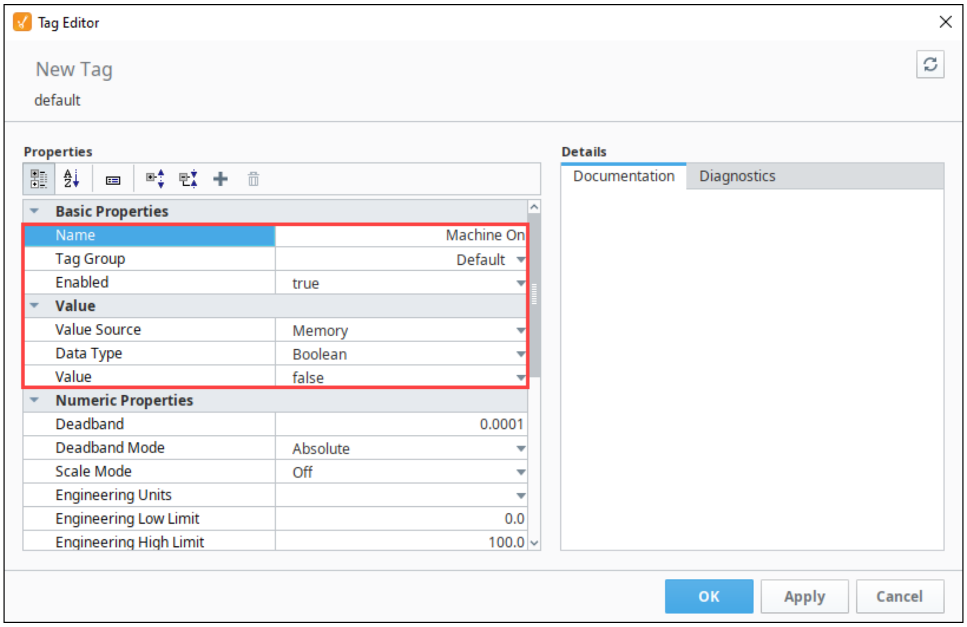
Create a Machine State Tag

First, you need to create a driving Tag if you don't already have one.

1. In the **Tag Browser**, right-click on the **Tags** folder, then go to **New Tag > Memory Tag** to create a memory Tag. The Tag Editor window is displayed.

2. In **Tag Editor**, enter the following:
   Name: **Machine On**
   Tag Group: **Default**
   Enabled: **true**
   Data Type: **Boolean**
   Value: **false**

3. Click **OK** to add the **Machine On** Tag to the Tag Browser.

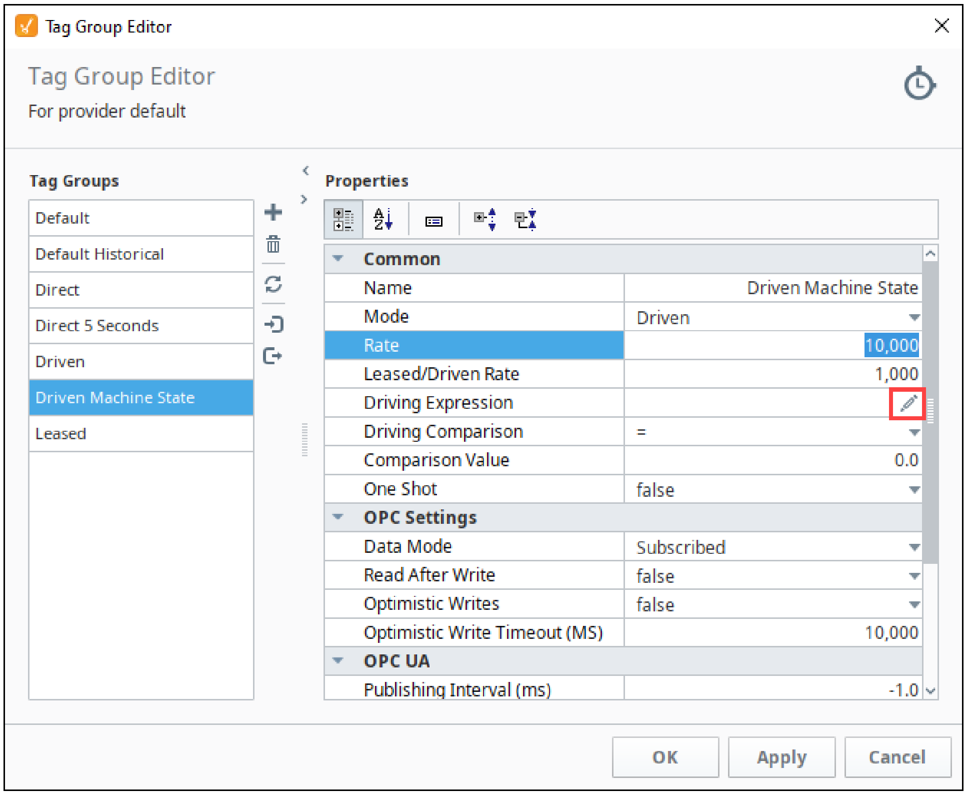


Basic Properties	
Name	Machine On
Tag Group	Default
Enabled	true
Value	
Value Source	Memory
Data Type	Boolean
Value	false
Numeric Properties	
Deadband	0.0001
Deadband Mode	Absolute
Scale Mode	Off
Engineering Units	
Engineering Low Limit	0.0
Engineering High Limit	100.0

Add a Driven Tag Group Based on the Machine State

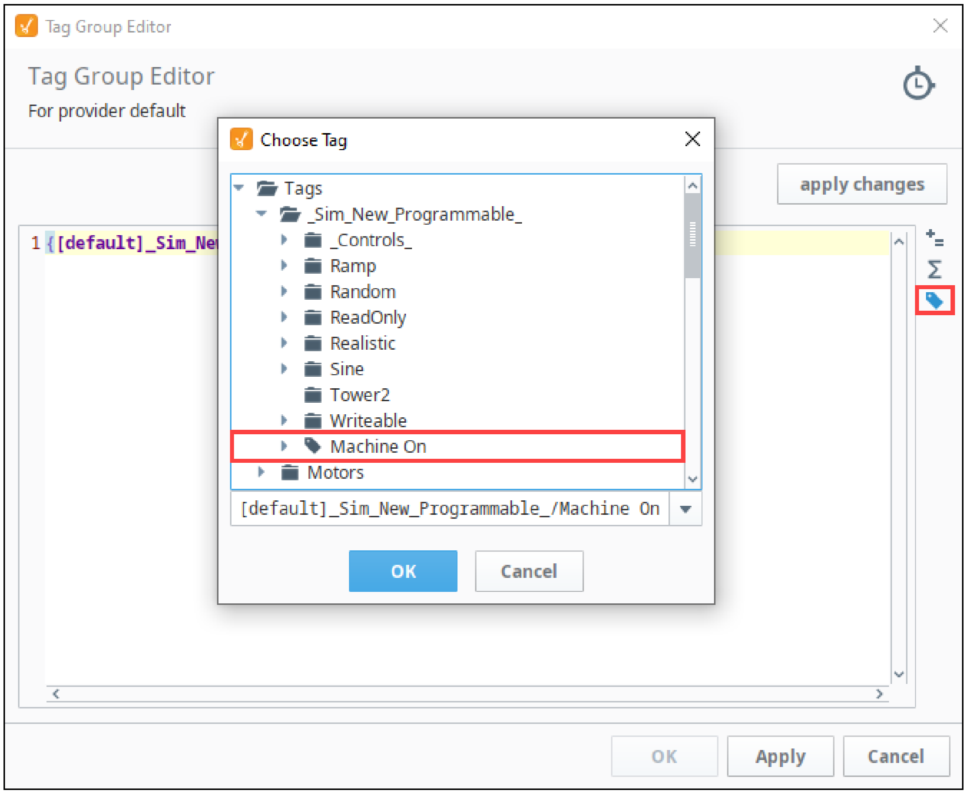
Once you have your driving Tag created, add a Driven Tag Group that updates the Tags based on when the machine is ON.

1. In the **Tag Browser**, click on the **Edit Tag Groups** ⏱ icon to open the Tag Group Editor.
2. To create a new Tag Group, click **Add** ✚ icon.
3. Enter the name of the Tag Group. For example, you can name it **Driven Machine State**, and set the **Mode** to **Driven**.
4. Set the **Rate** to 10,000 (10 seconds) and the **Leased/Driven Rate** to 1000 (1 second).
5. This Tag Group executes based on the condition you set on the **Driving Expression**. Set the **Driving Expression** by clicking the **Edit** ✎ icon.



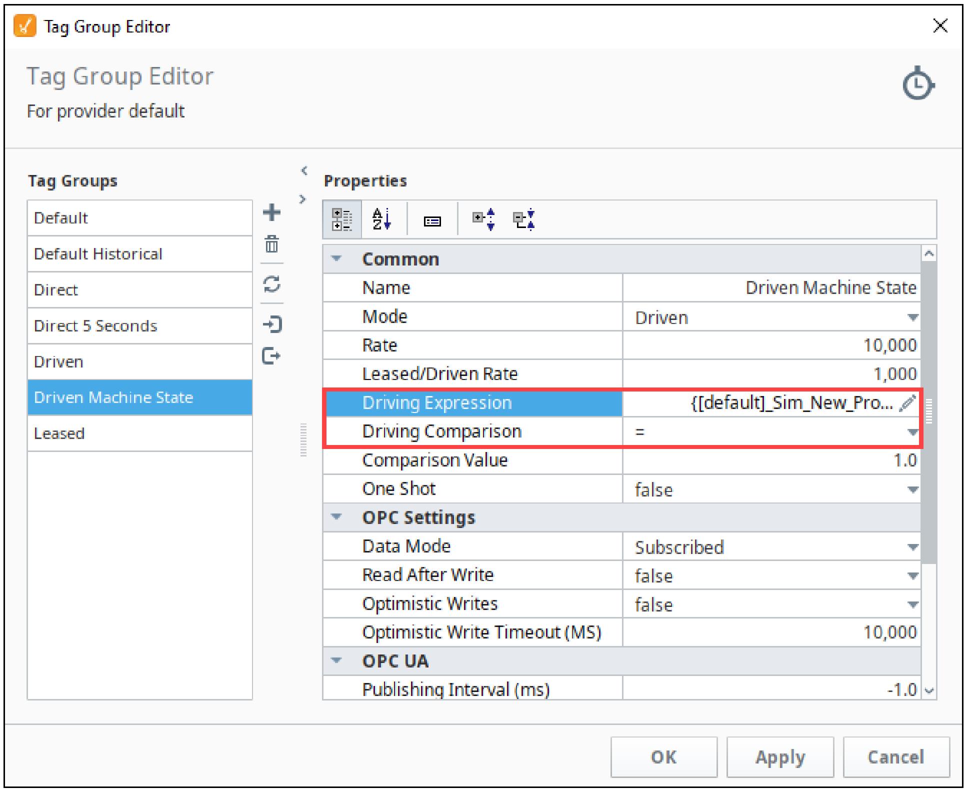
Common	
Name	Driven Machine State
Mode	Driven
Rate	10,000
Leased/Driven Rate	1,000
Driving Expression	✎
Driving Comparison	=
Comparison Value	0.0
One Shot	false
OPC Settings	
Data Mode	Subscribed
Read After Write	false
Optimistic Writes	false
Optimistic Write Timeout (MS)	10,000
OPC UA	
Publishing Interval (ms)	-1.0

6. This Tag Group executes based on the condition you set on the **Driving Expression**. Set the **Driving Expression** by clicking the **Edit** ✎ icon.
7. Click the **Tag** 🏷 icon and choose the Tag you want to set the condition on (i.e., Machine On).
8. Click **OK** on the popup and **apply changes** in the upper right of the Tag Group Editor.



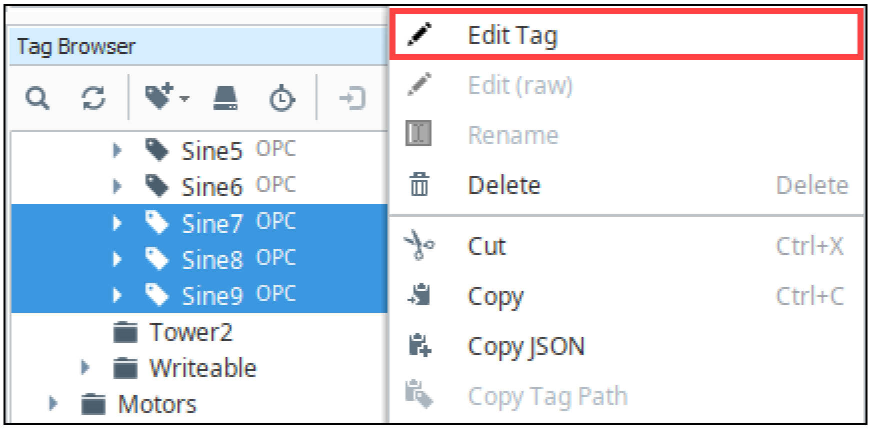
[default]_Sim_New_Programmable_/Machine On

9. While still in the Tag Group Editor, set the following values:
   a. Driving Comparison**: =**
   b. Comparison Value: **1**
10. Click **OK** in the lower right of the Tag Group Editor.

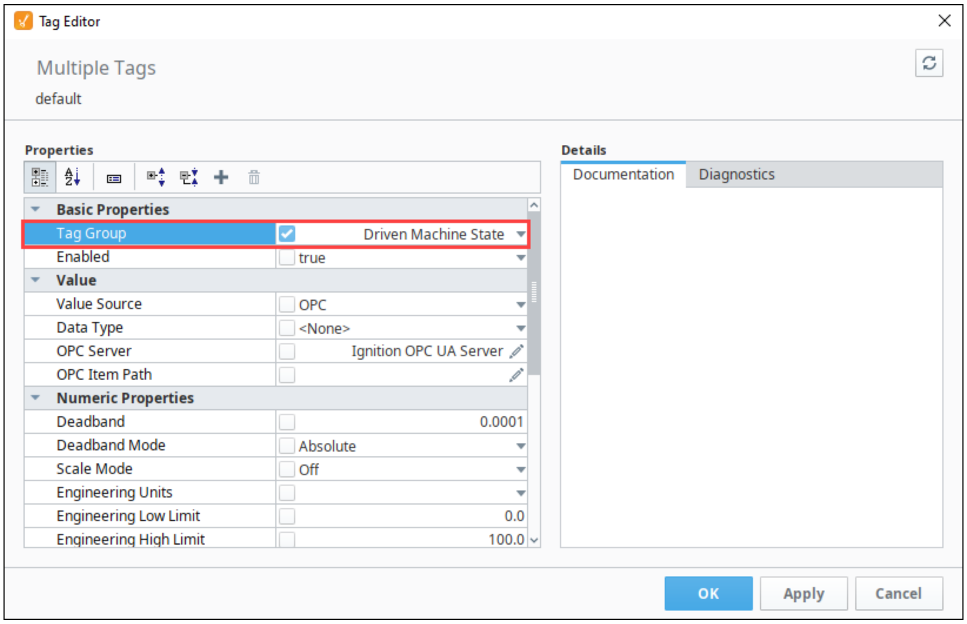


Common	
Name	Driven Machine State
Mode	Driven
Rate	10,000
Leased/Driven Rate	1,000
Driving Expression	{[default]_Sim_New_Pro...
Driving Comparison	=
Comparison Value	1.0
One Shot	false
OPC Settings	
Data Mode	Subscribed
Read After Write	false
Optimistic Writes	false
Optimistic Write Timeout (MS)	10,000
OPC UA	
Publishing Interval (ms)	-1.0

11. When the machine is on, it will use the Leased/Driven Rate (1 second). When the machine is off, it will use the Rate (10 seconds)
12. Next, select all the Tags that you want to use for the Drive Machine State Tag Group. This example uses **Sine7, Sine8**, and **Sine9** T ags.
13. Right click on the selected Tags and choose **Edit Tag**.

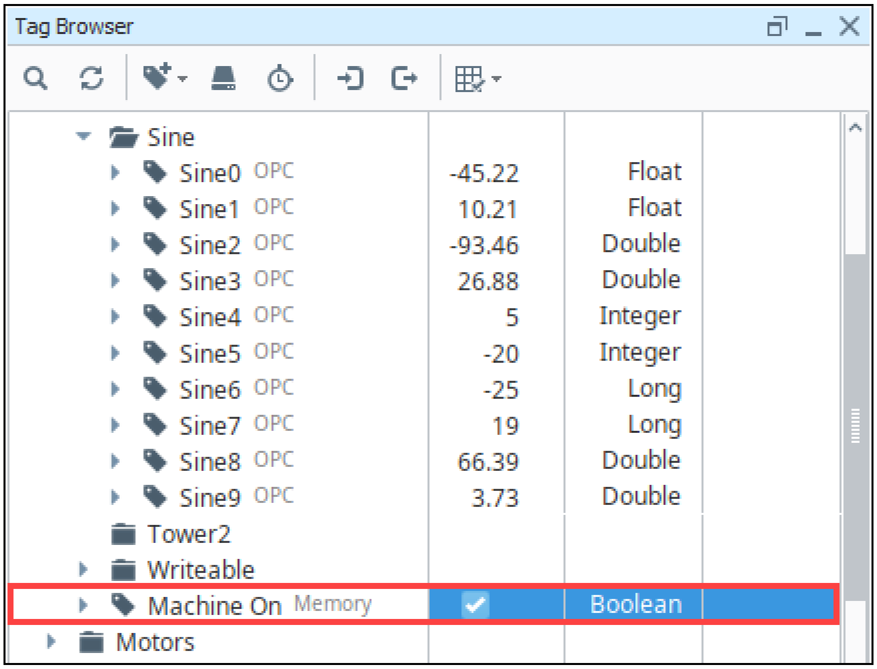


14. The Tag Editor will open showing you are editing multiple Tags. From the Tag Group dropdown list, choose **Drive Machine State**, and click **OK.**



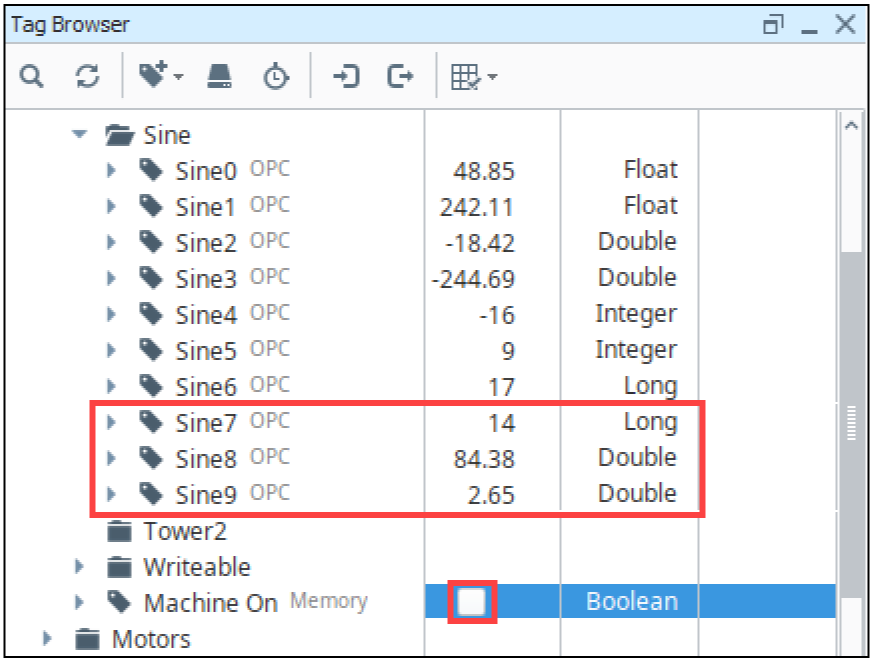
Properties		
Basic Properties		
Tag Group	☑	Driven Machine State
Enabled	□	true
Value		
Value Source	□	OPC
Data Type	□	<None>
OPC Server	□	Ignition OPC UA Server
OPC Item Path	□	
Numeric Properties		
Deadband	□	0.0001
Deadband Mode	□	Absolute
Scale Mode	□	Off
Engineering Units	□	
Engineering Low Limit	□	0.0
Engineering High Limit	□	100.0

15. Now let's turn the machine on. In the Tag Browser, mark the checkbox next to the **Machine On** Tag that we created earlier. Now, when the machine is on, polling will be a at the 1 second rate for the Sine Tags. When the machine is off, polling will be at the 10 second rate for the Sine Tags. Test it by toggling the Machine On Tag on and off.



Sine		
Sine0 OPC	-45.22	Float
Sine1 OPC	10.21	Float
Sine2 OPC	-93.46	Double
Sine3 OPC	26.88	Double
Sine4 OPC	5	Integer
Sine5 OPC	-20	Integer
Sine6 OPC	-25	Long
Sine7 OPC	19	Long
Sine8 OPC	66.39	Double
Sine9 OPC	3.73	Double
Tower2		
Writeable		
Machine On Memory	☑	Boolean
Motors		

You can also set the **Rate** (slow rate) to a value of 0, which means polling is stopped when the condition is false. So for example, when the machine is off, you will see the Tags stay at the last known value.



Tag Browser

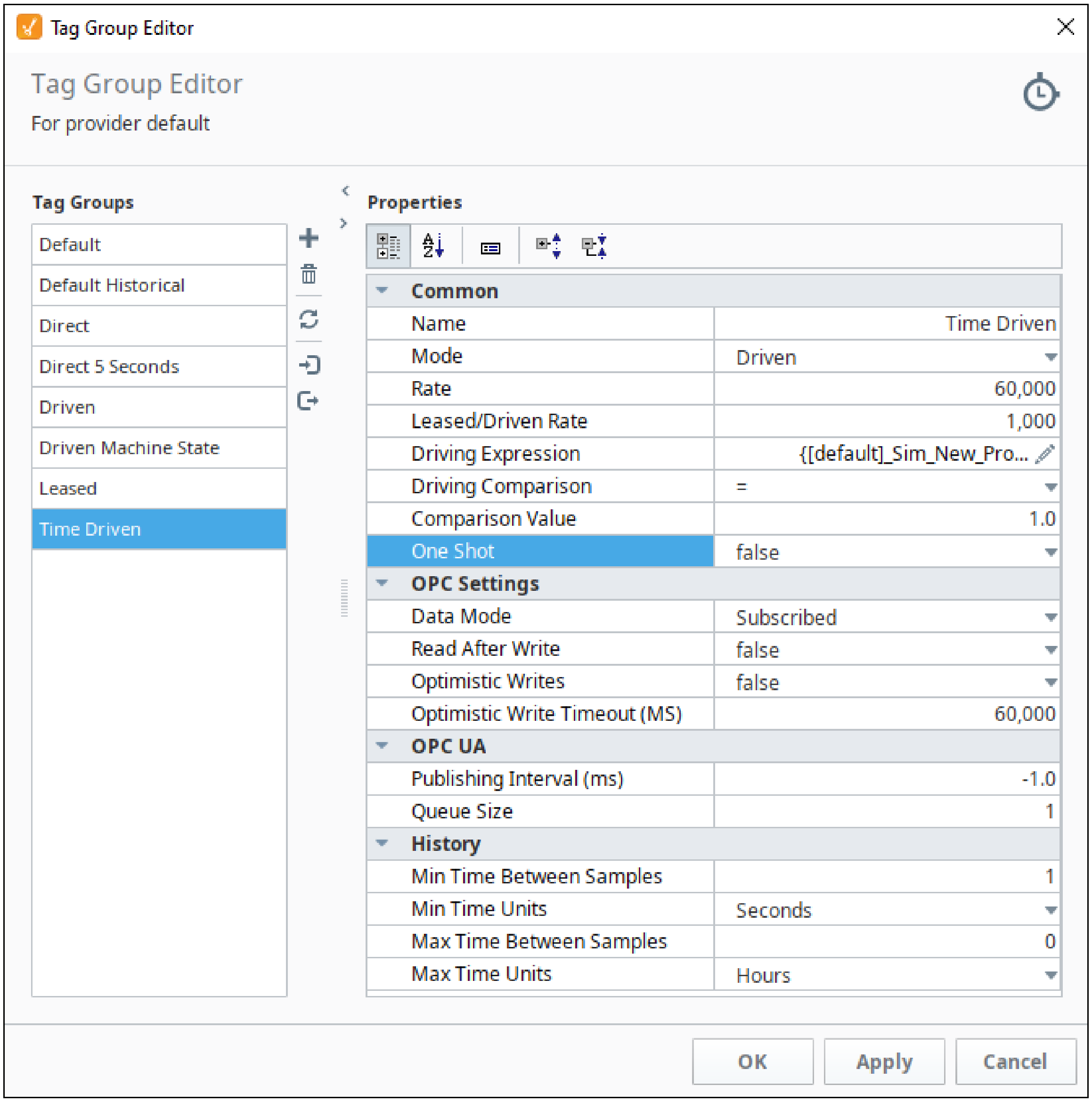
Sine		
Sine0 OPC	48.85	Float
Sine1 OPC	242.11	Float
Sine2 OPC	-18.42	Double
Sine3 OPC	-244.69	Double
Sine4 OPC	-16	Integer
Sine5 OPC	9	Integer
Sine6 OPC	17	Long
Sine7 OPC	14	Long
Sine8 OPC	84.38	Double
Sine9 OPC	2.65	Double
Tower2		
Writeable		
Machine On Memory	☐	Boolean
Motors		

Once the machine is turned back on, the Tags resume polling.

Driven Tag Group - Time of Day

Driven Tag Groups can be used as the polling rate for Tags to trigger at different rates at different times of the day. They can also be used as a one-shot event at a specific time of the day. You can accomplish this by setting the condition of the Driven Tag Group to be true at certain times of the day, and false at other times.

Keep in mind, that if you use a Driven Tag Group based on the time of day with a 0 Slow Rate for history, the Tag will only store history during those specified times.



Tag Groups:

- Default
- Default Historical
- Direct
- Direct 5 Seconds
- Driven
- Driven Machine State
- Leased
- Time Driven

Property	Value
Common	
Name	Time Driven
Mode	Driven
Rate	60,000
Leased/Driven Rate	1,000
Driving Expression	{[default]_Sim_New_Pro...
Driving Comparison	=
Comparison Value	1.0
One Shot	false
OPC Settings	
Data Mode	Subscribed
Read After Write	false
Optimistic Writes	false
Optimistic Write Timeout (MS)	60,000
OPC UA	
Publishing Interval (ms)	-1.0
Queue Size	1
History	
Min Time Between Samples	1
Min Time Units	Seconds
Max Time Between Samples	0
Max Time Units	Hours

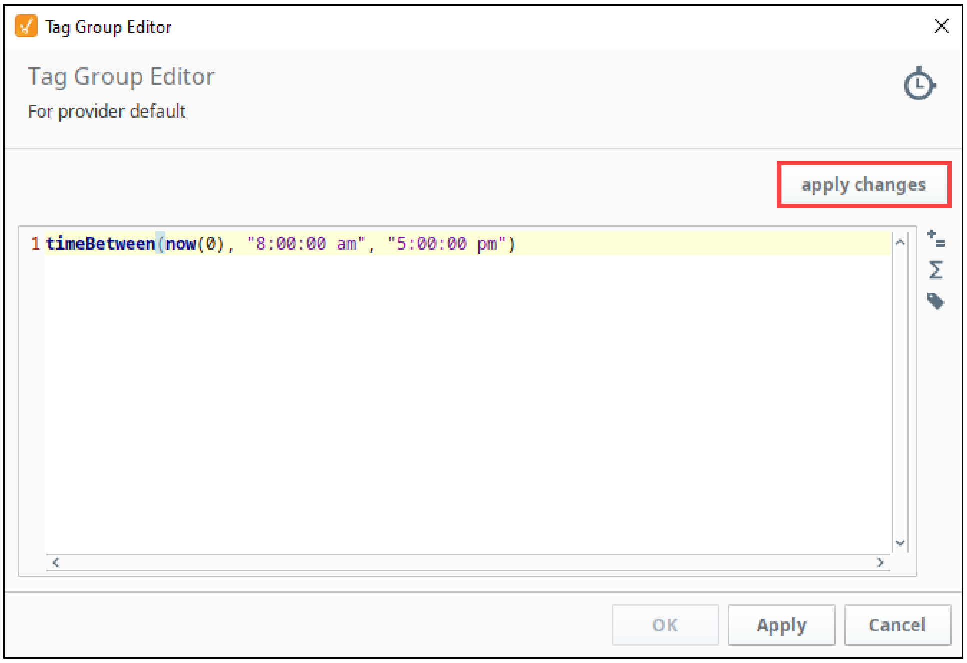
Add a Driven Tag Group Based on the Time of Day

Let's add a Driven Tag Group that updates the Tags based on a time of day. We will use an Expression to drive the Tag Group. You can use the functions that are in the expression language to poll the PLC during the hours of 8am to 5pm.

1. In the Tag Browser, click on the **Tag Groups** ⏱ icon to open the Tag Group Editor.
2. To create a new Tag Group, click on the **Add** ✚ icon.
3. Enter the name for the Tag Group. In this example, we named it **Time Driven**, and set the Mode to **Driven**. This Tag Group executes based on the condition of the **Driving Expression**. In this example, we'll use the Poll Time Tag we created above.
4. Set the **Rate** to 60,000ms so it polls at a slow rate, and the **Leased/Driven Rate** to 1,000ms so it polls at a faster rate.
5. Enter the **Driving Expression.** Click on the **Edit** ✎ icon, and copy and paste the following expression in the expression box, then click **Apply Changes**.

timeBetween and Machine On expression

```
timeBetween(now(0), "8:00:00 am", "5:00:00 pm")
```

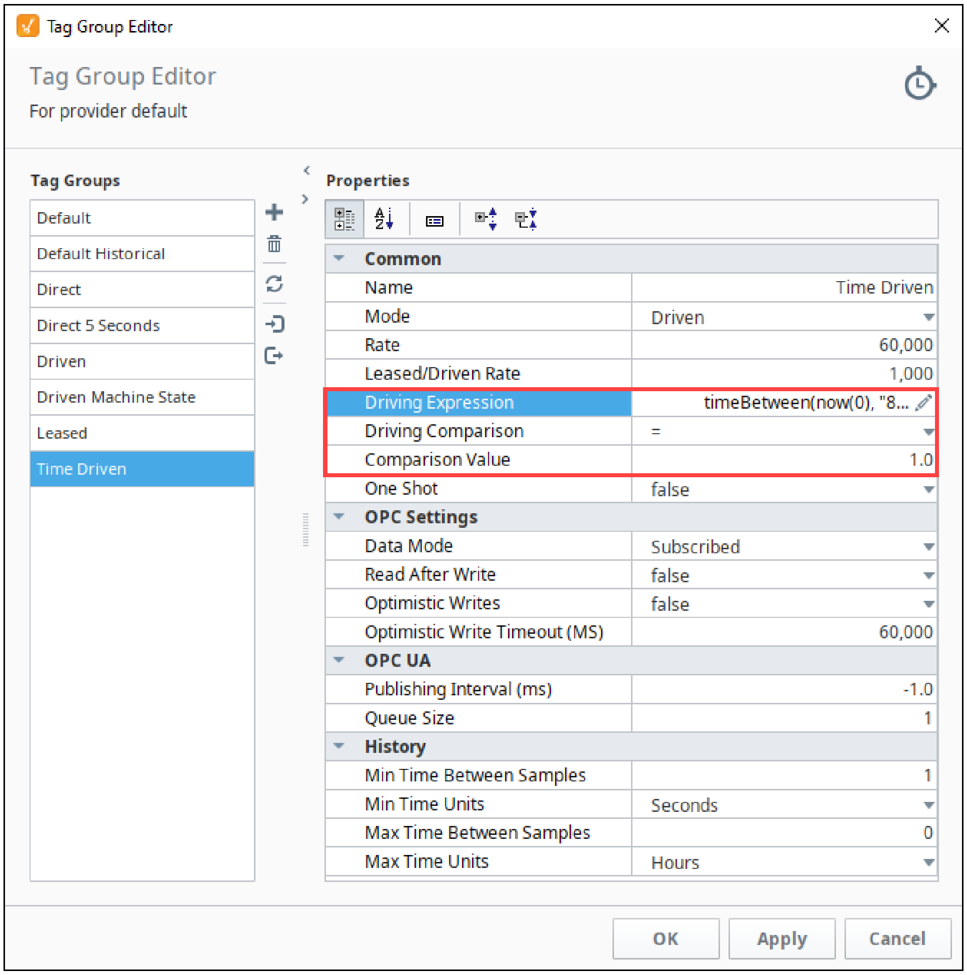


```
timeBetween(now(0), "8:00:00 am", "5:00:00 pm")
```

6. While still in the Tag Group Editor, set the following values:

    a. Driving Comparison**: =**
    b. Comparison Value: **1**

7. Click **OK**.



Tag Group Editor

Tag Group Editor

For provider default

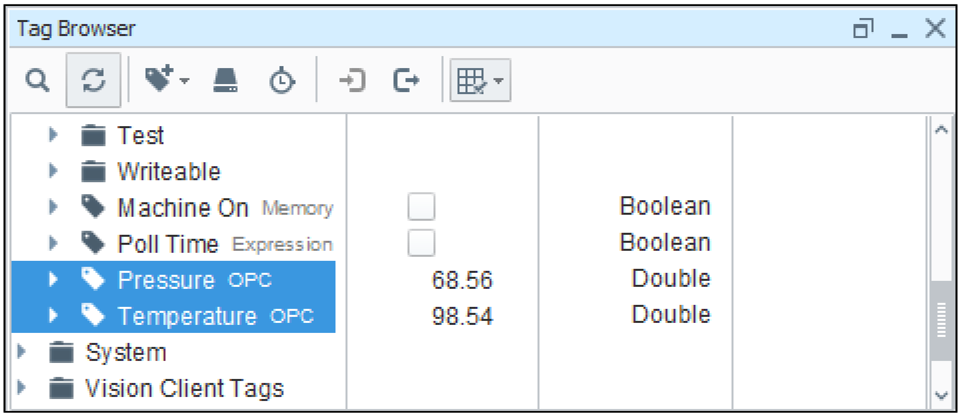
Tag Groups

- Default
- Default Historical
- Direct
- Direct 5 Seconds
- Driven
- Driven Machine State
- Leased
- Time Driven

Properties

Common	
Name	Time Driven
Mode	Driven
Rate	60,000
Leased/Driven Rate	1,000
Driving Expression	timeBetween(now(0), "8...
Driving Comparison	=
Comparison Value	1.0
One Shot	false
OPC Settings	
Data Mode	Subscribed
Read After Write	false
Optimistic Writes	false
Optimistic Write Timeout (MS)	60,000
OPC UA	
Publishing Interval (ms)	-1.0
Queue Size	1
History	
Min Time Between Samples	1
Min Time Units	Seconds
Max Time Between Samples	0
Max Time Units	Hours

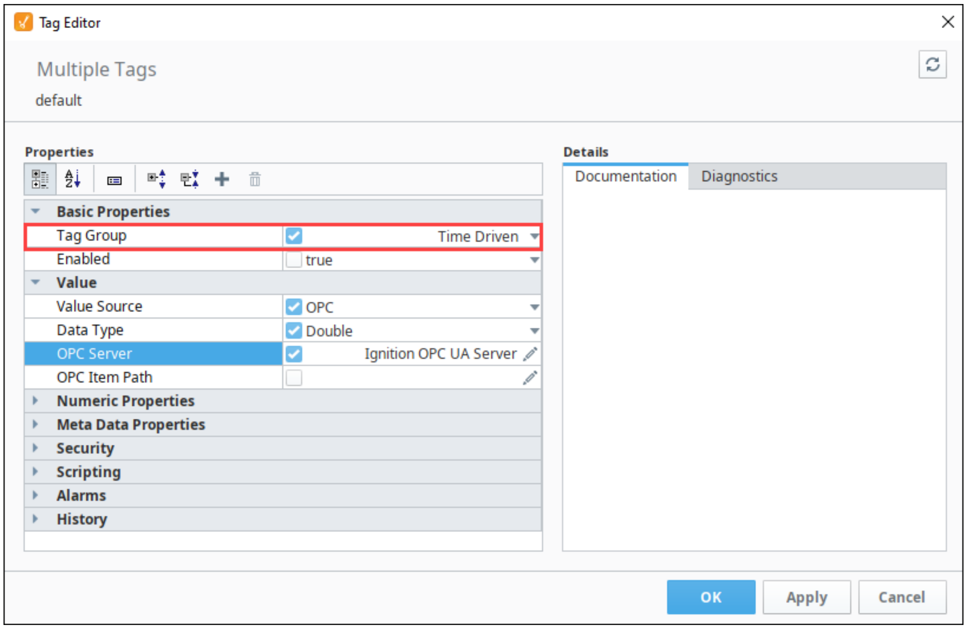
8. Next, select all the Tags that you want to specify in the Tag Group (i.e., **Pressure** and **Temperature**). Right click on the selected Tags, and choose **Edit Tag**.



Tag Browser

Test		
Writeable		
Machine On Memory	□	Boolean
Poll Time Expression	□	Boolean
Pressure OPC	68.56	Double
Temperature OPC	98.54	Double
System		
Vision Client Tags		

9. The Tag Editor will open showing you are editing multiple Tags. To change the Tag Group, choose **Time Driven** from the dropdown, and click **OK.**



Basic Properties		
Tag Group	☑	Time Driven
Enabled	□	true
Value		
Value Source	☑	OPC
Data Type	☑	Double
OPC Server	☑	Ignition OPC UA Server
OPC Item Path	□	
Numeric Properties		
Meta Data Properties		
Security		
Scripting		
Alarms		
History		

10. In the Tag Browser you can look at the Tags to see they are updating at the correct rate. Try adjusting the time range in your expression to change the rate of polling. When the driving conditions are true, that is the time between hours of 8am and 5pm, polling is at the Leased/Driven Rate of 1000 ms. When not, polling is stopped, and the last known value is displayed.

Related Topics ...

- One Shot Tag Group
- Tag Groups

Leased Mode

The Leased Tag Group allows you to poll Tags at a fast rate when they are displayed in a Vision Client window, Perspective Session view, or in the Designer. If the Tags are not displayed (if no Clients or Sessions are showing the Tags), the Tag Group runs at the slow polling rate.

The term "displayed" means the tag can be seen in one of the following ways

- The value is displayed on a component in a Perspective Session
- The value is displayed on a component in a Vision Client
- The value is displayed in the Designer, via a Vision or Perspective component binding. Viewing the tag only in the Tag Browser will not cause the tag group to execute at the Leased Rate.

Essentially, a Leased Tag Group polls Tags on-demand, only when someone needs to look at the Tag values. Note that this change in rate is applied per tag: if Tag A and Tag B are in the same Leased Tag Group, and only Tag A is displayed in a Vision Client, then Tag A will run at the fast rate while Tag B will continue to run at the slow rate.

For a complete list of Tag Group properties, see Tag Groups.

On this page ...

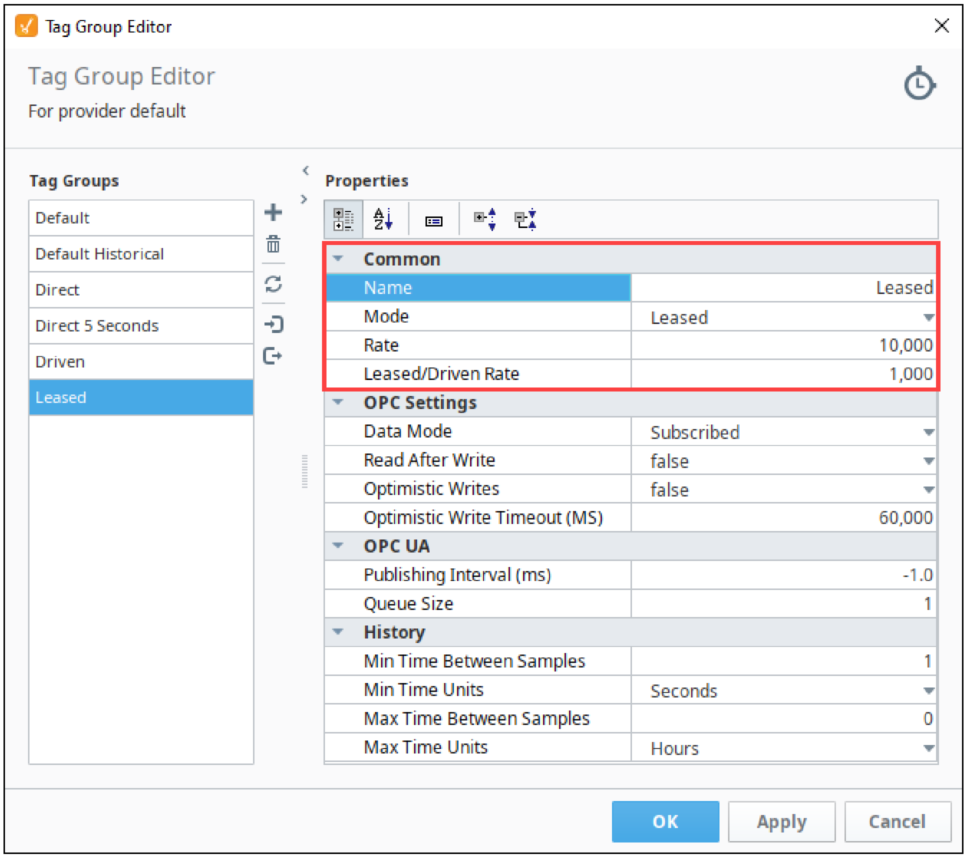
Leased Tag Group

Watch the Video

Add a Leased Tag Group

Let's add a new Leased Tag Group that polls a Tag from the PLC at a 1 second rate when someone needs to view that Tag in the Client. If the Client is closed, or the Tags are not displayed in the Tag Browser of the Designer, the Tags will not poll at all.

1. In the **Tag Browser**, click on the **Tag Group** ⏱ icon to open the **Tag Group Editor**.
2. On the left side of the Tag Group Editor window, you can see all existing Tag Groups. Click on the **Add +** icon on the lower left side of the window to add a new Tag Group.
3. Enter the name of the Tag Group. For example, you can name this new Tag Group **Leased**, and set the **Mode** to **Leased**.
4. Set the **Rate** to 10,000ms (10 seconds) and the **Leased/Driven Rate** to 1,000ms (1 second).
5. Click **OK**.



Tag Group Editor

For provider default

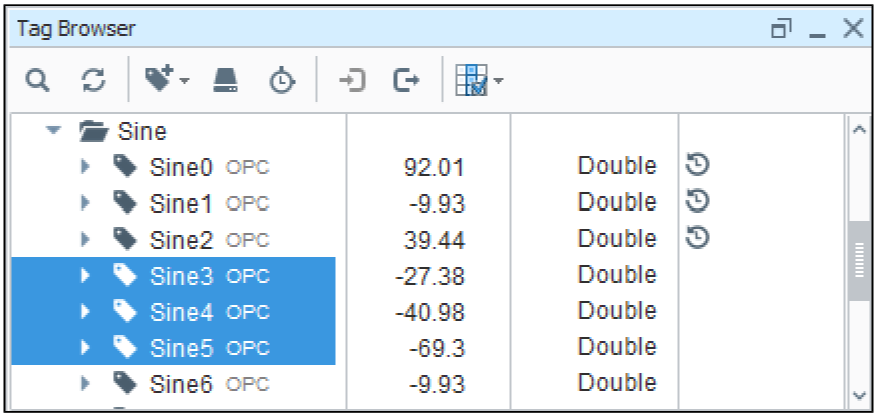
Tag Groups

- Default
- Default Historical
- Direct
- Direct 5 Seconds
- Driven
- Leased

Properties

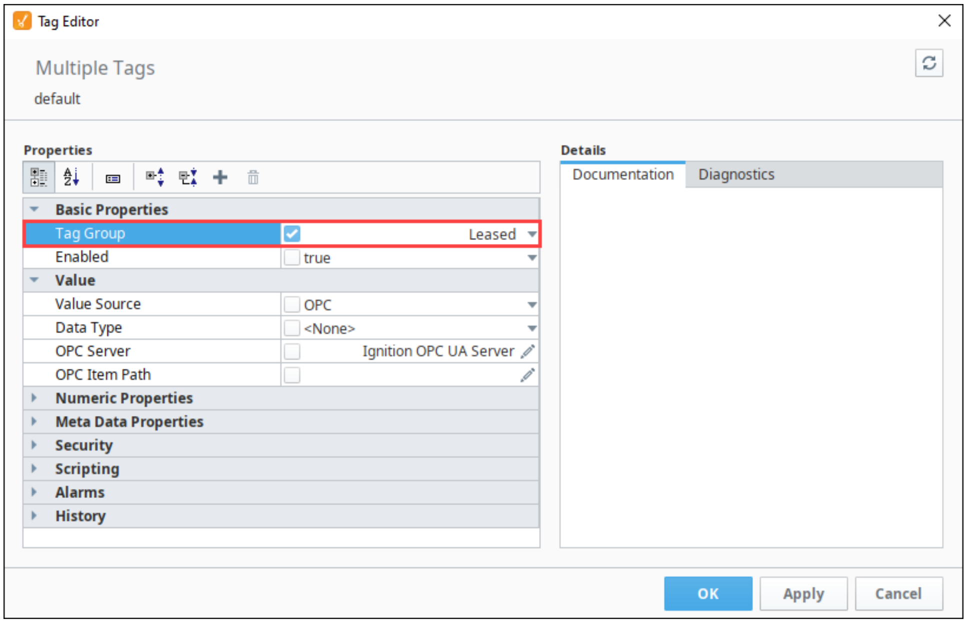
Common	
Name	Leased
Mode	Leased
Rate	10,000
Leased/Driven Rate	1,000
OPC Settings	
Data Mode	Subscribed
Read After Write	false
Optimistic Writes	false
Optimistic Write Timeout (MS)	60,000
OPC UA	
Publishing Interval (ms)	-1.0
Queue Size	1
History	
Min Time Between Samples	1
Min Time Units	Seconds
Max Time Between Samples	0
Max Time Units	Hours

6. Next, select the Tags you want to use on the new Leased Tag Group.
   a. Go to your Tag Browser, and select some Tags you want to add to the Leased Tag Group. This example uses three Sine Tags: Sine3, Sine4, and Sine5.
   b. Right click on the selected Tags, and click on **Edit tags**.



Sine		
Sine0 OPC	92.01	Double
Sine1 OPC	-9.93	Double
Sine2 OPC	39.44	Double
Sine3 OPC	-27.38	Double
Sine4 OPC	-40.98	Double
Sine5 OPC	-69.3	Double
Sine6 OPC	-9.93	Double

7. The Tag Editor will open showing you are editing multiple Tags. From the Tag Group dropdown list, choose **Leased**, and click **OK.**



Tag Editor

Multiple Tags

default

Properties

Basic Properties

Tag Group	☑	Leased
Enabled	□	true

Value

Value Source	□	OPC
Data Type	□	<None>
OPC Server	□	Ignition OPC UA Server
OPC Item Path	□	

Numeric Properties

Meta Data Properties

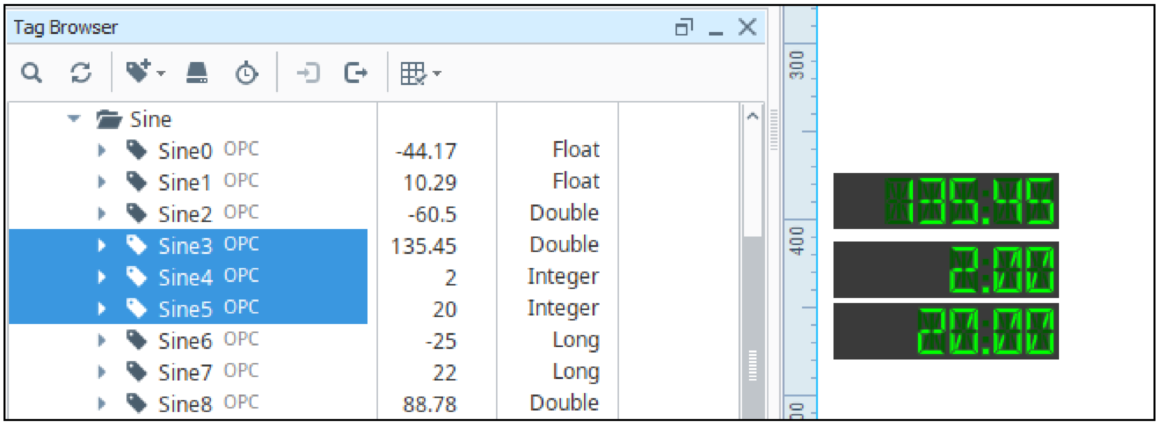
Security

Scripting

Alarms

History

You just created a new Leased Tag Group and added some Tags. These Tags are only polled quickly when a component that is bound to it is showing in a Client, Session, or when the Tags are showing in the Designer. For example, these three Sine Tags that are part of the Leased Tag Group are each bound to an LED component.



Tag Browser

Sine

Sine0 OPC	-44.17	Float
Sine1 OPC	10.29	Float
Sine2 OPC	-60.5	Double
Sine3 OPC	135.45	Double
Sine4 OPC	2	Integer
Sine5 OPC	20	Integer
Sine6 OPC	-25	Long
Sine7 OPC	22	Long
Sine8 OPC	88.78	Double

⚠ Remember that Tag History is stored based on the current Ignition Tag value. If your Leased Tag Group has a **0** slow rate, you will not get updated data to store for history. Always make sure your Tag Group slow speed is at least as fast as your History Tag Group speed.

Related Topics ...

- System Tags
- Tag Groups

One Shot Tag Group

The Driven Tag Group can be configured as a One-shot Tag Group. The One Shot Tag Group will not run at a rate, but instead it will be triggered by a change in the driving Tag's value. One-shot polling executes once when the condition becomes true, then waits for that condition to become false then true again. This makes it follow a 'rising edge' or 'falling edge' pattern.

Let's make a Driven One-shot Tag Group that only updates once when a Memory Tag goes to true. We are using a Memory Tag so a user can change the value when they want to poll the PLC. This is actually a good example of how an operator can ask the PLC for new values whenever they want to, using a refresh button on the screen.

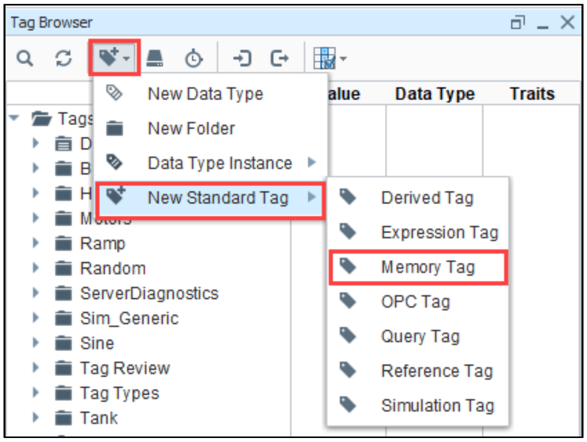
On this page ...

Driven Scan Class - One shot

Watch the Video

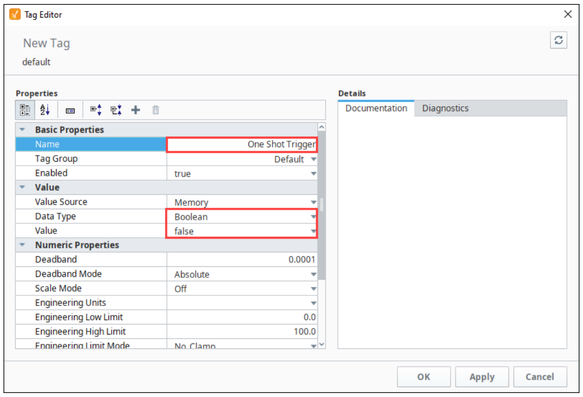
Create a Memory Tag

1. From Tag Browser, click on the New Tag icon or right click on a folder and select **New Standard Tag > Memory Tag**.



2. In the **Tag Editor**, enter the following:
   a. Tag Name: **One Shot Trigger**
   b. Data Type: **Boolean**
   c. Value: **false**

3. Click **OK** to add the One Shot Trigger Tag to the Tag Browser.



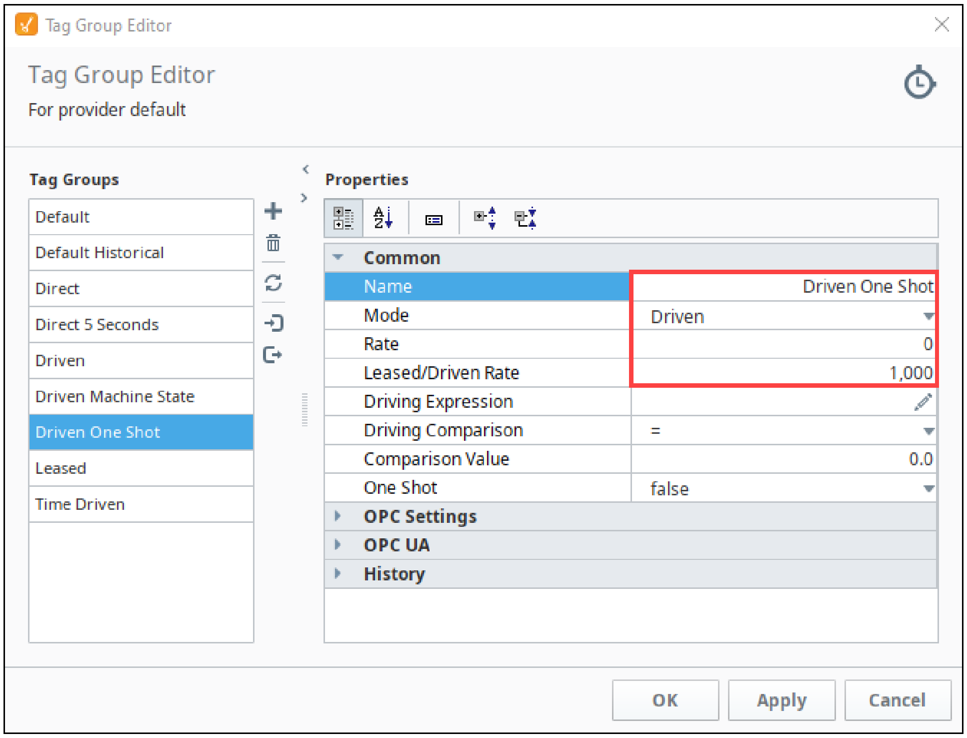
Basic Properties	
Name	One Shot Trigger
Tag Group	Default
Enabled	true
Value	
Value Source	Memory
Data Type	Boolean
Value	false
Numeric Properties	
Deadband	0.0001
Deadband Mode	Absolute
Scale Mode	Off
Engineering Units	
Engineering Low Limit	0.0
Engineering High Limit	100.0
Engineering Limit Mode	No Clamp

Create a Driven One-Shot Tag Group

Once you have your driving Tag created, add a Driven one-shot Tag Group that lets a user poll Tags manually.

1. In the Tag Browser, click on the **Tag Group** ⏱ **icon** to open the Tag Group Editor.
2. Create a new Tag group by clicking on the **Add** ✚ icon.
3. In **<u>Tag</u> Group Editor**, enter the following:

   Name: Driven One Shot
   Mode: Driven
   Rate: 0
   Leased/Driven Rate: 1000ms



Tag Group Editor

For provider default

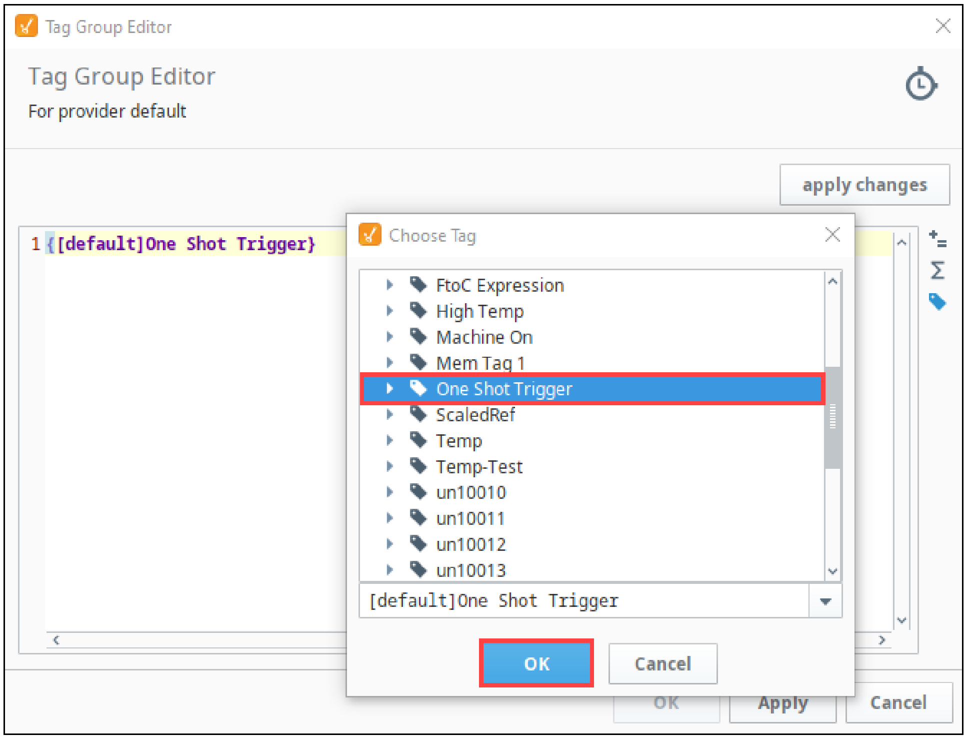
Tag Groups

- Default
- Default Historical
- Direct
- Direct 5 Seconds
- Driven
- Driven Machine State
- Driven One Shot
- Leased
- Time Driven

Properties

Common	
Name	Driven One Shot
Mode	Driven
Rate	0
Leased/Driven Rate	1,000
Driving Expression	
Driving Comparison	=
Comparison Value	0.0
One Shot	false
OPC Settings	
OPC UA	
History	

4. Set the driving **Expression** by clicking the **Edit** ✎ icon on the right.
5. Select the **Tag** 🏷 icon and choose the Tag you want to set the condition on (i.e., One Shot Trigger).
6. Click **OK** and then click **apply changes**. Note: This Tag group executes one time when the One shot condition is set to **true**.



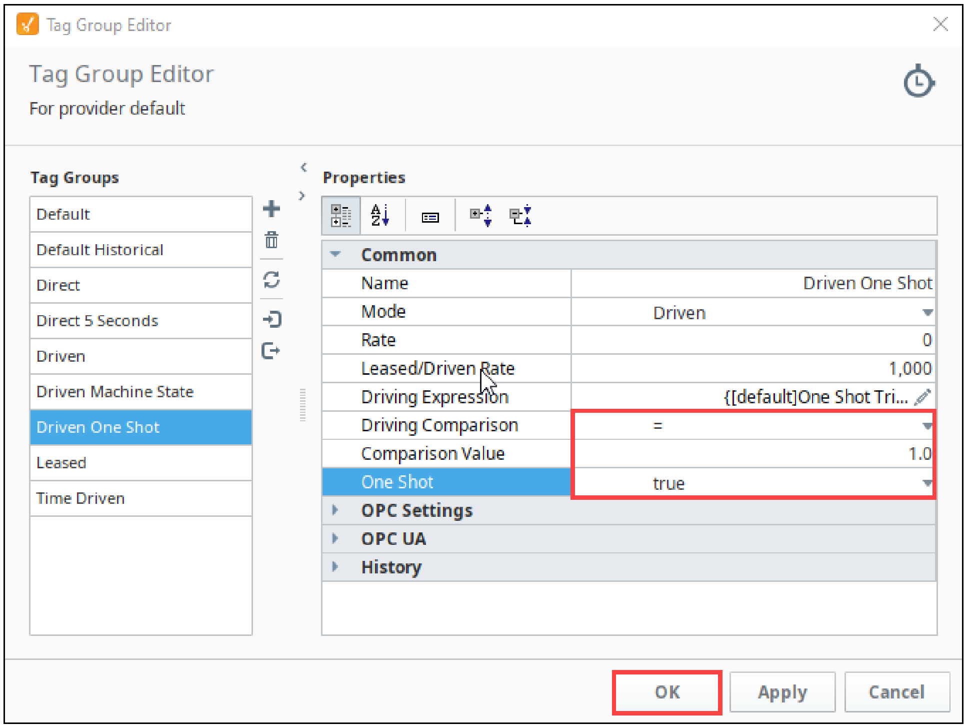
Tag Group Editor

For provider default

```
{[default]One Shot Trigger}
```

[default]One Shot Trigger

7. While still in the Tag Group Editor, set the following values:
   Driving Comparison: **=**
   Comparison Value: **1**
   One Shot: **true**



Tag Group Editor

For provider default

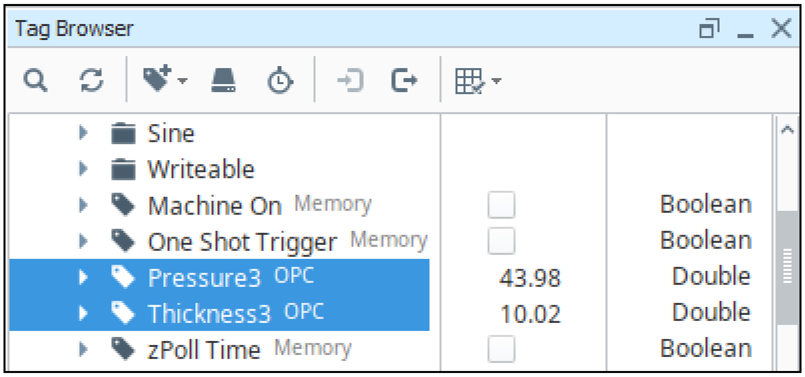
Common	
Name	Driven One Shot
Mode	Driven
Rate	0
Leased/Driven Rate	1,000
Driving Expression	{[default]One Shot Tri...
Driving Comparison	=
Comparison Value	1.0
One Shot	true

8. Click **OK** to save the Tag Group.

Apply the Driven One Shot Tag Group

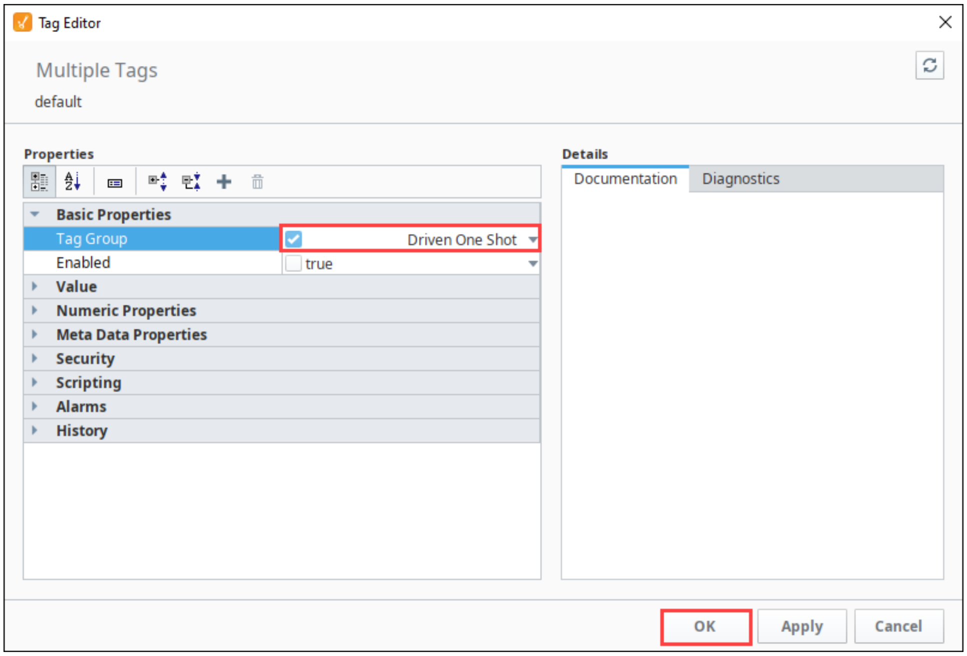
Now, we have to apply our Driven One Shot Tag Group to the One Shot Trigger Tag we created above.

1. Select all the Tags that you want to run in this Tag Group. This example uses Pressure3 and Thickness3 Tags.
2. Right click on one of the selected Tags, and click the **Edit Tag** ✎ icon.



Sine		
Writeable		
Machine On Memory	□	Boolean
One Shot Trigger Memory	□	Boolean
Pressure3 OPC	43.98	Double
Thickness3 OPC	10.02	Double
zPoll Time Memory	□	Boolean

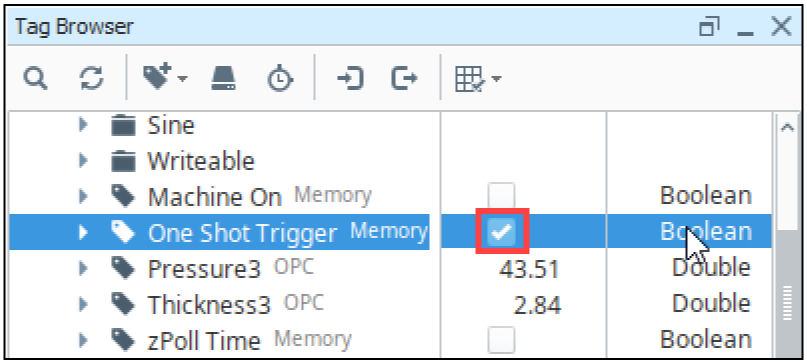
3. The Tag Editor will open showing you are editing multiple Tags. From the Tag Group dropdown list, choose **Driven One Shot**, and click **OK**.



Tag Group ☑ Driven One Shot

Enabled □ true

4. Let's test it out. In the Tag Browser, mark the checkbox for the **One Shot Trigger** Tag to set the value to **true**. You will see the Pressure3 and Thickness3 values update once, and then stop.
   Uncheck the One Shot Trigger Tag, to set the value to **false**, and check it again to set it to **true**, and you will see the values update again.



Sine		
Writeable		
Machine On Memory	□	Boolean
One Shot Trigger Memory	☑	Boolean
Pressure3 OPC	43.51	Double
Thickness3 OPC	2.84	Double
zPoll Time Memory	□	Boolean

5. You can also drag a Momentary Button component onto a Vision window, then drag your **One Shot Trigger** Tag onto that button.
6. Put the Project into **Preview** mode ▶ and then click the Momentary Button component. Now your operators can request new values and the button automatically resets the Tag.
7. In the **Tag Browser**, you will see the Tag values update once, and then stop.

Related Topics ...

- Tag Groups
- Leased Mode
- Driven Mode

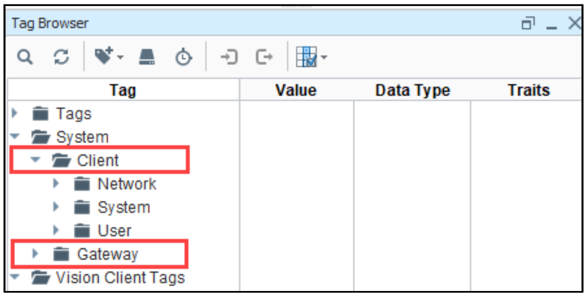
System Tags

System Tags

On this page

...

System Tags provide status about the Ignition system, such as memory usage, performance metrics, and so on. System Tags cannot be deleted or modified. In the Tag Browser, under the System folder, there two folders: **Client** and **Gateway**. The scope for each is slightly different.



System Tags

Watch the Video

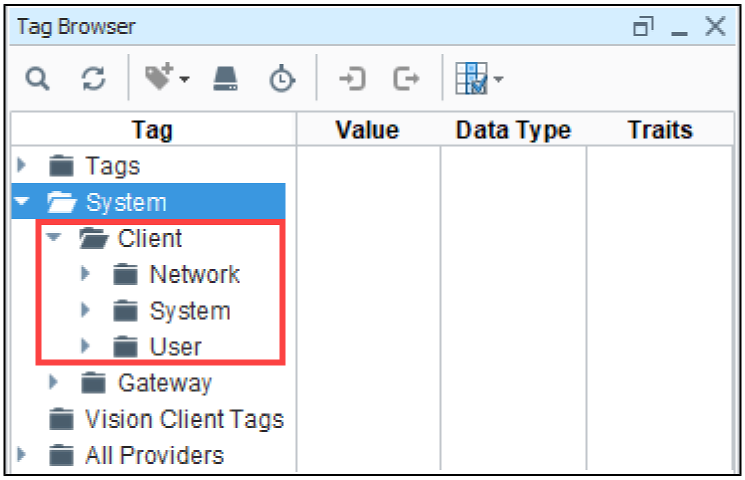
System Client Tags for Vision

Client-scoped System Tags provide status information about the client's system. Every individual client is going to have their own values like IP address, host name, and username. You cannot modify Client System Tags.

There are three folders within the System > Client Tags folder: Network, System, and User. These Tags are provided with Ignition. They can be used with the Vision module for any Vision Client. They cannot be modified.

Tag	Format or Example Value	Data Type
Network Folder		
GatewayAddress	Gateway URL address.	string
GatewayRedundancyRole	Redundancy State of the Gateway that the client is connected to. Independent, Master, Backup.	string
Hostname	Hostname (name) of the computer that the Client is running on.	string
IPAddress	IP Address of the computer that the Client is running on.	string
MACAddress	MAC Address of the computer that the Client is running on.	string
System Folder		
CurrentDateTime	Current system date and time. Format is yyyy-mm-dd hh:mm:ss a.	Date Time
DefaultDatabase	Name of the default database connection used by the project.	string
DefaultTagProvider	Name of the default Tag Provider used by the project.	string
FPMIVersion	Current Ignition version in use.	string
JavaVersion	Current Java version in use by the client.	string

OperatingSystem	Operating system of the computer that the Client is running on.	string
LastProjectUpdate	The following feature is new in Ignition version **8.0.4** Click here to check out the other new features Date of the last received update notification.	DateTime
ProjectDescription	Description field for the current project.	string
ProjectName	Name field for the current project.	string
ProjectTitle	Title field for the current project.	string
SystemFlags	A byte array of flags for the current state of the Client.	integer
UserSource	Name of the user source for the current Client.	string
User Folder		
Country	Two letter country code according to OS. for example: US.	string
CurrentWindow	The current main window open in the project (top most Floating, Maximized window).	string
DateFormatFull	Full date format according to the OS. Format: EEEE, MMMM d, y	string
DateFormatLong	Long date format: MMMM d, y	string
DateFormatMedium	Medium date format: MMM d, y	string
DateFormatShort	Short date format: M/d/yy	string
DateTimeFormatFull	Full date and time format: EEEE, MMMM d, y 'at' h:mm:ss a zzzz	string
DateTimeFormatLong	Long date and time format: MMMM d, y 'at' h:mm:ss a	string
DateTimeFormatMedium	Medium date and time format: MMM d, y, h:mm:ss a zzzz	string
DateTimeFormatShort	Short date and time format: M/d/y, h:mm a	string
HomeFolder	Home folder according to OS. For example: C:\Users\psmith.	string
Language	Language according to OS. For example: "en" for English.	string
OSUsername	OS user name, for example: PSmith	string
RolesDataSet	Dataset with Roles for currently logged in user. For example: Dataset[2R x 1C].	dataset
RolesString	Comma separated string with Roles for currently logged in user. For example: Administrator, Operator.	string
TimeFormatFull	Full time format according to OS. Format: h:mm:ss a zzzz	string
TimeFormatLong	Long time format: h:mm:ss a z	string
TimeFormatMedium	Medium time format: h:mm:ss a	string
TimeFormatShort	Sort time format: h:mm a	string
Timezone	Current timezone, for example, America/Los Angeles.	string
Username	Currently logged in username, for example, PSmith.	string



Gateway System Tags

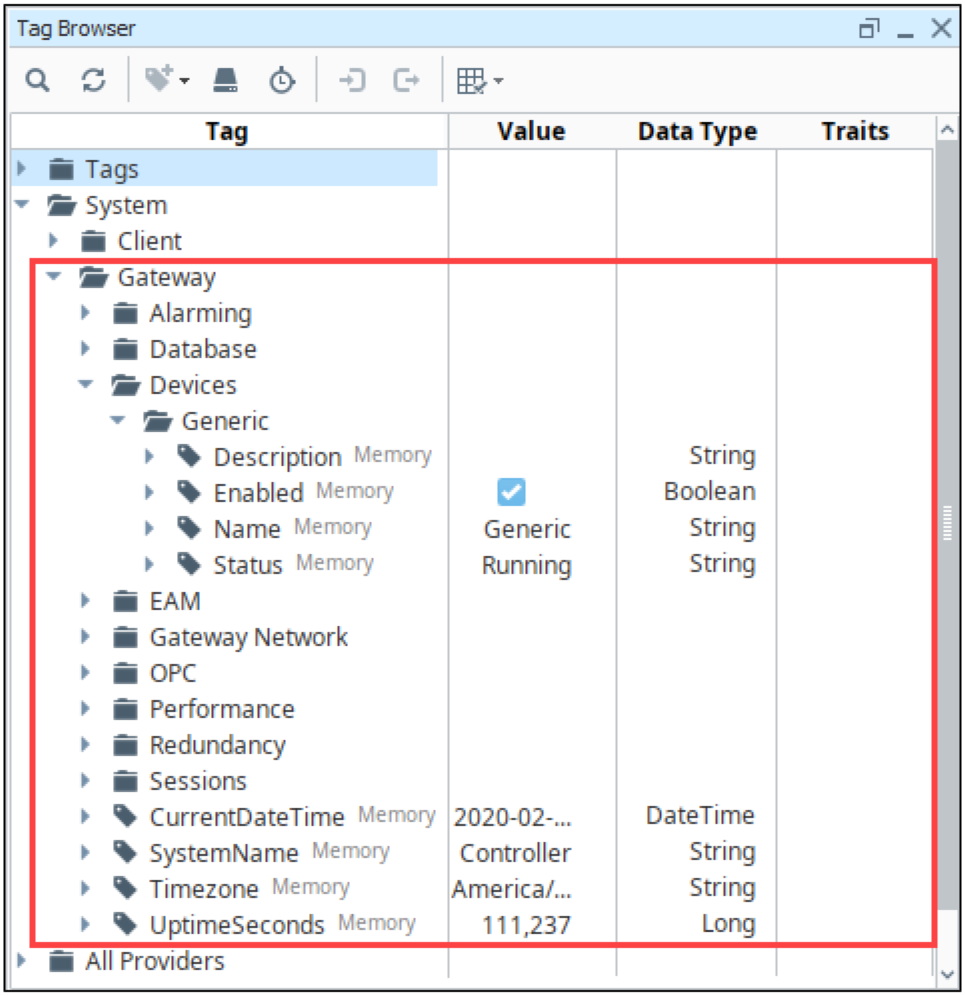
Gateway System Tags exist in the Gateway scope. There are nine folders within the **System > Gateway Tags** folder: Alarming, Database, Devices, EAM, Gateway Network, OPC, Performance Redundancy and Sessions.

The following Gateway-scoped System Tags are provided with Ignition. They can be used anywhere within Ignition.

Tag	Value	Data Type
CurrentDateTime	Current system date and time. Format is yyyy-mm-dd hh:mm:ss a.	DateTime
SystemName	Computer name where the Gateway ins installed.	String
Timezone	Timezone on the Gateway computer. For example, America/Los Angeles.	String
UptimeSeconds	Number of seconds since Ignition was started.	Long
Alarming Folder		
Active and Acked	Number of alarms currently active and acknowledged.	Integer
Active and Unacked	Number of alarms currently active and unacknowledged.	Integer
Clear and Acked	Number of alarms cleared and acknowledged.	Integer
Clear and Unacked	Number of alarms cleared and unacknowledged.	Integer
Database Folder There will be a subfolder for each database connection, or none if there are no connections. Each subfolder will have the following Tags.		
ActiveConnections	Number of active connections in the pool to this database connection.	Integer
Available	Indicates whether this datasource is available.	Boolean
AvailableThroughFailover	Indicates if any database along the failover chain attached to this data source can be reached.	Boolean
AvgQueryTime	Average time, in seconds, that it is taking database queries to run.	Integer
ConnectionSaturation	Percentage of possible query throughput that is being used (ratio of currently active connections to maximum possible connections).	Double
QueriesPerSecond	Number of queries running per second.	Integer

The following feature is new in Ignition version **8.0.10**
Click here to check out the other new features

Devices Folder A "Devices" folder was added in 8.0.10. For each device configuration on the gateway, a separate subfolder will exist. The contents of each subfolder are listed below.		
Description	Description of the device connection as configured on the device connection on the Gateway.	String
Enabled	Boolean representing whether or not the device connection is enabled.	Boolean
Name	Name of the device connection.	String
Status	Status of the connection.	String
Gateway Network Each system connected over the gateway network receives a special folder, containing the following two tags		
IsAvailable	Whether the remote gateway is available or not.	Boolean
LastComm	A timestamp representing the last known communication received from the remote gateway.	DateTime
OPC Folder There will be a subfolder for each OPC UA Server. Each subfolder will have the following three Tags.		
Connected	Whether the OPC UA server is connected to Ignition.	Boolean
Enabled	Whether the OPC UA server connection is enabled.	Boolean
State	The state name of the connection. For example: Connected, Faulted, Connecting.	String
Performance Folder		
Available Disk Space (MB)	Available disk space on the computer Ignition is installed on, in megabytes.	Long
CPU Usage	CPU Utilization as reported to the Java Virtual Machine.	Double
Disk Utilization	Percentage of hard disk that is in use.	Double
Max Memory	Maximum amount of RAM the Gateway can use, in megabytes.	Long
Memory Usage	Amount of RAM currently in use by the Gateway, in megabytes.	Long
Memory Utilization	Current memory useage/maximum memory usage.	Double
Redundancy Folder		
Connection, Is Connected	Whether this Gateway is connected to another for redundancy.	Boolean
Connection, PeerId	The ID of the Gateway connected to, empty string if not connected.	String
ActivityLevel	Indicates where the Gateway is in the redundant state. Can be undecided, cold, warm, hot, or active.	String
IsActive	Whether the Gateway is running.	Boolean
IsMaster	Whether the Gateway is the master. False if the backup is in control.	Boolean
Role	Named role of the Gateway. Options: Independent, Master, Backup.	String
Sessions Folder		
SessionCount	Number of active sessions on this Gateway. Designers, Clients, and Sessions.	Integer



Tag	Value	Data Type	Traits
Tags			
System			
Client			
Gateway			
Alarming			
Database			
Devices			
Generic			
Description Memory		String	
Enabled Memory	☑	Boolean	
Name Memory	Generic	String	
Status Memory	Running	String	
EAM			
Gateway Network			
OPC			
Performance			
Redundancy			
Sessions			
CurrentDateTime Memory	2020-02-...	DateTime	
SystemName Memory	Controller	String	
Timezone Memory	America/...	String	
UptimeSeconds Memory	111,237	Long	
All Providers			

Tag Providers

On this page ...

Tag Providers and Drivers

At the highest level of Tag configuration is the Tag Provider. A provider is a Tag database (a collection of Tags) and a name. An Ignition Gateway can have any number of Tag providers, and therefore the name is used to distinguish which provider a Tag comes from. Tag Providers can be set up with security or even disabled independent of each other.

Every copy of Ignition has its own Tags. With the Remote Tag provider, Ignition can also see the Tags on another Gateway, as long as the two Gateways are connected through a Gateway Network.

All Tags reside in a Tag Provider and have realtime values. Additionally, there is the concept of Tag historian providers, which can store and query historical data for Tags. Each Tag can optionally have a historian provider assigned to it to whom it will report value changes for historical storage.

Realtime Provider Types

There are no differences between the two types of Tag Providers other than where the Tag's configuration is stored and executed. You can have as many Tag providers as you want, and there are security settings to either lock them or open them to the network.

In these examples we refer to two copies of Ignition. The "local" Ignition is the Ignition that you are currently configuring. The "remote" Ignition is another installation of Ignition somewhere on your network.

Internal Tag Provider

Internal Tag Providers store all configuration and do any execution (read, write, history, alarms) through the local Ignition Gateway. Every new Ignition installation automatically creates an Internal Tag Provider named "default." You can add as many Internal Tag Providers as you want.

This provider can be exposed or hidden from other Gateways on the network through the Gateway's OPC-UA settings.

Tag Providers

Watch the Video

Remote Tag Provider

Remote Tag Providers connect a remote installation of Ignition and access those Tags. The Remote Tag Provider works by creating a link from the local Gateway to a Tag provider on a remote Gateway using a Gateway Network connection. The local Ignition may be allowed to read and write to the remote Tags, but any execution is handled by the remote Gateway. So, things like writing to a PLC, alarms, and history will still be handled by the remote Ignition.

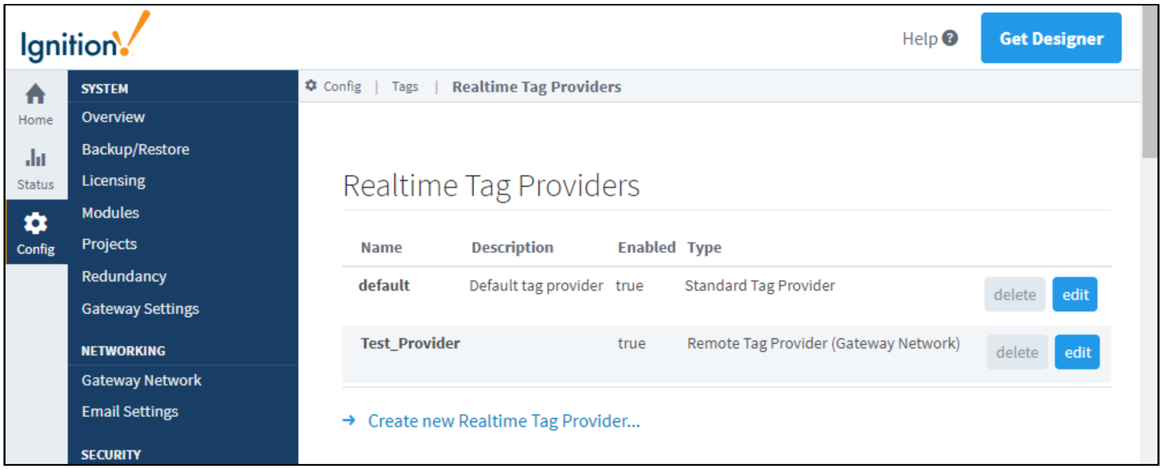
By default, a Remote Tag Provider will fall under the Default Security Zone and be read only.

Remote Tag Provider

Watch the Video

Configuring Realtime Providers

Realtime Tags providers are configured in the Gateway's **Config** section under **Tags > Realtime**. After installation, the Ignition Gateway will start with an internal provider defined. You can edit its name and settings by selecting **edit** to the right of its entry in the table, or create new providers by selecting **Create new Realtime Tag Provider** below the table.



Realtime Tag Providers

Name	Description	Enabled	Type
default	Default tag provider	true	Standard Tag Provider
Test_Provider		true	Remote Tag Provider (Gateway Network)

Note: When setting up your Realtime Tag Provider, naming your Provider the same name as the database connection is **not** recommended. Doing so may cause no values to be returned when querying Tag data.

There are two types of Realtime Tag Providers that you can choose from:

- Standard Tag Provider
- Remote Tag Provider

Standard Tag Provider

Tags are stored inside of Ignition and executed by the system.

Setting	Description
Name	The name of the provider.
Description	The description of the provider.
Enabled	If true, tag provider is enabled. Default is true.
Default Database	The default database connection to use for expression tags that run SQL queries. All query tags with default database providers selected with run their queries against this database source.
Tag Editing Permissions	The following feature is new in Ignition version **8.0.16** Click here to check out the other new features Determines the roles required to edit, create, or delete tags in the provider. Expects a path to a Security Level. Multiple levels can be specified, separated by a comma. For example: `Authenticated/Roles/Administrator, SecurityZones/MyZone` When multiple security levels are provided, the radio buttons determine if the user needs all of the listed security levels, or at least one.

Remote Tag Provider

Tag Provider from one Gateway is brought in to another Gateway.

Setting	Description
Name	The name of the provider.
Description	The description of the provider

Gateway	The name of the Gateway on the Gateway Network that this provider is coming from.
Provider	The name of the provider as it is on the remote Gateway. This does not have to be the same as its name on the new Gateway.
History Access Mode	This setting dictates how tag history is queried for remote tags. Gateway Network will go through the Gateway Network to query history. Database will query the database directly.
History Datasource	The datasource to query when History Access Mode is set to Database.
History Driver	If querying the database directly, this is the gateway name of the remote system. It is used to identify data from that system in the database.
History Provider	If querying the database directly, this is the name of the tag provider on the remote system. It is used along with driver name to identify the correct tags in the database.
Alarms Enabled	If true, alarms configured on the remote gateway will be enabled on the new Gateway.
Alarm Mode	How alarm state should be monitored. In 'queried', state will be queried through the Gateway network when necessary. In 'subscribed', the state will be subscribed, and updates will be sent asynchronously. Subscribed provides better performance, but uses more memory.

Related Topics ...

- Creating Tags
- User Defined Types - UDTs
- Tag Groups

Tag Access

In addition to setting up security on individual Tags, you can set up security policies specific to each Security Zone. Tag Access is one of the options for a Security Policy. This is configured on the Gateway Webpage in the **Config** tab under **Service > Security**.

For a Security Zone, you can set up one of five access levels to each Tag Provider:

Access Level	Definition
Inherited	The security zone inherits the access level that is set in the Default Provider Access Level field.
None	The security zone has no access to this tag provider.
ReadOnly	The security zone has Read only access to this tag provider.
ReadWrite	The security zone has Read and write access to this tag provider.
ReadWriteEdit.	The security zone has Read, Write, and Edit access to this tag provider.

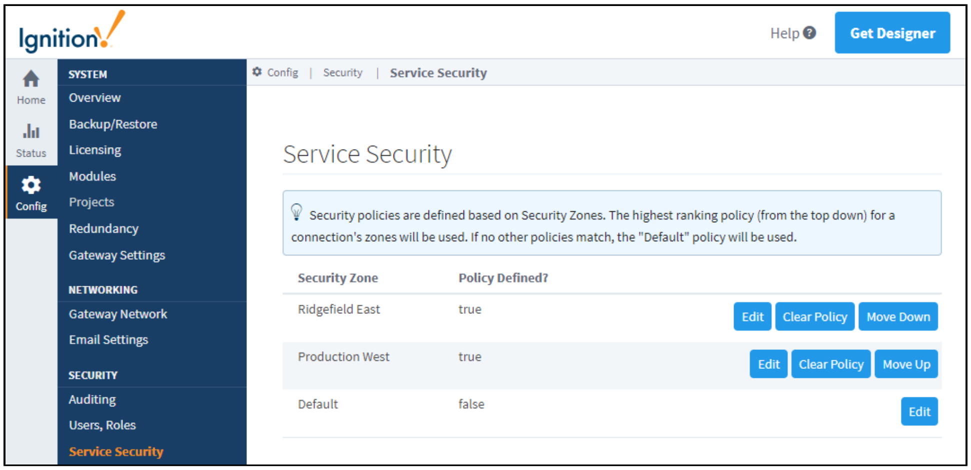
On this page

...

Configuring Tag Access

Typically, the Administrator will identify what zones have access to which Tag Providers.

1. Go to the **Config** tab of the Gateway Webpage.
2. Choose **Security> Service Security** from the menu on the left.
   The **Service Security** page is displayed. You will see a list of the existing Security Zones for your Gateway.

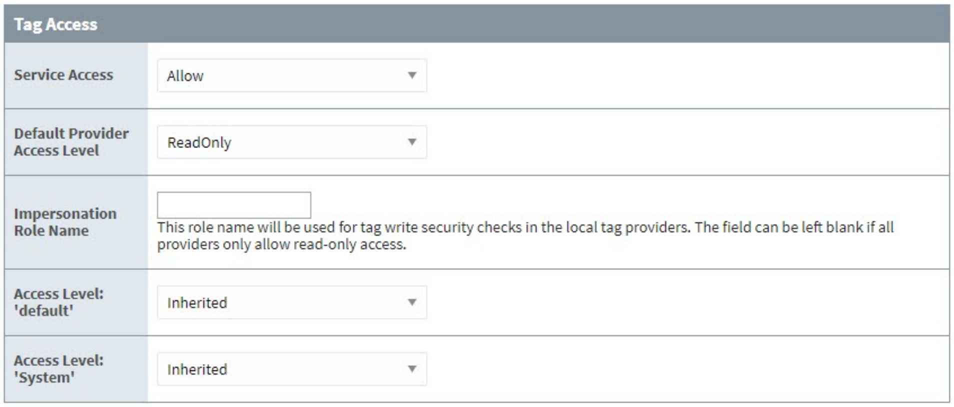


Service Security

Security policies are defined based on Security Zones. The highest ranking policy (from the top down) for a connection's zones will be used. If no other policies match, the "Default" policy will be used.

Security Zone	Policy Defined?	
Ridgefield East	true	Edit · Clear Policy · Move Down
Production West	true	Edit · Clear Policy · Move Up
Default	false	Edit

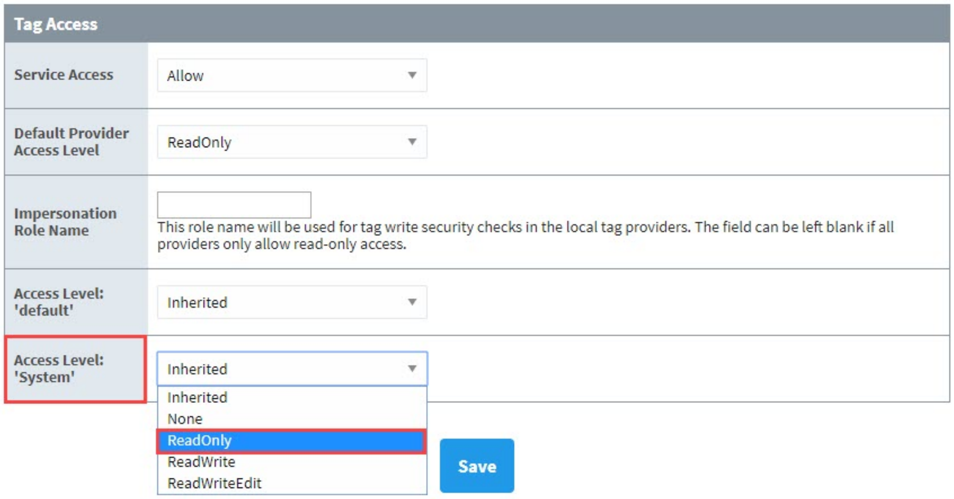
3. To setup or modify Tag access for a zone, click on the Edit button for that zone. The Security Policy page is displayed.
4. Scroll down to the Tag Access section.



Tag Access

Service Access	Allow
Default Provider Access Level	ReadOnly
Impersonation Role Name	This role name will be used for tag write security checks in the local tag providers. The field can be left blank if all providers only allow read-only access.
Access Level: 'default'	Inherited
Access Level: 'System'	Inherited

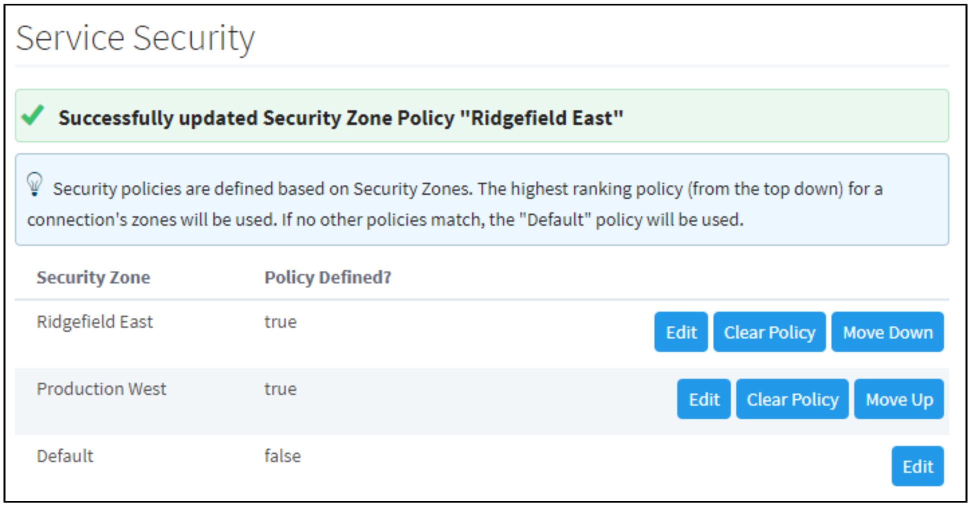
5. In the Default Provider Access level, set the default access level to ReadWrite.
6. For the access level to the System Tag Provider, set the option to ReadOnly. Now this zone will only be able to see System Realtime Tag Provider Tags, but not write to them or edit them.



Tag Access

Service Access	Allow
Default Provider Access Level	ReadOnly
Impersonation Role Name	This role name will be used for tag write security checks in the local tag providers. The field can be left blank if all providers only allow read-only access.
Access Level: 'default'	Inherited
Access Level: 'System'	Inherited

7. Click the **Save** button. You will see a confirmation message on the Service Security page. For more information, see Security Zones.



Service Security

✔ **Successfully updated Security Zone Policy "Ridgefield East"**

Security policies are defined based on Security Zones. The highest ranking policy (from the top down) for a connection's zones will be used. If no other policies match, the "Default" policy will be used.

Security Zone	Policy Defined?	
Ridgefield East	true	Edit Clear Policy Move Down
Production West	true	Edit Clear Policy Move Up
Default	false	Edit

Related Topics ...

- Tag Security

- Understanding Tags
- Security

Tag Security

Tag security is often the best way to configure security for data access. By defining security on a Tag, you affect the Tag across wherever it is used, as opposed to configuring component security on each component that displays or controls that Tag.

Prior to release 8.0.12, users with specific roles and zones can be given read/write access to a Tag, while other users with other roles are excluded from modifying the Tag. Tags can be given read only permission or read/write permission for all users.

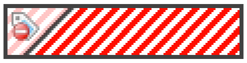
The following feature is new in Ignition version **8.0.13**
Click here to check out the other new features

As of release 8.0.13, Tag security has three options:

- Read Permissions: Defines the security levels required in order to read values from a Tag
- Read Only: Defines whether a Tag is read-only or writable
- Write Permissions: Defines the security levels required in order to write values to a Tag

Users with specific roles and zones can be given read/write access to a Tag, while other users with other roles are excluded from modifying the Tag.

If a user opens a Perspective view or a Vision client window that has components that are bound to a Tag they do not have permissions for, the user will see a forbidden overlay on top of the component. For more information, see Tag Quality and Overlays.



On this page ...

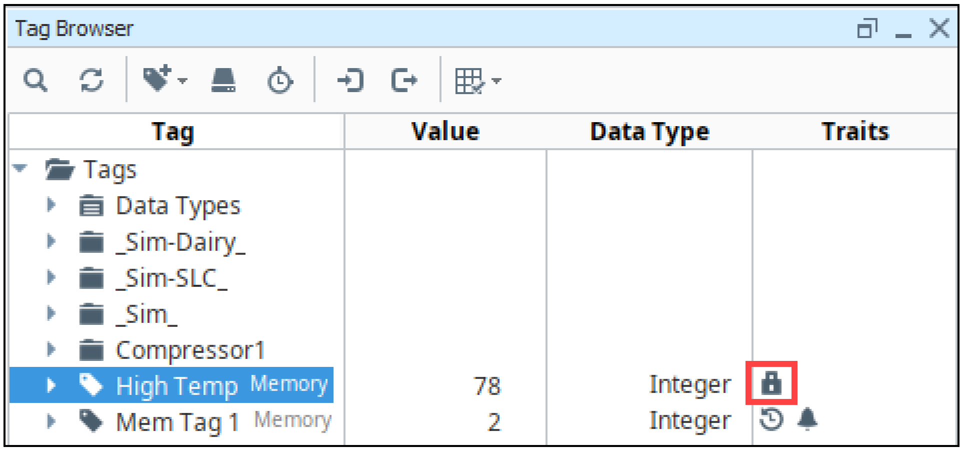
INDUCTIVE UNIVERSITY

Tag Security

Watch the Video

Read Only Security

When a Tag is set to read only, a **Lock** 🔒 icon is displayed next to the Tag in the Tag Browser.



Tag	Value	Data Type	Traits
Tags			
Data Types			
Sim-Dairy			
Sim-SLC			
Sim			
Compressor1			
High Temp Memory	78	Integer	🔒
Mem Tag 1 Memory	2	Integer	⟲ 🔔

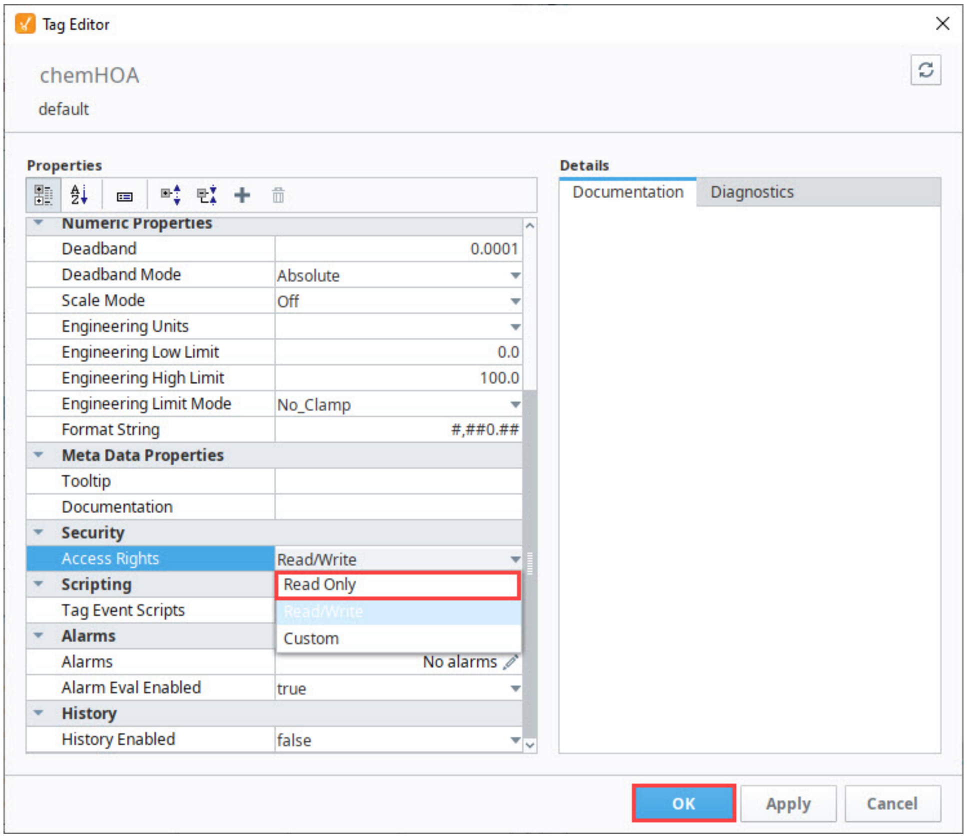
Tag Security Prior to Release 8.0.13

These sections provide information on how Tag Security functions in release 8.0.12 and earlier.

Tag Read/Write Security

By default, Tags have read/write security. You can change it to read only security using the Tag Browser in the Designer.

1. In the **Tag Browser**, right-click on the Tag, and select the **Edit Tag** ✎ icon.
2. Scroll down to the Security section. Use the dropdown list to choose Read Only.



Tag Editor

chemHOA

default

Properties

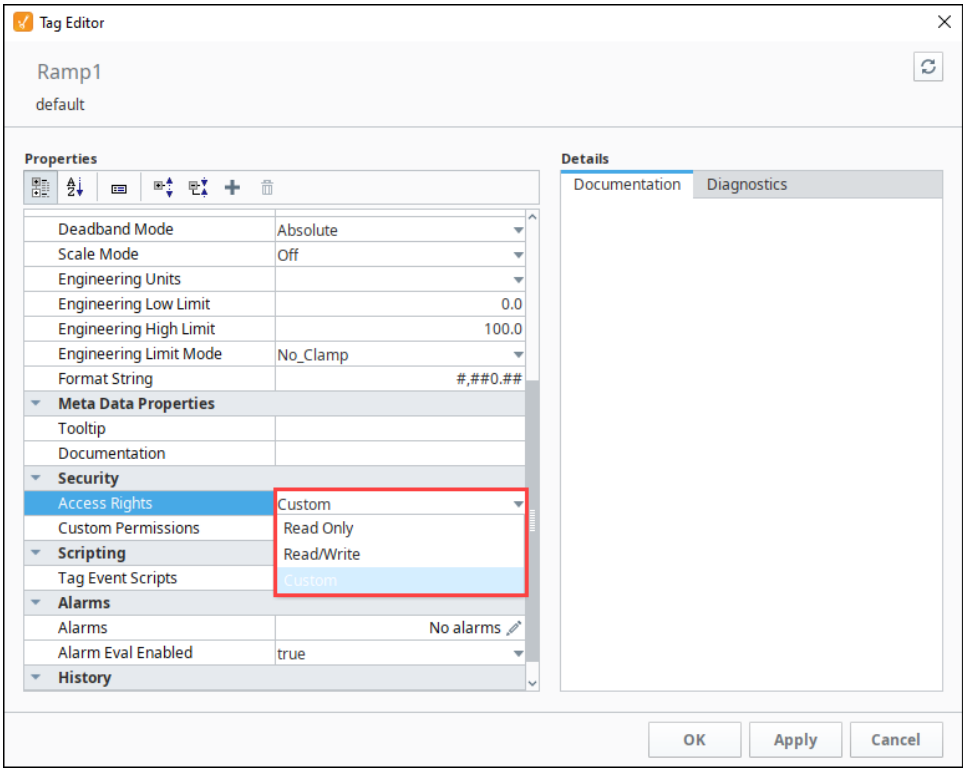
Numeric Properties	
Deadband	0.0001
Deadband Mode	Absolute
Scale Mode	Off
Engineering Units	
Engineering Low Limit	0.0
Engineering High Limit	100.0
Engineering Limit Mode	No_Clamp
Format String	#,##0.##
Meta Data Properties	
Tooltip	
Documentation	
Security	
Access Rights	Read/Write
Scripting	
Tag Event Scripts	
Alarms	
Alarms	No alarms
Alarm Eval Enabled	true
History	
History Enabled	false

3. Click **OK** to close the Tag Editor.

Tag Custom Security

You can configure custom security on individual Tags, giving access to only certain users with only certain roles or in certain zones.

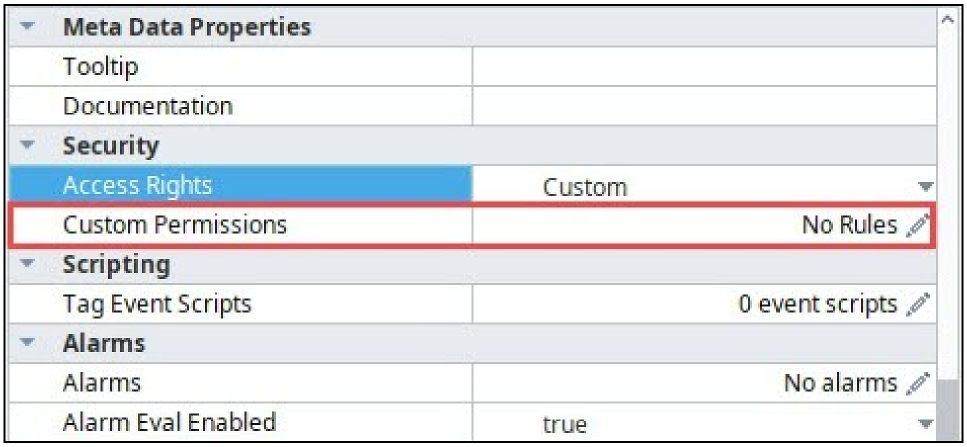
1. In the Tag Browser, right-click on the Tag, and select the **Edit Tag** ✎ icon. This opens the Tag Editor.
2. In the Tag Editor, scroll down to the **Security** section. Use the dropdown list to choose **Custom**.



Ramp1
default

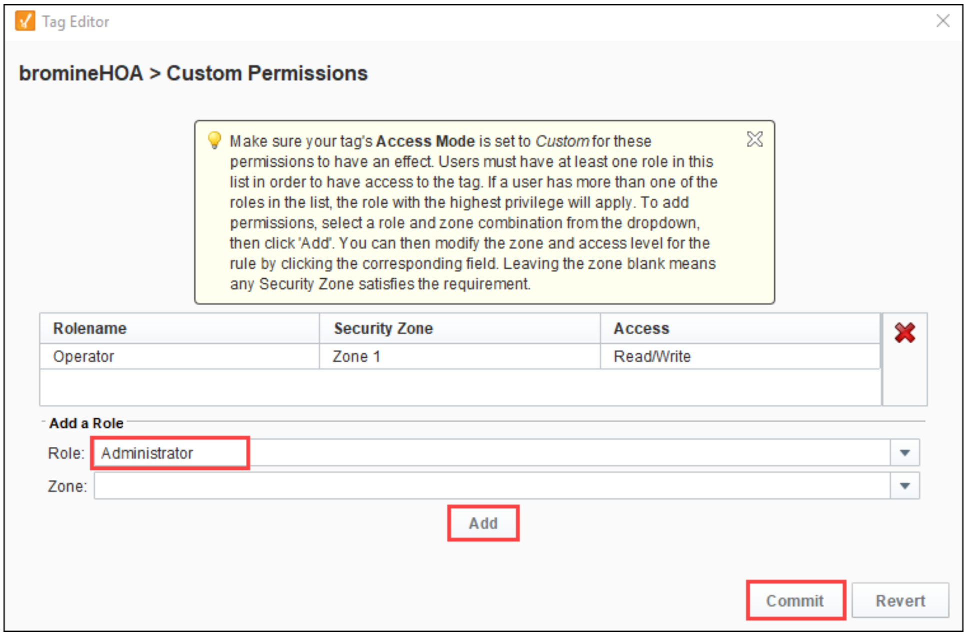
Property	Value
Deadband Mode	Absolute
Scale Mode	Off
Engineering Units	
Engineering Low Limit	0.0
Engineering High Limit	100.0
Engineering Limit Mode	No_Clamp
Format String	#,##0.##
Meta Data Properties	
Tooltip	
Documentation	
Security	
Access Rights	Custom Read Only Read/Write Custom
Custom Permissions	
Scripting	
Tag Event Scripts	
Alarms	
Alarms	No alarms
Alarm Eval Enabled	true
History	

3. The **Custom Permissions** option will now appear under Security. By default, it is set to "No Rules." Click on the **Edit** ✎ icon to set up permissions.



Property	Value
Meta Data Properties	
Tooltip	
Documentation	
Security	
Access Rights	Custom
Custom Permissions	No Rules
Scripting	
Tag Event Scripts	0 event scripts
Alarms	
Alarms	No alarms
Alarm Eval Enabled	true

4. You can now give specific roles permission to edit the Tag. In the **Role** field, enter the name of the role. In the **Zone** field, enter a zone if you want to limit access to a zone as well.
5. Click **Add**. Repeat for any more roles and zones you want to add.
6. Click **Commit**.



Make sure your tag's **Access Mode** is set to *Custom* for these permissions to have an effect. Users must have at least one role in this list in order to have access to the tag. If a user has more than one of the roles in the list, the role with the highest privilege will apply. To add permissions, select a role and zone combination from the dropdown, then click 'Add'. You can then modify the zone and access level for the rule by clicking the corresponding field. Leaving the zone blank means any Security Zone satisfies the requirement.

Rolename	Security Zone	Access
Operator	Zone 1	Read/Write

Add a Role

Role: Administrator

Zone:

7. Click **OK** to close the Tag Editor.

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

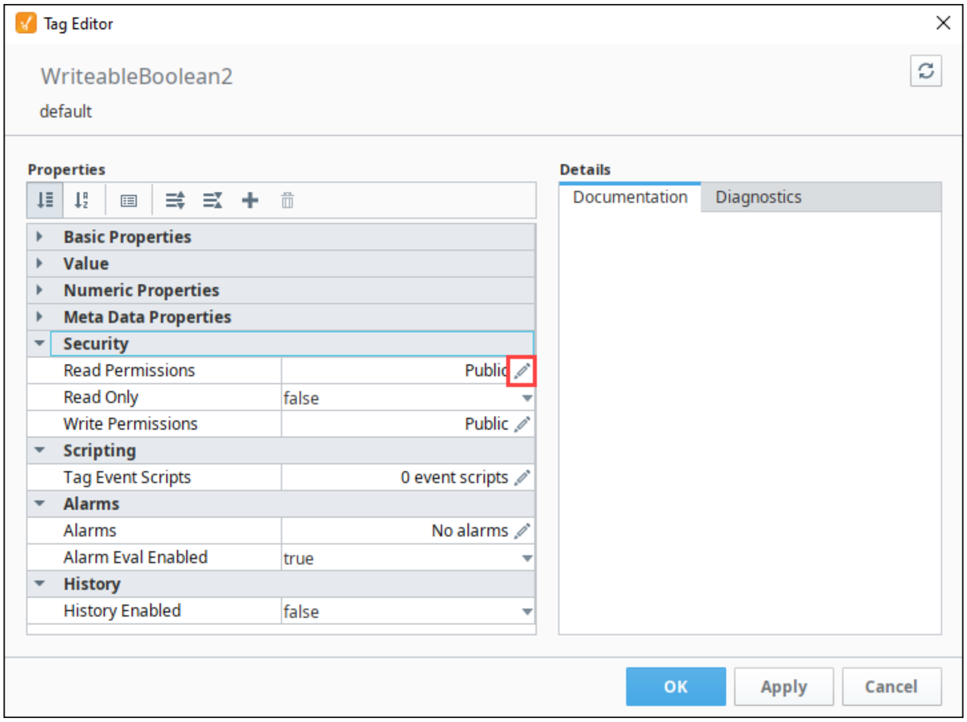
Tag Security after Release 8.0.13

Tag Security has been significantly updated in release 8.0.13. Tag Security now used Security Levels for all Tags, not just in Perspective. Instead of Read/Write and Custom permissions, there are now just Read Permissions and Write Permissions, both of which use security levels. You no longer have to type out role names or security zones. The checkbox tree you are presented with will show you all of the security levels configured in the Gateway Config > Security > Security Levels page.

Read Permissions

Read permissions define the security levels required in order to read values from a Tag. By default, Tags have Read Permissions set to "Public".   You can change the Read security using the Tag Browser in the Designer.

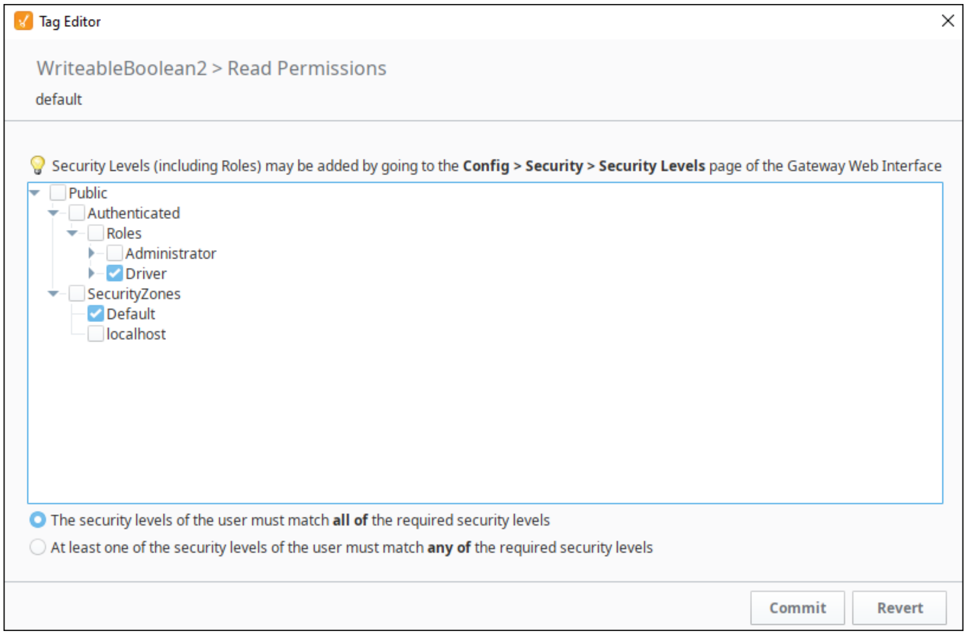
1. In the **Tag Browser**, right-click on the Tag, and select the **Edit Tag** ✎ icon.
2. Scroll down to the Security section. In the Read Permissions section, click the **Edit** ✎ icon.



WriteableBoolean2
default

- Basic Properties
- Value
- Numeric Properties
- Meta Data Properties
- Security
  - Read Permissions: Public
  - Read Only: false
  - Write Permissions: Public
- Scripting
  - Tag Event Scripts: 0 event scripts
- Alarms
  - Alarms: No alarms
  - Alarm Eval Enabled: true
- History
  - History Enabled: false

3. On the screen, choose the security levels you want to have Read permissions for this Tag. In this example, only users with role of Driver will be able to see the Tag value.



WriteableBoolean2 > Read Permissions
default

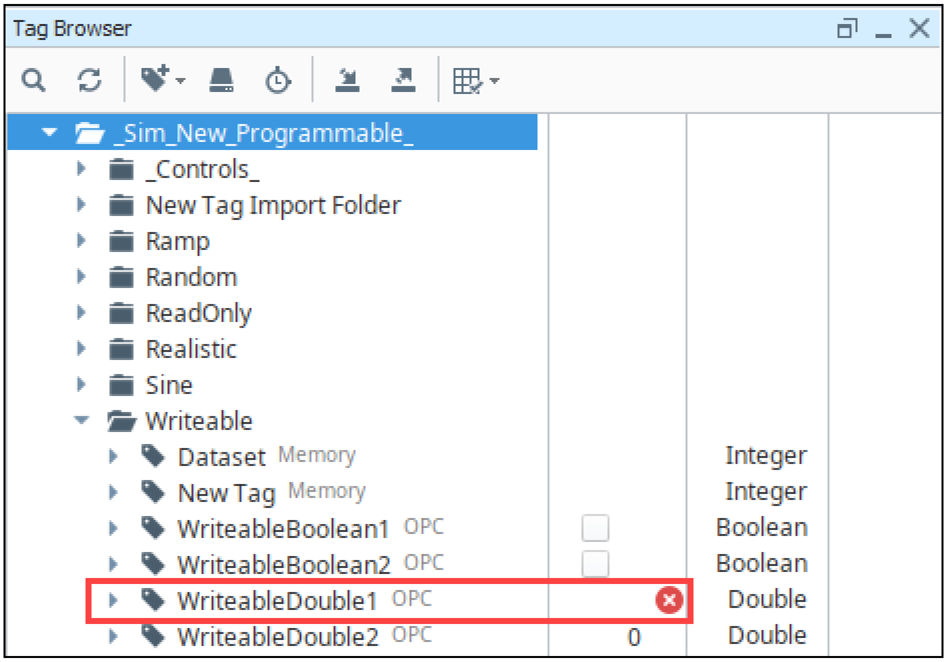
Security Levels (including Roles) may be added by going to the **Config** > **Security** > **Security Levels** page of the Gateway Web Interface

- □ Public
  - □ Authenticated
    - □ Roles
      - □ Administrator
      - ☑ Driver
  - □ SecurityZones
    - ☑ Default
    - □ localhost

● The security levels of the user must match **all of** the required security levels

○ At least one of the security levels of the user must match **any of** the required security levels

4. Click **Commit** to accept the settings.
5. Click **OK** to save the changes to the Tag.
6. If you are logged in as a user other than Driver, you will now see the ✖ icon instead of the Tag's value.



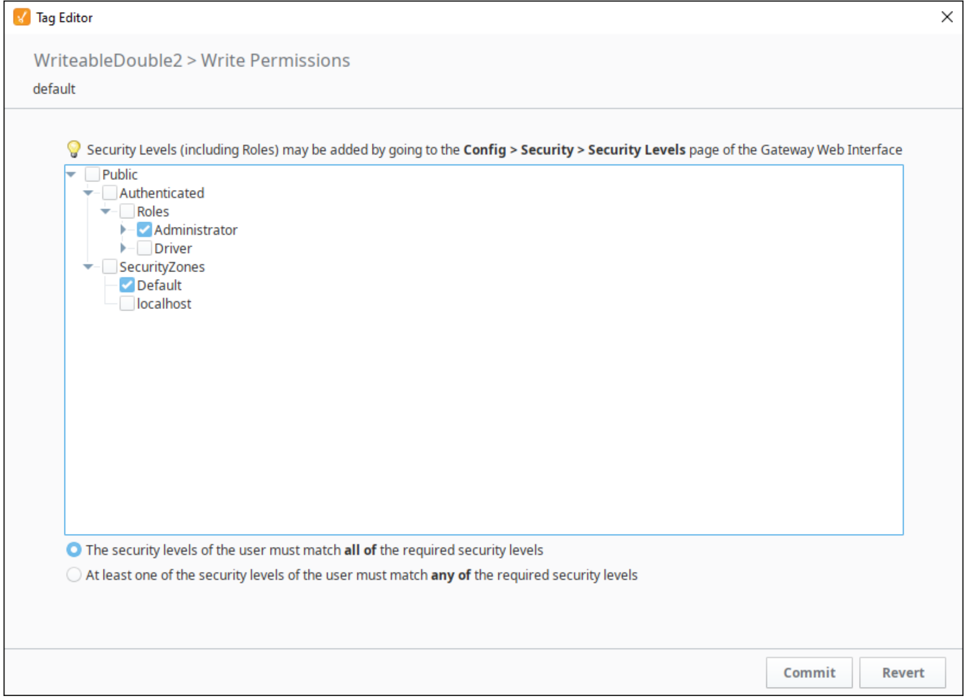
Tag Browser

- _Sim_New_Programmable_
  - _Controls_
  - New Tag Import Folder
  - Ramp
  - Random
  - ReadOnly
  - Realistic
  - Sine
  - Writeable
    - Dataset Memory — Integer
    - New Tag Memory — Integer
    - WriteableBoolean1 OPC — □ — Boolean
    - WriteableBoolean2 OPC — □ — Boolean
    - WriteableDouble1 OPC — ☒ — Double
    - WriteableDouble2 OPC — 0 — Double

Write Permissions

Write permissions define the security levels required in order to read values from a Tag. By default, Tags have Write Permissions set to "Public". You can change the Write security using the Tag Browser in the Designer.

1. In the **Tag Browser**, right-click on the Tag, and select the **Edit Tag** ✎ icon.
2. Scroll down to the Security section. In the Write Permissions section, click the **Edit** ✎ icon.
3. On the screen, choose the security levels you want to have write permissions for this Tag. In this example, only users with role of Administrator will be able to write to the Tag value.



Tag Editor

WriteableDouble2 > Write Permissions

default

Security Levels (including Roles) may be added by going to the **Config** > **Security** > **Security Levels** page of the Gateway Web Interface

- □ Public
  - □ Authenticated
    - □ Roles
      - ☑ Administrator
      - □ Driver
  - □ SecurityZones
    - ☑ Default
    - □ localhost

● The security levels of the user must match **all of** the required security levels

○ At least one of the security levels of the user must match **any of** the required security levels

Migrating Tag Security from Earlier Versions of Ignition

Any existing Tag Security will be migrated to the functional equivalent in the new model on upgrade. This also applies to Tag imports from previous versions.

Related Topics ...

- Audit Log and Profiles
- Quality Overlays
- Permission Properties

Exporting and Importing Tags

Ignition can export and import Tag configurations to and from the **JSON** (JavaScript Object Notation) format. You can import **XML** (Extensible Markup Language) or **CSV** (Comma Separated Value) file formats as well, but Ignition will convert them to **JSON** format while in a Tag provider. Tag export files can be edited directly in any text editor, allowing you to make bulk edits to Tags before importing them back into a Tag provider.

On this page ...

INDUCTIVE UNIVERSITY

Importing and Exporting Tags

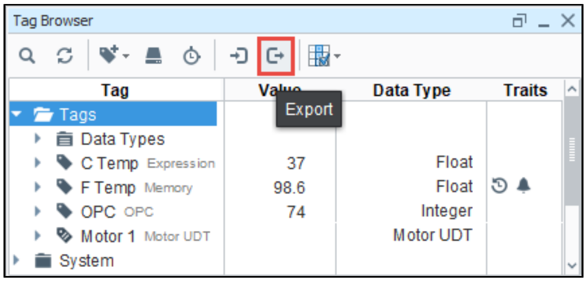
Watch the Video

Export Tags

1. In the **Designer**, go to the **Tag Browser**, and select the Tags folder to export all your Tags. You can also select an individual folder that contains Tags you want to export. You can even export individual Tags as long the individual Tags are in the same folder.

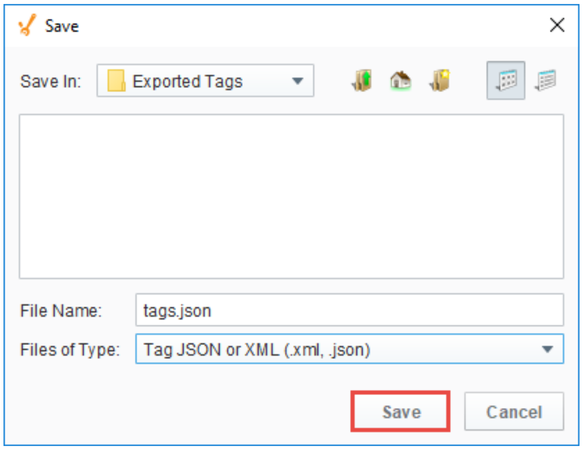
   > ⚠ **Exporting UDTs and UDT Instances**
   >
   > If you select a UDT instance to export, the UDT definition is not automatically included. You must export the definition as a separate file.
   >
   > When later importing these UDTs, it is recommended to import UDT definitions before importing any instances.

2. On the **Tag Browser** toolbar, click the **Export** ↪ icon to export your Tags.



Tag	Value	Data Type	Traits
Tags			
Data Types			
C Temp Expression	37	Float	
F Temp Memory	98.6	Float	⟲ 🔔
OPC OPC	74	Integer	
Motor 1 Motor UDT		Motor UDT	
System			

3. The **Save** window will open. Specify the folder where you want to save your exported Tag files, and then click **Save**. Ignition will export Tags to a **.json** file.



Save In: Exported Tags

File Name: tags.json

Files of Type: Tag JSON or XML (.xml, .json)

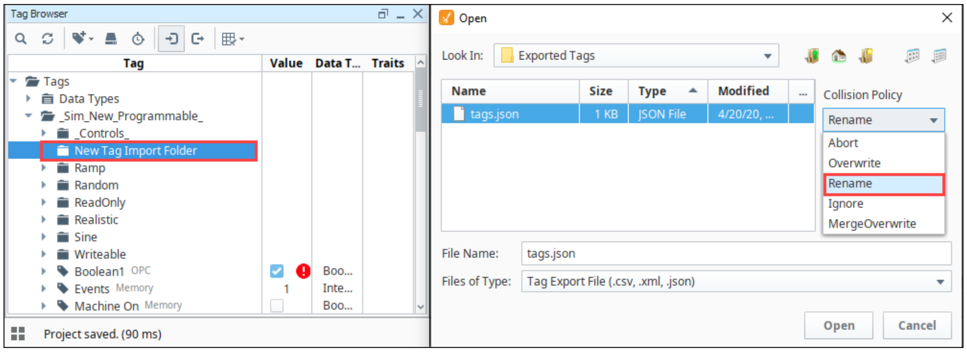
Import Tags

You can import Tags to a individual folder or under the Tags tab. To import Tags under the Tags tab, you can click the Tags tab or the empty space at the bottom of the Tag Browser below all your Tags. When importing Tags, you need to tell Ignition how to handle duplicate Tags. If any of the Tags being imported already exist in the folder you specify, Ignition can abort the import, overwrite the Tags, rename them, ignore them, or merge them. This is called a collision policy.

Collision Policy Options Table

Policy	Description
Abort	Aborts the import if duplicate Tags are found.
Overwrite	Overwrites any Tags in the folder that have the same name as Tags being imported. Note this a complete overwrite of the Tag.
Rename	Renames any duplicate Tags.
Ignore	Ignores duplicate Tags and imports only those that are unique.
MergeOverwrite	Overwrites the Tag with the exception of any properties that aren't defined in the import folder. Those properties will be merged.

1. On the Tag Browser toolbar, click the **Import** ⇥ icon.
2. Specify the folder you want to import your Tags from, and choose a previously exported file either **.json**, **.xml**, or **.csv** file type.
3. Choose a Collision Policy, which indicates how Ignition will deal with duplicate Tags.
4. Click **Open** to import the Tags.



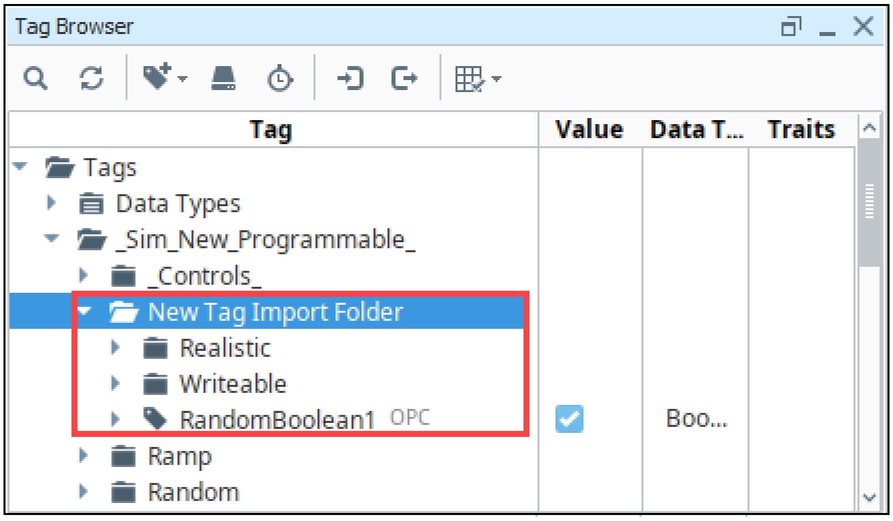
Look In: Exported Tags

Collision Policy: Rename

File Name: tags.json

Files of Type: Tag Export File (.csv, .xml, .json)

5. The Tags now appear in the Tag Browser in the folder you indicated for the import.



Tag File Formats

Tags can be imported from CSV, JSON, and XML. Tags can only be exported in XML or JSON. There are many configuration settings for Tags than what is displayed in a JSON or XML export file. The Tag export feature only exports the configuration properties that have been edited in at least one of the Tags in the selected export folder. Therefore, to ensure the desired configuration setting is available in the export file, at least one Tag within the selected export folder must have that configuration property changed.

CSV Format

Legacy Ignition systems exported Tags as a CSV formatted file. This functionality was deprecated in Ignition 7.6.0.

Modern Gateways can not export Tags to a CSV file, but they can still import Tags from this legacy CSV format. This is offered as a means to support those migrating to Ignition from a third-party system. Note that importing Tags from the CSV format in this manner will not create any alarms on the imported Tags.

Below is an example of the legacy CSV format, which contains a couple of different Tag types, purely for demonstrative purposes. It contains:

- An OPC Tag
- A Folder
- An OPC Tag located in a folder
- A Derived Tag
- An Expression Tag
- A Memory Tag
- A Query Tag

```
Path,Name,Owner,TagType,DataType,Value,Enabled,AccessRights,OPCServer,OPCItemPath,ScanClass,DriverName,
ScaleMode,RawLow,RawHigh,ScaledLow,ScaledHigh,ClampMode,ScaleFactor,Deadband,DeadbandMode,FormatString,
EngUnit,EngLow,EngHigh,EngLimitMode,Tooltip,Documentation,ExpressionType,Expression,OPCWriteBackServer,
OPCWriteBackItemPath,SQLBindingDatasource,HistoryEnabled,PrimaryHistoryProvider,HistoricalScanclass,
HistoricalDeadband,HistoricalDeadbandMode,InterpolationMode,HistoryMaxAgeMode,HistoryMaxAge,
HistoryTimestampSource,UDTParentType,PersistValue,SourceDataType,SourceTagPath,SQLBindingPollRate,
Permissions
# version=1,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
,_types_,,6,2,,TRUE,Read_Write,,,Default,,0,0,100,0,10,0,0,1.00E-04,0,"#,##0.##",,0,100,0,,,0,,,,,FALSE,,
Default Historical,0.01,0,3,0,1,0,,FALSE,,,,
,A Folder,,6,2,,TRUE,Read_Write,,,Default,,0,0,100,0,10,0,0,1.00E-04,0,"#,##0.##",,0,100,0,,,0,,,,,
FALSE,,Default Historical,0.01,0,3,0,1,0,,FALSE,,,,
A Folder/,OPC in a folder,,0,2,,TRUE,Read_Write,,,Default,,0,0,100,0,10,0,0,1.00E-04,0,"#,##0.##",,
0,100,0,,,0,,,,,FALSE,,Default Historical,0.01,0,3,0,1,0,,FALSE,,,,
,Derived Tag,,13,2,100,TRUE,Read_Write,,,Default,,0,0,100,0,10,0,0,1.00E-04,0,"#,##0.##",,0,100,0,,,
0,,,,,FALSE,,Default Historical,0.01,0,3,0,1,0,,FALSE,,[~]Expression Tag,,
,Expression Tag,,1,2,100,TRUE,Read_Write,,,Default,,0,0,100,0,10,0,0,1.00E-04,0,"#,##0.##",,0,100,0,,,1,"
//This is an expression
100",,,,FALSE,,Default Historical,0.01,0,3,0,1,0,,FALSE,,,,
,Memory Tag,,1,7,I'm a memory Tag,TRUE,Read_Write,,,Default,,0,0,100,0,10,0,0,1.00E-04,0,"#,##0.##",,
0,100,0,,,0,,,,,FALSE,,Default Historical,0.01,0,3,0,1,0,,FALSE,,,,
,OPC Tag,,0,2,,TRUE,Read_Write,Ignition OPC-UA Server,[devicename]folder/path,Default,,
0,0,100,0,10,0,0,1.00E-04,0,"#,##0.##",,0,100,0,,,0,,,,,FALSE,,Default Historical,0.01,0,3,0,1,0,,
FALSE,,,,
,Query Tag,,1,2,,TRUE,Read_Write,,,Default,,0,0,100,0,10,0,0,1.00E-04,0,"#,##0.##",,0,100,0,,,2,"/*Some
Query*/
SELECT 100",,,,FALSE,,Default Historical,0.01,0,3,0,1,0,,FALSE,,,,
```

Property Values in the CSV Import

The following table shows the configuration property names and values contained in legacy CSV Tag import files. Tags were overhauled in Ignition 8.0, so the properties listed here are not the same as Tags export from an Ignition 8.0+ system. For modern tag properties, see the Tag Properties and Tag Property Enum Reference pages.

Tag Properties

Property Name	Type	Values (if applicable)	Description

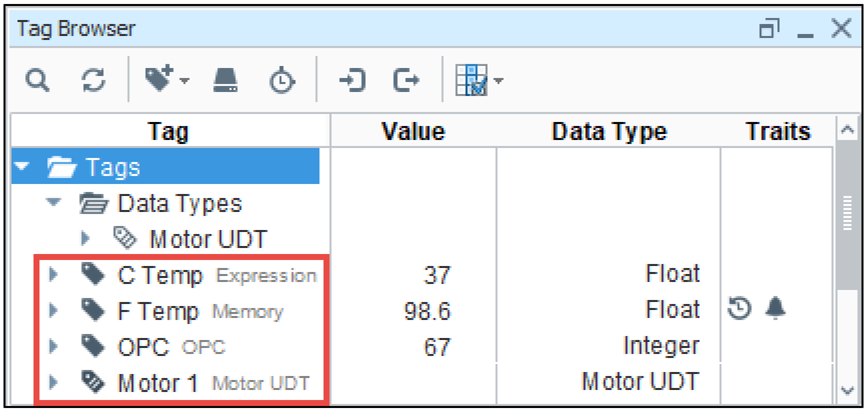
Value			The value of the Tag, dependent on the data type.
Data Type	Int	0 - Int1 1 - Int2 2 - Int4 3 - Int8 4 - Float4 5 - Float8 6 - Boolean 7 - String 8 - DateTime 9 - DataSet	
Enabled	Boolean	true/false	
Tagtype	Int	0 - OPC Tag 1 - DB Tag (see ExpressionType) 2 - Client Tag 6 - Folder 13 - Derived Tag	Determines the type of the Tag. A value of one is a "DB Tag", which is either a Memory Tag, Query Tag, or Expression Tag, depending on the value of the ExpressionType field.
ExpressionType	Int	0 - None 1 - Expression 2 - SQL Query	Used in conjunction when the TagType is set to 1 (DB Tag), otherwise this field is ignored.
AccessRights	Int	0 - Read Only 1 - Read/Write 2 - Custom	If custom, will be defined by a **Permissions** Tag.
OPCServer	String		
OPCItemPath	String		
OPCWriteBackServer	String		Write back target for expression Tags.
OPCWriteBackItemPath	String		
ScaleMode	Int	0 - Off 1 - Linear 2 - Square Root 3 - Exponential Filter	
ScaleFactor	Float		For exponential filter.
RawLow	Float		Defines scale range.
RawHigh	Float		
ScaledLow	Float		
ScaledHigh	Float		
ClampMode	Int	0 - None 1 - Low 2 - High 3 - Both	
Deadband	Float		
DeadbandMode	Int	0 - Absolute 1 - Percentage	
FormatString	String		
EngUnit	String		
EngLow	Float		
EngHigh	Float		
EngLimitMode	Int	0 - None 1 - Low 2 - High 3 - Both	

Tooltip	String		
Documentation	String		
DriverName	String		Used for external Tags.
ScanClass	String		The export will only include the name of the Scanclass, not the configuration of the Scanclass itself. A Scanclass with the same name needs to already exist on the Gateway that the Tags are being imported to, prior to importing them.
HistoryEnabled	Boolean	true/false	
PrimaryHistoryProvider	String		The history provider to use if storing history.
HistoricalDeadband	Float		
HistoricalDeadbandMode	Int	0 - Absolute 1 - Percentage	
HistoricalScanclass	String		
InterpolationMode	Int	0 - Discrete 2 - Analog (deadband) 3 - Analog (compressed)	How values are interpolated. 2 exists for backwards compatibility (and is equivalent to 1), but only 0 or 3 should be used in the future.
HistoryTimestampSource	Int	0 - System 1 - Value	
HistoryMaxAgeMode	Int	0 - Unlimited 1 - Limited	
HistoryMaxAge	Int		Max cycles between storage.
UDTParentType	String		The path to the parent UDT type. Used by sub-types and instances.

JSON Example

In this example, we exported all Tags in our Tag Browser (with the exception of the Motor UDT). The following Tags were exported.

- C Temp - Expression Tag with an Alarm
- F Temp - Memory Tag with History Settings
- OPC - OPC Tag
- Motor 1 - UDT Instance



Tag	Value	Data Type	Traits
Tags			
Data Types			
Motor UDT			
C Temp Expression	37	Float	
F Temp Memory	98.6	Float	
OPC OPC	67	Integer	
Motor 1 Motor UDT		Motor UDT	

The following exported Tag file is in JSON format. As you browse through the JSON file, you will see the Tag properties and configuration settings for each of the four Tags listed above.

Below is an example of a JSON format Tag export. Descriptions on the various properties can be found on the Tag Properties and Tag Property Enum Reference pages.

Exported Tags in JSON Format

```
{
"name": "",
  "tagType": "Provider",
  "tags": [
    {
      "valueSource": "expr",
      "expression": "(5/9) * ({[~]F Temp} - 32)",
      "dataType": "Float4",
      "sourceTagPath": "[ExportTags]F Temp",
      "name": "C Temp",
      "tagType": "AtomicTag"
    },
    {
      "valueSource": "opc",
      "opcItemPath": "ns\u003d1;s\u003d[Generic]_Meta:Sine/Sine0",
      "name": "OPC",
      "tagType": "AtomicTag",
      "opcServer": "Ignition OPC UA Server"
    },
    {
      "name": "Motor 1",
      "typeId": "Motor UDT",
      "parameters": {
        "MotorNumber": "1"
      },
      "tagType": "UdtInstance",
      "tags": [
        {
          "name": "Amps",
          "tagType": "AtomicTag"
        },
        {
          "name": "HOA",
          "tagType": "AtomicTag"
        }
      ]
    },
    {
      "name": "_types_",
      "tagType": "Folder",
      "tags": [
        {
          "name": "Motor UDT",
          "parameters": {
            "MotorNumber": ""
          },
          "tagType": "UdtType",
          "tags": [
            {
              "opcItemPath": {
                "bindType": "parameter",
                "binding": "ns\u003d1;s\u003d[Dairy]_Meta:Overview/Motor {MotorNumber}/Amps"
              },
              "valueSource": "opc",
              "dataType": "Int4",
              "name": "Amps",
              "tagType": "AtomicTag",
              "opcServer": "Ignition OPC UA Server"
            },
            {
              "opcItemPath": {
                "bindType": "parameter",
                "binding": "ns\u003d1;s\u003d[Dairy]_Meta:Overview/Motor {MotorNumber}/HOA"
              },
              "valueSource": "opc",
              "dataType": "Int4",
              "name": "HOA",
              "tagType": "AtomicTag",
              "opcServer": "Ignition OPC UA Server"
            }
          ]
        }
      ]
    },
    {
      "valueSource": "memory",
      "historyProvider": "MySQL",
      "dataType": "Float4",
      "alarms": [
        {
          "mode": "OutsideValues",
          "setpointA": 25.0,
          "setpointB": 75.0,
          "name": "Alarm"
        }
      ],
      "name": "F Temp",
      "historyEnabled": true,
      "value": 98.6,
      "tagType": "AtomicTag",
      "sampleMode": "TagGroup"
    }
  ]
}
```

XML Example

In this example, we exported the same four Tags from our Tag Browser, that were also used in the JSON example, in XML format. As you browse through the XML file, you will see the Tag properties and configuration settings for each of the same four Tags.

Below is an example of an XML format tag export. Descriptions on the various properties can be found on the Tag Properties and Tag Property Enum Reference pages.

Exported Tags in XML Format

```
<Tags MinVersion="8.0.0">
   <Tag name="" type="Provider">
      <Tags MinVersion="8.0.0">
         <Tag name="C Temp" type="AtomicTag">
            <Property name="valueSource">expr</Property>
            <Property name="expression">(5/9) * ({[~]F Temp} - 32)</Property>
            <Property name="dataType">4</Property>
            <Property name="sourceTagPath">[ExportTags]F Temp</Property>
         </Tag>
         <Tag name="OPC" type="AtomicTag">
            <Property name="valueSource">opc</Property>
            <Property name="opcItemPath">ns=1;s=[Generic]_Meta:Sine/Sine0</Property>
            <Property name="opcServer">Ignition OPC UA Server</Property>
         </Tag>
         <Tag name="Motor 1" type="UdtInstance">
            <Property name="typeId">Motor UDT</Property>
            <Parameters>
               <Property name="MotorNumber" type="String">1</Property>
            </Parameters>
            <Tags MinVersion="8.0.0">
               <Tag name="Amps" type="AtomicTag">
               </Tag>
               <Tag name="HOA" type="AtomicTag">
               </Tag>
            </Tags>
         </Tag>
         <Tag name="_types_" type="Folder">
            <Tags MinVersion="8.0.0">
               <Tag name="Motor UDT" type="UdtType">
                  <Parameters>
                     <Property name="MotorNumber" type="String"></Property>
                  </Parameters>
                  <Tags MinVersion="8.0.0">
                     <Tag name="Amps" type="AtomicTag">
```

```
                        <Property name="opcItemPath">{bindType=parameter, binding=ns=1;s=[Dairy]_Meta:
Overview/Motor {MotorNumber}/Amps}</Property>
                        <Property name="valueSource">opc</Property>
                        <Property name="dataType">2</Property>
                        <Property name="opcServer">Ignition OPC UA Server</Property>
                     </Tag>
                     <Tag name="HOA" type="AtomicTag">
                        <Property name="opcItemPath">{bindType=parameter, binding=ns=1;s=[Dairy]_Meta:
Overview/Motor {MotorNumber}/HOA}</Property>
                        <Property name="valueSource">opc</Property>
                        <Property name="dataType">2</Property>
                        <Property name="opcServer">Ignition OPC UA Server</Property>
                     </Tag>
                  </Tags>
               </Tag>
            </Tags>
         </Tag>
         <Tag name="F Temp" type="AtomicTag">
            <Property name="valueSource">memory</Property>
            <Property name="historyProvider">MySQL</Property>
            <Property name="dataType">4</Property>
            <CompoundProperty name="alarms">
               <PropertySet>
                  <Property name="mode">5</Property>
                  <Property name="setpointA">25.0</Property>
                  <Property name="setpointB">75.0</Property>
                  <Property name="name">Alarm</Property>
               </PropertySet>
            </CompoundProperty>
            <Property name="historyEnabled">true</Property>
            <Property name="value">98.6</Property>
            <Property name="sampleMode">2</Property>
         </Tag>
      </Tags>
   </Tag>
</Tags>
```

Related Topics ...

- Tag Data Types
- Tag Properties

Alarming

On this page

...

Overview

Alarming is a core feature of the Ignition platform. It provides the functionality and flexibility to configure your alarms, provides up-to-date status of alarms, stores alarm history, builds the logic for how, why, and when alarm notifications are delivered, manages alarm notifications for user groups, and sends Email, SMS or Voice notifications. With all these features and functions in Alarming, you can easily create alarms, and design and manage your alarm notifications any way you choose.

You can set up alarms quickly, or you can customize an entire alarm notification process with features such as pipelines, schedules, and rosters. The Alarming features and functions translate to convenience and flexibility for your site's alarms and notifications.

- **Alarm Settings** - There are a host of alarm settings so you can build and change the behavior of your alarm.
- **Alarm Status** - Displays current status of alarms.
- **Alarm Journal** - Stores historical information about alarms.
- **Alarm Notification** - Lets you select the delivery channel for how alarms notifications are sent via Email, SMS, or Voice.
- **Users, Schedules, and Rosters** - Defines the users who will receive alarms based on user schedules and on-call rosters.
- **Alarm Notification Pipelines** - Lets you define how alarm notifications are sent out and acknowledged.

Alarms can be configured on Tags or OPC items in SQL Bridge (Transaction Groups). The different Tag types that you can configure alarms on include Memory Tags, Query Tags, Expression Tags, as well as Tags inside of a UDT. You can also put alarms on System Tags that Ignition inherently provides such as the Gateway Performance and CPU Usage.

Configuring Alarms

Alarm configuration in Ignition is flexible and highly customizable to your needs. You can configure alarms with one alarm on a Tag or multiple alarms on a Tag. You can add alarms in UDTs so every instance of that Tag will automatically have alarms configured when a new instance of your Tag is created. You can use the alarm settings to create alarms that equal or don't equal a setpoint, above or below a setpoint, between setpoints, outside setpoints, dynamic setpoints, out of range, bad quality, etc. Alarms can be configured for any alarm condition imaginable.

More information about configuring alarms can be found in Alarming Properties and Configuring Alarms pages.

The image below shows an alarm configured on an OPC Tag. You can see that an alarm has quite a few properties including alarm mode settings where you can set specific alarm attibute values.

? Unknown Attachment

How to Monitor Alarm Status

Setting up the Alarm Status Table is quick and easy. Drag the Alarm Status Table component onto your window, and the current alarms are displayed immediately into one view. The Alarm Table is highly customizable and alarms can be configured to show active, unacknowledged, cleared, and acknowledged alarms. You can Acknowledge or Shelve, and you can filter on Alarm Status properties, show or hide alarm property information, reorganize columns, or view alarm details, notes, and history of an alarm.

? Unknown Attachment

Alarm Status Tags

Ignition provides a set of System Tags to view information about the Ignition server which includes four Tags that count the number of alarms in each state. A quick way to see if any alarms are currently active and get an alarm count is to add a Label component on the Navigation window. The four system alarm states are:

- Active and Unacknowledged
- Active and Acknowledged
- Clear and Acknowledged
- Clear and Unacknowledged

In the **Tag Browser** of the Designer, scroll down to the **System > Gateway > Alarming** folder. You can see all four of the system Tags that Ignition provides. You can also see how many alarms are currently **Active and Acked**, **Active and Unacked**, **Clear and Acked**, and **Clear and Unacked**.

These system Tags can easily be used to visualize all alarms in the system.

How to View Alarm History

The Alarm Journal stores historical information about alarms in a database. It stores basic data about alarms that have occurred, such as their source and timestamp, associated data on an alarm, and the values of an alarm's properties at the time the event occurred. It captures all status changes for each alarm, as well as acknowledgement to an external SQL database of your choosing. To begin viewing alarm history, all you need to do is create an Alarm Journal Profile in the Gateway webpage.

Like the Alarm Status Table, the Alarm Journal enables you to filter on alarm history properties. The alarms are color coded so you know what each status represents. The Date Range is a very common filter type since users typically want to filter for alarm events within a specific period of time.

Alarm Notifications

Alarms can also generate notifications that are delivered to users allowing Ignition to immediately communicate events and problems to your users. Alarm Notification Pipelines control how and when notifications are sent to users. You can select the delivery channel for how alarms are sent: Email, SMS, or Voice. The notification system has access to Ignition's Authentication Profiles so users can easily be added to notification On-Call Rosters. Schedules can be created allowing users to receive notifications only when on-schedule, so there is no need to worry about notifying a supervisor or manager when they are not on-site, or if it's in the middle of the night. Pipelines coupled with on-call rosters and schedules allow you to build your own custom alarm notification process.

On-Call Rosters

The On-Call Roster is a collection of users that are notified when an alarm occurs. When an alarm is triggered, a notification is sent to a designated On-Call Roster where it evaluates the users schedules, and only notifies those users that have an active schedule.

Roster Management from the Vision Client Window

Schedules

Schedules define the times of users on-call availability and unavailability. You can set up a schedule for each user in the alarm notification system so users only recieve notifications when they have an active schedule.

Schedule from the Vision Client Window

Alarm Notification Pipelines

The Alarm Notification Pipeline feature is an innovative tool that lets you easily create routes for your alarms. By designing your own alarm notification routing, you have control of what happens when an alarm goes active, when an alarm is sent out, and who receives the alarm notification.

The alarm notification pipeline has a simple drag-and-drop interface so you can build various types of alarm logic. Its built-in pipeline block functionality lets you delay sending alarm notifications, escalate higher priority alarms to a different group of users, consolidate multiple alarms so recipients aren't flooded with messages, and distribute specific alarm types to different contact groups.

Alarm notification pipelines can be very simple to very complex. In this simple notification pipeline, when an alarm is triggered, the people listed in the On-call Roster are notified via Email. If no one acknowledges the alarm in 20 seconds, the alarm notification is routed back to the same users listed in the On-Call Roster.

? Unknown Attachment

In this more complex alarm notification pipeline, if an operator doesn't respond to the alarm after three attempts, the pipeline jumps it to another pipeline (possibly an escalation pipeline).

? Unknown Attachment

To learn more about building your own pipelines, go to Alarm Notification Pipelines.

Related Topics ...

- Configuring Alarms
- Using Alarm Status Tags in Vision
- Alarm Pipeline Designer Interface
- Pipeline Blocks

In This Section ...

Alarm Journal

By default, current alarm data is only stored in memory, and a finite number of events are retained for each alarm. Fortunately, Ignition can easily be configured to store alarm data into a SQL database with an Alarm Journal Profile. The journal can store basic data about alarms that have occurred, such as their source and timestamp, associated data on the alarm, and the values of the alarm's properties at the time the event occurred. The Alarm Journal is used by the **Alarm Journal Table** component, and can be accessed through scripting functions and direct database queries.

The Gateway can have more than one Alarm Journal. Alarm Journals have options to filter which Alarms are stored in the journal, therefore, by having more than one alarm journal configured on the Gateway, it is possible to store some alarms in one journal, and different alarms in another journal. Once configured, the journal can be accessed by the **Alarm Journal Table** component, scripting functions, or direct database queries.

Alarm Journals can store data in one of three ways, and store data indefinitely unless a Data Pruning value is set:

- In a database, using an existing database connection from the Gateway
- Remotely, using another Ignition Gateway's Alarm Journal profile
- Internally, storing alarm information into the Ignition install directory

It is strongly encouraged to set a Data Pruning value for Internal Alarm Journal Profiles. Otherwise, it could cause your computer to run out of hard drive space.

⚠ **Configuring an Alarm Journal**

You must have an Alarm Journal Profile created and have a valid database connection to use the Alarm Journal Table.

On this page ...

Creating an Alarm Journal Profile

In Ignition, an Alarm Journal stores historical information about alarms in a database. It can store basic data about alarms that have occurred, such as source and timestamp, along with associated data on the alarm, and the values of the alarm's properties at the time the event occurred. You can choose to store your alarm data on an external database or automatically send it to a remote gateway's Alarm Journal to be logged. Both options are described in the sections below.

Create a single Alarm Journal Profile to store all of your alarms, or create multiple journals to store alarms across multiple databases. Each journal stores alarms based on the filters you set up and can prune data automatically after a set time limit.

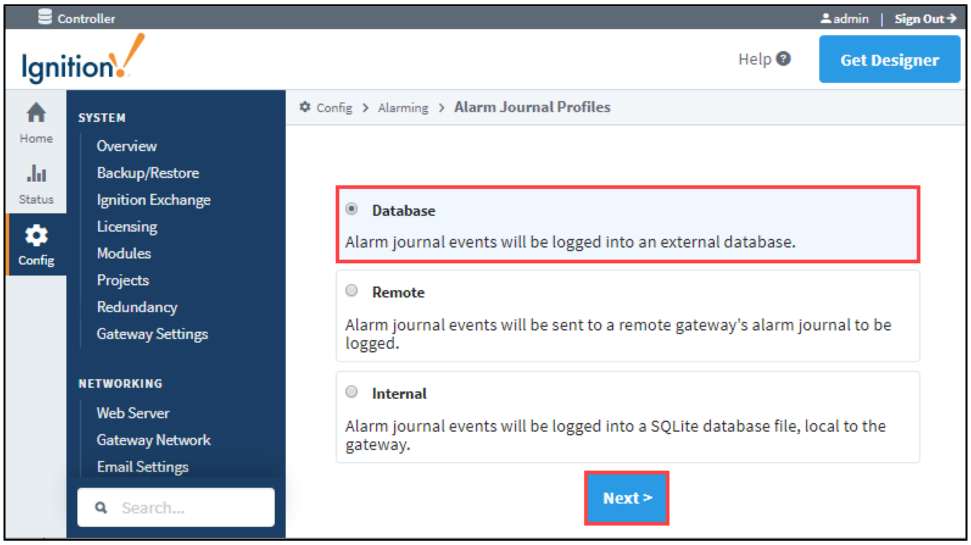
Create Alarm Journal Profile

Watch the Video

Create an Alarm Journal to Log Events to an External Database

1. Go to the **Config** section of the Gateway webpage.
2. Choose **Alarming > Journal** from the menu on the left.
3. Look for the blue arrow and click on **Create new Alarm Journal Profile...**. The Alarm Journal Profiles screen will be displayed.

4. You have the option of logging alarm journal events to an external database, logging locally, or sending them to a remote gateway's Alarm Journal. In this example, select **Database**, and click **Next.** (Configuring alarm journal events to be sent to a remote gateway is addressed in Create an Alarm Journal to Log Events to a Remote Gateway section on this page.

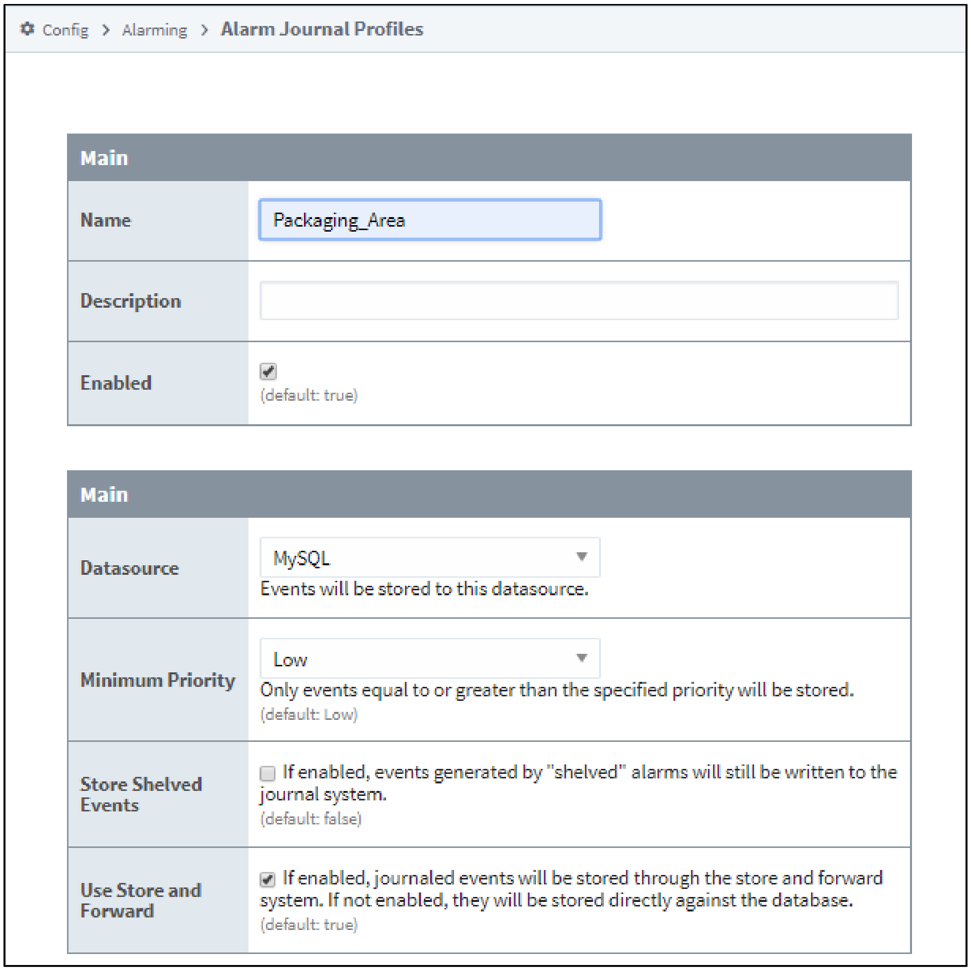


● **Database**
Alarm journal events will be logged into an external database.

○ **Remote**
Alarm journal events will be sent to a remote gateway's alarm journal to be logged.

○ **Internal**
Alarm journal events will be logged into a SQLite database file, local to the gateway.

5. Enter the **Name** of your alarm journal profile. The default name is 'Journal.' Most of the fields have default settings. Refer to the journal properties table below for setting descriptions, and update as necessary. Click the **Create New Alarm Journal Profile** button at the bottom of the page. Once completed, the tables will be created for you once an alarm event occurs.



Config > Alarming > Alarm Journal Profiles

Main

Name	Packaging_Area
Description	
Enabled	☑ (default: true)

Main

Datasource	MySQL Events will be stored to this datasource.
Minimum Priority	Low Only events equal to or greater than the specified priority will be stored. (default: Low)
Store Shelved Events	☐ If enabled, events generated by "shelved" alarms will still be written to the journal system. (default: false)
Use Store and Forward	☑ If enabled, journaled events will be stored through the store and forward system. If not enabled, they will be stored directly against the database. (default: true)

Using Multiple Alarm Journals

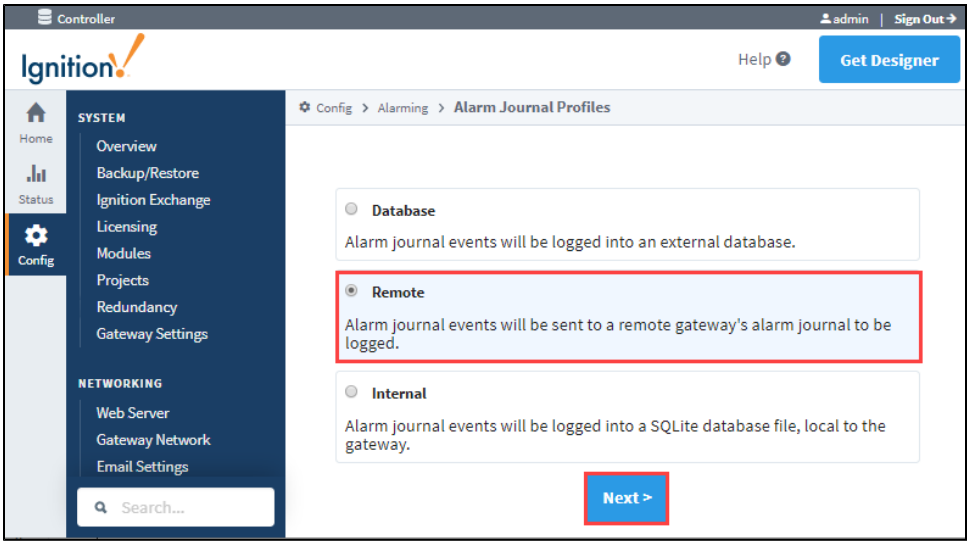
If you only have one alarm journal specified on your Gateway, then you do not need to specify the journal name on the Journal Name property. Ignition will set this for you. If you have more than one alarm journal created, then you need to provide the name of the journal you'd like to query in the Journal Name component property of the Property Editor.

The following feature is new in Ignition version **8.0.7**
Click here to check out the other new features

Create an Alarm Journal to Log Events to a Remote Gateway

Just like configuring alarm journal events to be logged into a database, it is done from the Gateway Webpage, **Config > Alarming > Journal**.

1. To have your alarm journal events automatically sent to a remote gateway's alarm journal, select **Remote**, and click **Next**.



○ **Database**

Alarm journal events will be logged into an external database.

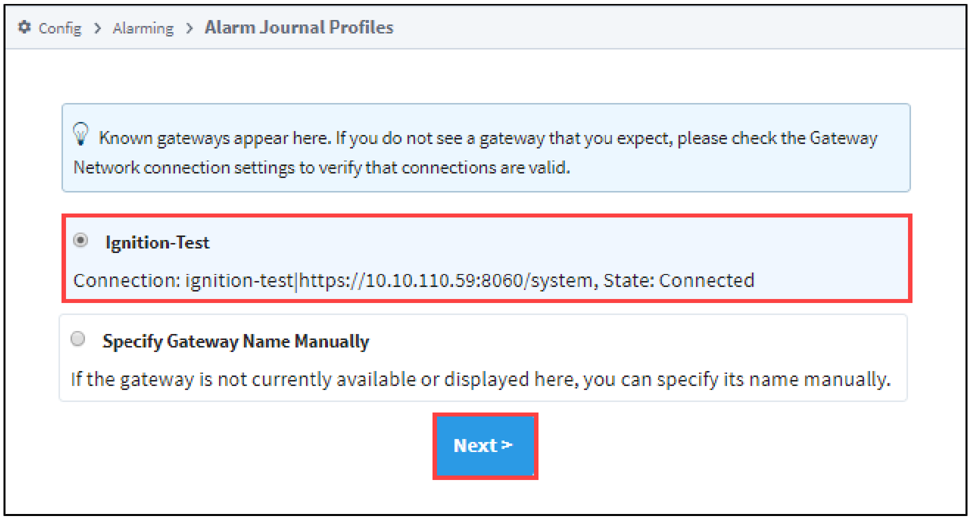
● **Remote**

Alarm journal events will be sent to a remote gateway's alarm journal to be logged.

○ **Internal**

Alarm journal events will be logged into a SQLite database file, local to the gateway.

2. A list of known Gateways will be displayed. If you don't see a gateway that you expected to see, check your Gateway Network settings to verify that the connections are valid. You also have the option to specify a gateway manually. This example selects a valid gateway. Click **Next**.



Known gateways appear here. If you do not see a gateway that you expect, please check the Gateway Network connection settings to verify that connections are valid.

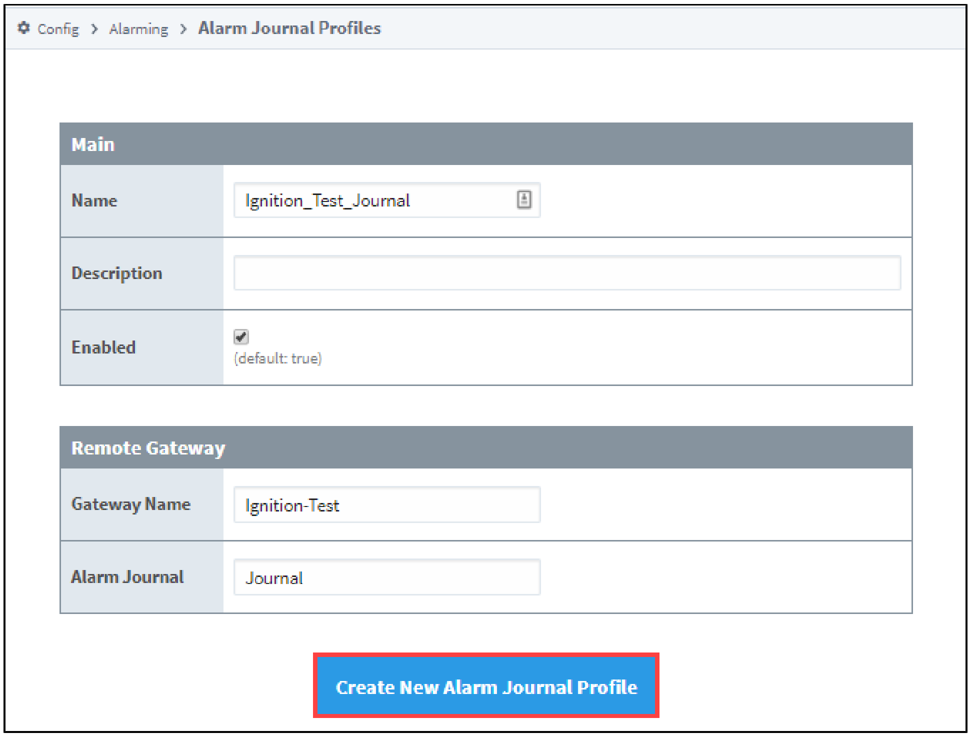
● **Ignition-Test**

Connection: ignition-test|https://10.10.110.59:8060/system, State: Connected

○ **Specify Gateway Name Manually**

If the gateway is not currently available or displayed here, you can specify its name manually.

3. If an Alarm Journal exists on the remote gateway, the fields will autopopulate. The name of the gateway and the Alarm Journal Profile name will appear in the **Name** field prefaced with the alarm journal profile name,(i.e., Ignition_Test_Journal), as shown in the following example. Click **Create New Alarm Journal Profile**.



Config > Alarming > Alarm Journal Profiles

Main

Name: Ignition_Test_Journal

Description:

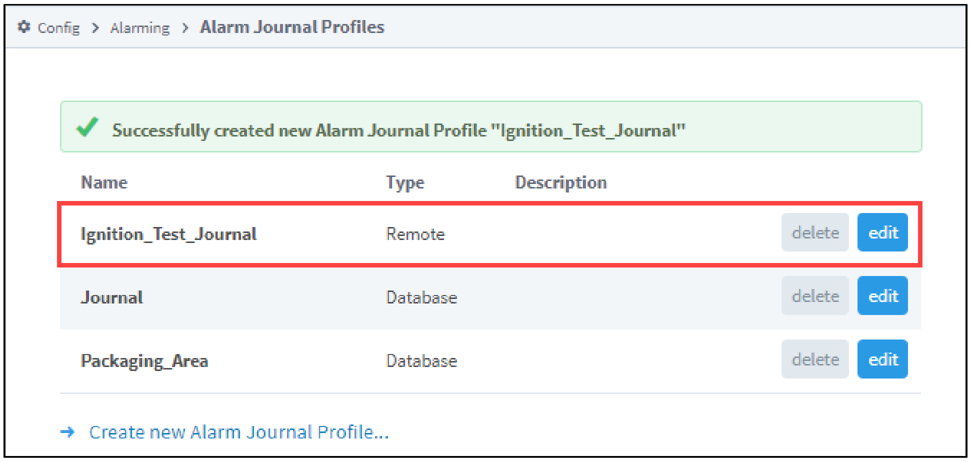
Enabled: ☑ (default: true)

Remote Gateway

Gateway Name: Ignition-Test

Alarm Journal: Journal

4. You will receive a successful message stating your new Alarm Journal Profile was created.



Config > Alarming > Alarm Journal Profiles

✓ Successfully created new Alarm Journal Profile "Ignition_Test_Journal"

Name	Type	Description	
Ignition_Test_Journal	Remote		delete edit
Journal	Database		delete edit
Packaging_Area	Database		delete edit

Remote Gateway Alarm Journal Properties Table

Main	
Name	The default name, is the name of the Remote Gateway and Alarm Journal.
Description	Description of the journal profile. Optional
Enabled	By default. The journal profile is enabled.
Remote Gateway	
Gateway Name	Name of the Remote Gateway.
Alarm Journal	Name of the Alarm Journal on the Remote Gateway.

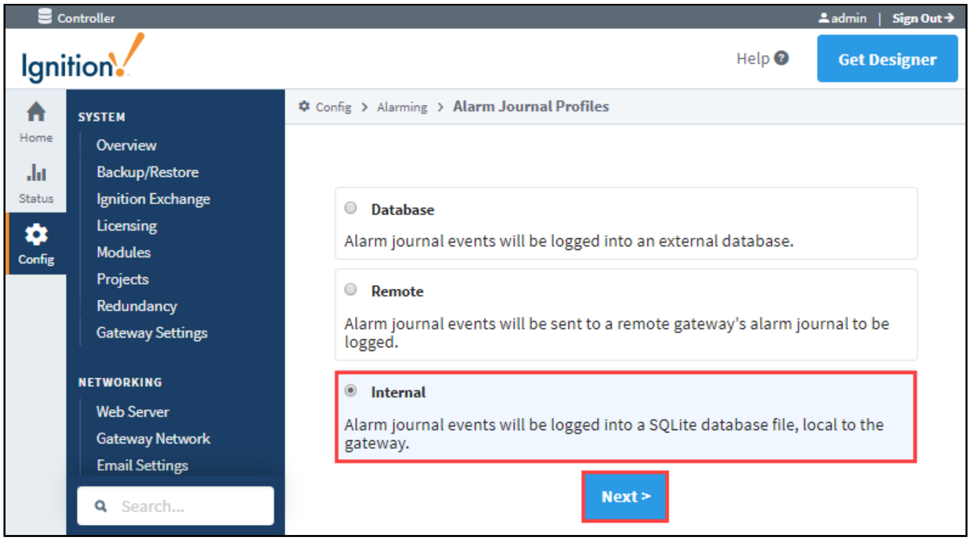
The following feature is new in Ignition version **8.0.8**
Click here to check out the other new features

Create an Internal Alarm Journal to Log Events Locally

Ignition Gateways can now create an Internal Alarm Journal Profile that stores Journal entries locally.

Go to the Gateway webpage, **Config > Alarming > Journal** to create the Internal alarm journal profile.

1. Click on the the **Create new Alarm Journal Profile...** link.
2. Select **Internal** to have your alarm journal events logged locally.

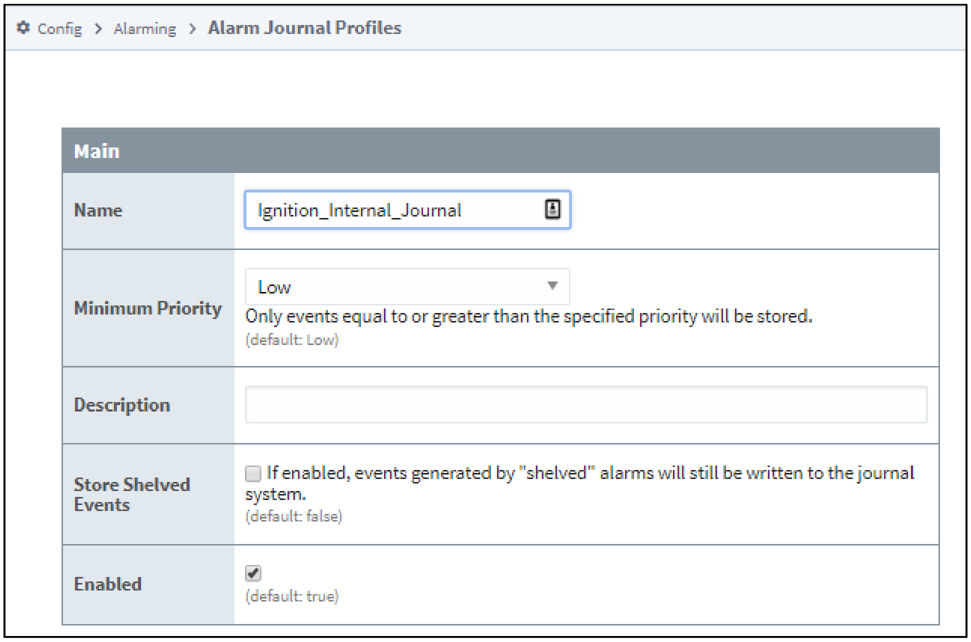


○ **Database**
Alarm journal events will be logged into an external database.

○ **Remote**
Alarm journal events will be sent to a remote gateway's alarm journal to be logged.

● **Internal**
Alarm journal events will be logged into a SQLite database file, local to the gateway.

3. Enter the name of your alarm journal profile and update any settings as required, then click **Next**.



Main	
Name	Ignition_Internal_Journal
Minimum Priority	Low Only events equal to or greater than the specified priority will be stored. (default: Low)
Description	
Store Shelved Events	☐ If enabled, events generated by "shelved" alarms will still be written to the journal system. (default: false)
Enabled	☑ (default: true)

4. You will receive a successful message stating your new Alarm Journal Profile was created.

Internal Alarm Journal Properties Table

Main	
Name	The default name, is the name of the Remote Gateway and Alarm Journal.
Minimum Priority	Only events equal to or greater than the specified priority will be stored.
Description	Description of the journal profile. Optional
Stored Shelved Events	If enabled, events generated by "shelved" alarms will still be written to the journal system. Default is false.
Enabled	By default. The journal profile is enabled.

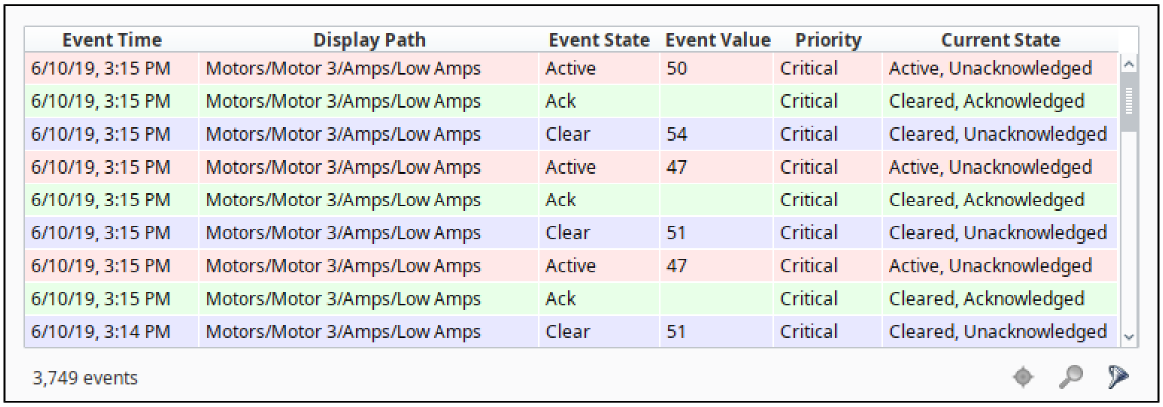
Third Party Accessibility

Because the Alarm Journal uses a SQL database to log alarm events, any application that has access to the database can retrieve journal data. Alarm events can be made freely available outside of Ignition, and integrated into other software packages. Additionally, other applications can write to the same tables, so Ignition applications can monitor activity in other systems.

Alarm Journal Component

While a SQL query will return data from the journal, the Alarm Journal Table component will automatically do so without manually writing a query. The component can filter on both Display Path and Source Path, as well as Date Range. The component can be configured to a single Journal Profile, so multiple instances of the component in the same project may look at different profiles.

More information on this component can be found on the Alarm Journal Table Component page.



Event Time	Display Path	Event State	Event Value	Priority	Current State
6/10/19, 3:15 PM	Motors/Motor 3/Amps/Low Amps	Active	50	Critical	Active, Unacknowledged
6/10/19, 3:15 PM	Motors/Motor 3/Amps/Low Amps	Ack		Critical	Cleared, Acknowledged
6/10/19, 3:15 PM	Motors/Motor 3/Amps/Low Amps	Clear	54	Critical	Cleared, Unacknowledged
6/10/19, 3:15 PM	Motors/Motor 3/Amps/Low Amps	Active	47	Critical	Active, Unacknowledged
6/10/19, 3:15 PM	Motors/Motor 3/Amps/Low Amps	Ack		Critical	Cleared, Acknowledged
6/10/19, 3:15 PM	Motors/Motor 3/Amps/Low Amps	Clear	51	Critical	Cleared, Unacknowledged
6/10/19, 3:15 PM	Motors/Motor 3/Amps/Low Amps	Active	47	Critical	Active, Unacknowledged
6/10/19, 3:15 PM	Motors/Motor 3/Amps/Low Amps	Ack		Critical	Cleared, Acknowledged
6/10/19, 3:14 PM	Motors/Motor 3/Amps/Low Amps	Clear	51	Critical	Cleared, Unacknowledged

3,749 events

Journal Properties

Main	
Name	The default name is Journal.
Description	Description of the journal profile.
Enabled	By default. the journal profile is enabled.
Datasource	This is the ONLY required setting which must be a valid database connection. Events are stored to this datasource.
Minimum Priority	Only events equal to or greater than the specified priority will be stored. The default is Low. You can set the priority to be: Diagnostic, Low, Medium, High, and Critical.
Store Shelved Events	Not enabled by default. If enabled, events generated by "shelved" alarms will still be written to the journal system.
Use Store and Forward	Enabled by default, which means the alarm journaled events will be stored through the Store and Forward system. If not enabled, they will be stored directly against the database. This system protects data from being lost due to temporary database connectivity issues.

Stored Event Data

Alarm events consist of two main types of data: the primary information about the alarm, such as transition state, time, etc. , and the event data.
The following settings specify what type of event data is stored:

Static Config	By default, it is not selected. If selected, will store the values of static alarm configuration. That is, the alarm properties that are *not* bound. These do not change during evaluation, only when a user modifies them in the Designer, and so they are not stored by default.
Dynamic Config	If selected, will store the values of dynamically bound alarm configuration properties. The value of these properties can change at any time, and the values at the time of the alarm are captured on the alarm event.
Static Associated Data	If selected, will store the values of non-bound associated data (properties created by the user) properties on alarm that do not change during execution.
Dynamic Associated Data	If selected, will store the values of dynamically bound associated data (properties created by the user) properties.

Data Filters

> The following three properties interact via logical AND, meaning an alarm must meet the criteria for all three for it to be logged in the Journal. Thus, if you supply values for all three Data Filer properties, then an alarm must match the **Filter by Alarm Source**, **Filter by Display Path**, and **Filter by Display Path or Source** properties.
>
> Example: If a journal has values for all three properties, and an alarm only meets the requirements for **Filter by Alarm Source** and **Filter by Display Path or Source**, but not **Filter by Display Path**, then the alarm will not be logged to the Journal.
>
> If you want to filter on both the Display Path and Source Path, you would want to use only one of the two following approaches:
>
> - **Filter by Alarm Source** and **Filter by Display Path**
> - Only use **Filter by Display Path or Source**
>
> It is recommended to avoid using all three properties simultaneously on the same Journal.

Filter by Alarm Source	Only events matching the source will be stored. Multiple sources to match can be comma separated. Leave blank to store events from all sources.
Filter by Display Path	Only events matching the display path will be stored. Multiple display paths to match can be comma separated. Leave blank to store events from all display paths.
Filter by Display Path or Source	Only events matching the display path, if defined, will be stored. Multiple matches can be comma separated. If no display path is defined, only events matching the source will be stored. Leave blank to store all events.

Data Pruning

Enable Data Pruning	If selected, data will be deleted after the specified time period as set by the Prune Age and Units below. Default is false. Note that since the data is stored directly in a database, an administrator is free to manually delete data at any time.
Prune Age	The number of Prune Age Units to store data for. i.e., 1 year, 5 hours, etc. The default is 1.
Prune Age Units	The type of Prune Age Unit. Default is Years. You can choose the unit to be Milliseconds, Seconds, Minutes, Hours, Days, Weeks, Months, or Years.

Advanced

These settings let you specify your own table names. This is especially useful when trying to use multiple alarm profiles within a single database (not common, but can happen, especially with multiple systems sharing a single database).

Table Name	

	The table name for the core event table. The default is `alarm_events`.
Event Data Table Name	The table name for event data associated with alarms. The default is `alarm_event_data`.

Table Definitions

The Alarm Journal system will automatically create the necessary tables for you, and scripting functions can be used to query the system without having to know about the table structure. However, understanding the structure of the Alarm Journal tables can be useful for accessing the data in situations where SQL queries are more convenient.

Alarm Events (`alarm_events`)

This table stores the core data for each event that occurs. An event is a transition for an alarm between active, cleared, or acknowledged. Additionally, other events may be stored in this table that aren't directly related to an alarm, such as a system shutdown event. This table defines a primary key "id", that is used as a foreign key by the Alarm Event Data table, which stores additional information for each event.

Column Name	Data Type	Description
id	integer	A unique integer id for each event entry event
eventid	string	The UUID of the alarm event.
source	string	The qualified path of the entity that generated the alarm event. See below for more information about qualified paths.
display path	string	The value set for the "Display Path" of the alarm. Generally a user defined, friendlier version of the source.
priority	integer	The priority or severity of the alarm: 0: Diagnostic 1: Low 2: Medium 3: High 4: Critical
eventtype	integer	The type of transition represented by this event: 0: Active 1: Clear 2: Acknowledgement
eventflags	integer	A numeric bitmask flag field providing additional information about the event. Bit 0: System Event - One of the designated system events. (System Startup, System Shutdown) Bit 1: Shelved Event - The alarm was "shelved" at the time that the event occurred. Shelving alarms does not prevent execution, so if the journal is configured to store shelved events, they will be stored even if they're not sent to the notification system, or shown to users. Bit 2: System Acknowledgement - When the "live event limit" (defined in general alarm settings) is exceeded, the system will automatically acknowledge overflow events, and the acknowledgment event will have this flag set. Bit 3: Acknowledge Event - The event was acknowledged at the time of the event. For events that are cleared after being acknowledged. Bit 4: Cleared Event - The event was cleared at the time of the event. For alarms that are acknowledged after being cleared.
eventtime	datetime	The time of the event.

Alarm Event Data (alarm_event_data)

This table stores the properties associated with an alarm event. The individual event is referenced through the `ID` column, against the alarm event table.

Column Name	Data Type	Description

id	integer	The id that corresponds to the alarm event in the alarm_events table.
propname	string	The name of the property. May be one of the common alarm properties (a configuration setting), or the name of an associated data property.
dtype	integer	The data type of the property, indicating which data column should be used: 0: Integer 1: Float 2: String
intvalue, floatvalue, strvalue	integer, float (double), string	The corresponding value columns for the property. Unused columns will receive "null" values.

Qualified Paths

A qualified path in Ignition is a path to an object, described by various annotated components. Each component has a type identifier and a value, separated by a colon (`:`), and each component is separated by colon-forward slash (`:/`). For example, an alarm is identified by `alm:Alarm Name`. It usually exists under a Tag, in which case, its fuller path would be `tag:Path/To/Tag:/alm:Alarm Name`. Paths can be built up further depending on the level of specificity required by the situation.

Configuring Alarms

On this page

...

Alarms are conditions that are evaluated when the value of the Tag changes. When the condition becomes true, the alarm is said to be **active**. When it becomes false, the alarm is said to be **clear**. Alarms may also be **acknowledged**. This flags the alarm in Ignition so the acknowledgement state of each alarm can be made visible throughout Ignition. Acknowledgement also allows users to 'claim' alarms while letting colleagues know that the alarm is being handled. Until an alarm has been acknowledged, it is considered to be **unacknowledged** (unacked).

Alarms can be configured on Tags or OPC items in SQL Bridge (Transaction Groups). The different Tag types that you can configure alarms on include Memory Tags, Query Tags, Expression Tags, as well as Tags inside of a UDT. You can also put alarms on System Tags that Ignition inherently provides such as the Gateway Performance and CPU Usage.

Alarm Names and Forward Slashes

Alarm names can make use of forward slashes ("/"). However the alarm name property throughout Ignition is designed to only show characters after the rightmost forward slash. Assume you name an alarm with the following:

```
one/two/three
```

The alarm name property (as seen on the various alarm table components, Tag Browser, and various alarm scripting functions) will omit "one" and "two", leaving the alarm name as:

```
three
```

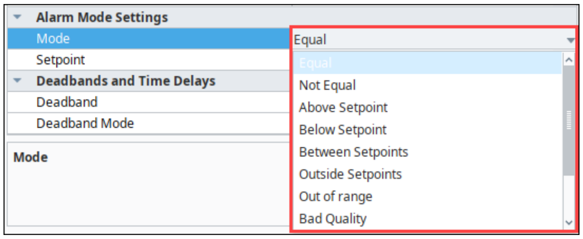
This behavior can be confusing, but is easily avoided by not using forward slashes in alarm names.

Alarm Properties

Similar in concept to properties on Vision components, alarm settings, also known as alarm properties, allow you to modify the behavior of each alarm. Each alarm will have its own alarm properties such as a unique Name, Priority, Display Path, Notes, and many other properties, some of which are optional. Descriptions of each alarm property can be found on the Tag Properties page.

Alarm Modes and Setpoints

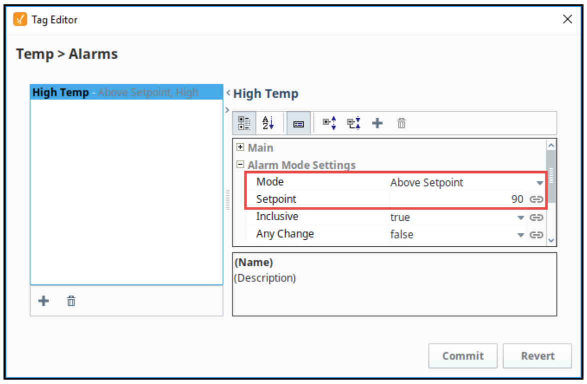
There are many different conditions that can be set up on an alarm. Alarm Mode Settings is where you define the actual conditions when the alarm goes 'true.' Some of the various Alarm Modes are shown in the image below.



Each alarm is configured with one mode, and usually one or more Setpoints. **Modes** determine the method in which alarm activity is evaluated, while Setpoints are the thresholds or limits that determine when a Tag is within the alarm state. Both properties work in together to determine when alarms become active, as well as when they are cleared.

For example, when Mode is set to "Above Setpoint", the Setpoint attribute is compared to the Tag's value. When the Tag value is above the the Setpoint value, the alarm becomes active. Once the value of the Tag is less than the Setpoint value, the alarm will transition to a cleared state.

For a complete list of Alarm Modes and their descriptions, refer to the Reference Table on the Tag Alarm Properties page.


Tag Editor

Temp > Alarms

High Temp - Above Setpoint, High

High Temp

Main	
Alarm Mode Settings	
Mode	Above Setpoint
Setpoint	90
Inclusive	true
Any Change	false

(Name)
(Description)

Alarming on Individual Bits

Additionally, alarms can become active by bit state. A common practice for PLC programming is to store alarm conditions as series of bits inside the PLC and expose them to the world as an integer value. Each bit is essentially a binary value. A series of 8 bits, for example, could be represented as an 8-bit integer. This integer value can be monitored by Ignition's OPC-UA server as an integer value as a Tag. This Tag can then have as many alarms as the integer has bits by monitoring the integer's bit state. Using the Bit Position Mode, multiple alarms in Ignition can be configured to monitor the state of each bit.

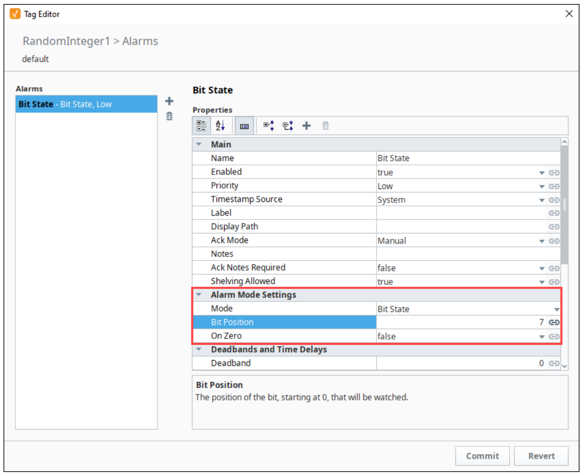
The following table shows binary and decimal equivalents:

Binary	Decimal
10001110	142

In this case, the Tag would be an integer with a value of 142. This Tag would have eight alarms. Each one of these alarms become active in accordance with its Boolean value.

Create Alarms on an Integer Tag Value

1. Edit a Tag that has an integer value.
2. Scroll down to Alarms, and next to **'No alarms,'** click on the **Edit** ✎ icon to create an alarm.
3. Click the **Add** + icon to add a new alarm to the Tag.
4. Fill in the appropriate properties, for example, enter the **Name, Priority**, and other properties as appropriate.
5. From the **Mode** section dropdown, select **Bit State** and enter a value for the **Bit Position**.
6. Click **Commit** to save the alarm. Repeat the same process for the remainder of the bits.


Tag Editor

RandomInteger1 > Alarms
default

Alarms
Bit State - Bit State, Low

Bit State

Properties

Main	
Name	Bit State
Enabled	true
Priority	Low
Timestamp Source	System
Label	
Display Path	
Ack Mode	Manual
Notes	
Ack Notes Required	false
Shelving Allowed	true
Alarm Mode Settings	
Mode	Bit State
Bit Position	7
On Zero	false
Deadbands and Time Delays	
Deadband	0

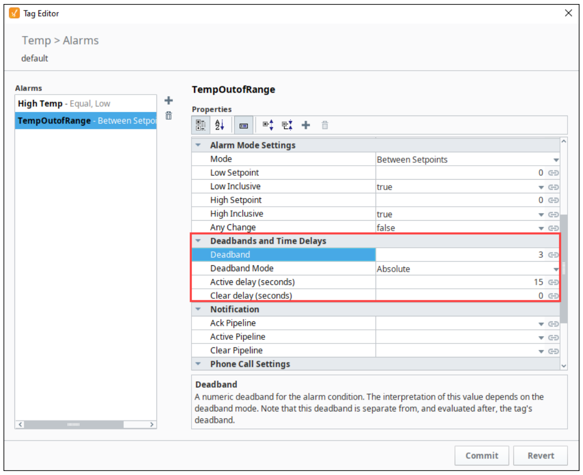
Bit Position
The position of the bit, starting at 0, that will be watched.

Deadband and Time Delays

The value for the deadband is interpreted by the Deadband Mode. It's important to note that all alarms are only evaluated after the Tag's value changes, which means that the Tag's own deadband will be considered first. When the deadband is positive, an active alarm condition needs to clear its setpoint(s) by the amount of the deadband for the alarm to clear.

For example, suppose we have a "Between Setpoints" alarm with a Low Setpoint of 45 and a High Setpoint of 65, and with a Deadband of 3. The alarm will go active if the value is between 45 and 65, but will only clear if the value falls below 42 or rises above 68.

In other situations, a Tag may frequently enter and leave an alarm state, but only for a brief moment. Normally an alarm would be generated each time the alarm condition was met, but the **Active Delay** attribute will prevent any alarms from being generated until the value on the Tag has stayed in an alarm state for a set period of time.



High Temp - Equal, Low
TempOutofRange - Between Setpoi

TempOutofRange

Alarm Mode Settings
Mode: Between Setpoints
Low Setpoint: 0
Low Inclusive: true
High Setpoint: 0
High Inclusive: true
Any Change: false

Deadbands and Time Delays
Deadband: 3
Deadband Mode: Absolute
Active delay (seconds): 15
Clear delay (seconds): 0

Notification
Ack Pipeline
Active Pipeline
Clear Pipeline

Phone Call Settings

Deadband
A numeric deadband for the alarm condition. The interpretation of this value depends on the deadband mode. Note that this deadband is separate from, and evaluated after, the tag's deadband.

To learn more about deadband and time delays, refer to the Reference Table on the Tag Alarm Properties page.

Associated Data

You can extend the list of alarm properties by adding your own Associated Data or custom properties to an alarm you already have configured. These values can be static or dynamic. Static properties are excellent for filtering alarms. Dynamic properties can be driven by another Tag, or evaluated with an expression. Regardless of the type, the Associated Data property values will be attached to the alarm event, viewed in real-time, and recorded in the Alarm Journal system.

Configuring Alarms

Alarms can be configured on any Tag type: Memory Tag, Query Tag, Expression Tag, as well as Tags inside of a UDT.

INDUCTIVE UNIVERSITY

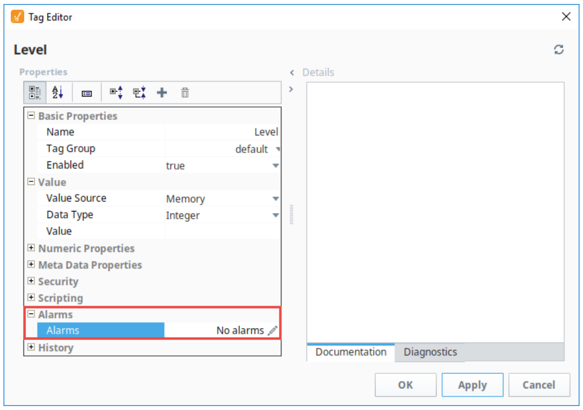
Configure an Alarm

Watch the Video

Configuring an Alarm on a Memory Tag

Let's use a simple Memory Tag with an Integer data type and create an alarm.

1. In the Tag Browser, double click a memory Tag (i.e. Level) to open the **Tag Editor**.
2. Scroll down to Alarms, and next to **'No alarms,'** click on the **Edit** ✎ icon to create an alarm, and the alarm UI will slide in from the right.



Tag Editor

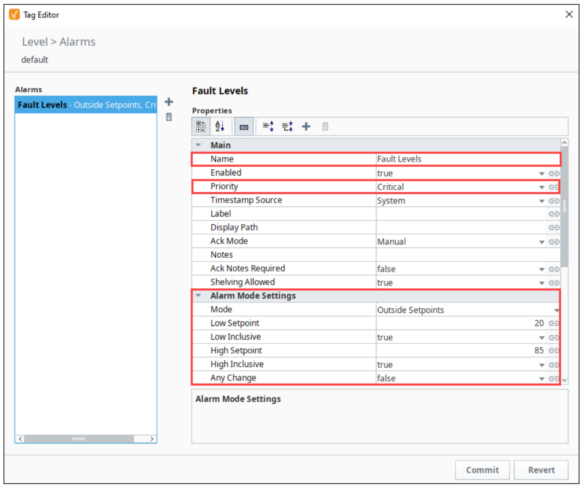
Level

Properties

Basic Properties	
Name	Level
Tag Group	default
Enabled	true
Value	
Value Source	Memory
Data Type	Integer
Value	
Numeric Properties	
Meta Data Properties	
Security	
Scripting	
Alarms	
Alarms	No alarms
History	

3. Click the **Add** ➕ icon in the bottom left corner of the window, or double click **New Alarm** item to add a new alarm to the Tag.
4. Configure the following alarm settings:
   Name: **Fault Levels**
   Priority: **Critical**
   Alarm Mode: **Outside Setpoints**
   Low Setpoint: **20** (when the Low Setpoint is below 20, an alarm is triggered)
   Low Inclusive: **true** (alarm is triggered when the low setpoint is equal to 20 or below)
   High Setpoint: **85** (when the High Setpoint is above 85, an alarm is triggered)
   High Inclusive: **true** (when the High Setpoint is equal to or above 85, an alarm is triggered)

5. Click **Commit** to save your alarm, or **Revert** to cancel.



Tag Editor

Level > Alarms

default

Alarms

Fault Levels - Outside Setpoints, Cri

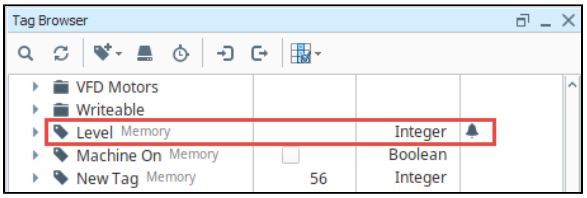
Fault Levels

Properties

Main	
Name	Fault Levels
Enabled	true
Priority	Critical
Timestamp Source	System
Label	
Display Path	
Ack Mode	Manual
Notes	
Ack Notes Required	false
Shelving Allowed	true
Alarm Mode Settings	
Mode	Outside Setpoints
Low Setpoint	20
Low Inclusive	true
High Setpoint	85
High Inclusive	true
Any Change	false

Alarm Mode Settings

6. Click **OK** to save your Tag edits.
7. Ignition will start monitoring the alarm immediately. Note the Alarm 🔔 icon will appear in the **Traits** column of the Tag Browser next to the name of the Tag. This means that at least one alarm is configured on this Tag.



Tag Browser

VFD Motors			
Writeable			
Level Memory		Integer	🔔
Machine On Memory	□	Boolean	
New Tag Memory	56	Integer	

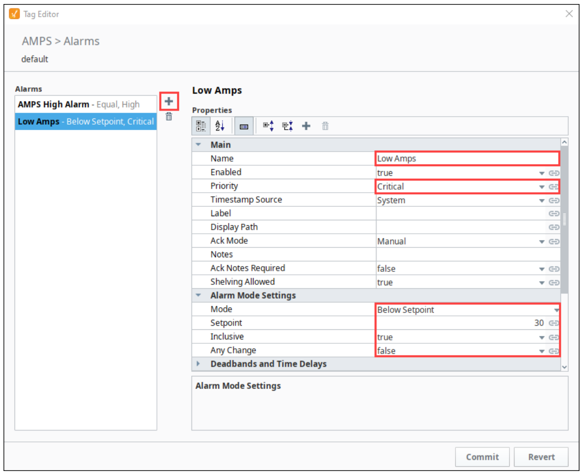
Configuring Alarms on Tags in a UDT

Alarms can be configured on Tags inside a UDT so when you create instances of that UDT, the alarms will automatically be configured.

In this example, we have a Motor UDT that contains two Tags: Amps and HOA. Let's configure an alarm on the Amps Tag.

1. In the **Tag Browser**, double click on a UDT (i.e., Motor) to open the **Tag Editor**.
2. In the Type Structure area, click on a Tag (i.e., Amps).
3. Scroll down to Alarms, and next to **No alarms,** click on the **Edit** ✎ icon to create an alarm.
4. Click the **Add** ✚ icon.
5. For this example, we entered the following:
   Name: **Low Amps**
   Priority: **High**
   Alarm Mode: **Below Setpoint**
   Setpoint: **30**
   Inclusive: **true** (an alarm is triggered when the low setpoint is equal to 30 or below)

6. Click **Commit** to save your alarm.



Tag Editor

AMPS > Alarms
default

Alarms
- AMPS High Alarm - Equal, High
- Low Amps - Below Setpoint, Critical

Low Amps

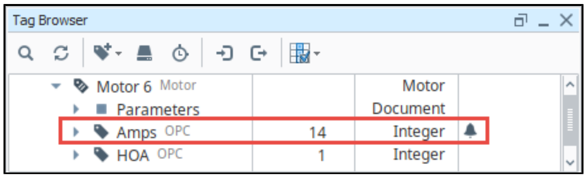
Properties

Main	
Name	Low Amps
Enabled	true
Priority	Critical
Timestamp Source	System
Label	
Display Path	
Ack Mode	Manual
Notes	
Ack Notes Required	false
Shelving Allowed	true
Alarm Mode Settings	
Mode	Below Setpoint
Setpoint	30
Inclusive	true
Any Change	false
Deadbands and Time Delays	

Alarm Mode Settings

7. Click **OK** to save the UDT. Now you're ready to create instances of the UDT. Once you create your UDT instances, Ignition will start monitoring the alarm immediately.

> ⚠ The Alarm 🔔 icon will appear in the **Traits** column of the Tag Browser next to the name of the Tag in the UDT instance. (Notice how you see the name of the Motor UDT next to the Amps Tag of the Motor 6 instance.)



Tag Browser

Motor 6 Motor		Motor	
Parameters		Document	
Amps OPC	14	Integer	🔔
HOA OPC	1	Integer	

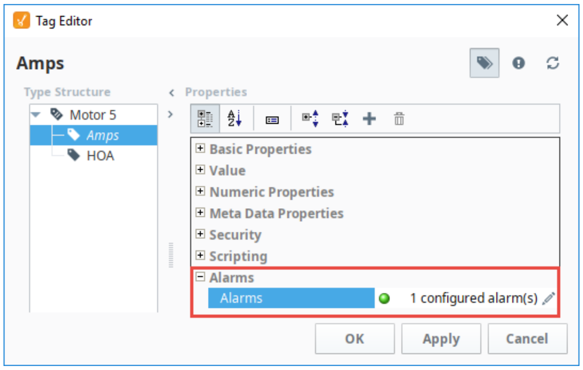
Updates to the UDT and Overriding Alarm Settings

If the alarm in the UDT is updated, the instance will automatically receive the updates and be refreshed. It's not uncommon that you may need to make alarm attribute values unique for particular instances, in which case, you can override alarm settings. Click the override button on the alarm UI of the Tag Editor making it green and edit the alarm properties and/or their values To learn more, refer to Overriding Properties in UDT Instances.

INDUCTIVE UNIVERSITY

Configure Alarm in UDT

Watch the Video



Configuring Multiple Alarms on a Single Tag

Tags can have multiple alarms configured, each with unique setpoints and other configurations. The number of alarms that you can configure is virtually unlimited and will never exceed the bounds of reasonable design consideration. Adding additional alarms is just as easy as adding the first alarm. Simply click on the **Add** + icon in the alarm UI and start configuring the new alarm. This way separate alarms can monitor different setpoints, so Tags can have alarms for high setpoints and low setpoints.

> **Tip**
>
> Alarms can be copied and pasted with keyboard shortcuts. Select an alarm from the Tag Editor, press **Ctrl-C** to copy, and **Ctrl-V** to paste. Alarms with similar configurations can rapidly be replicated to hasten development.

INDUCTIVE UNIVERSITY

Multiple Alarms on a Tag

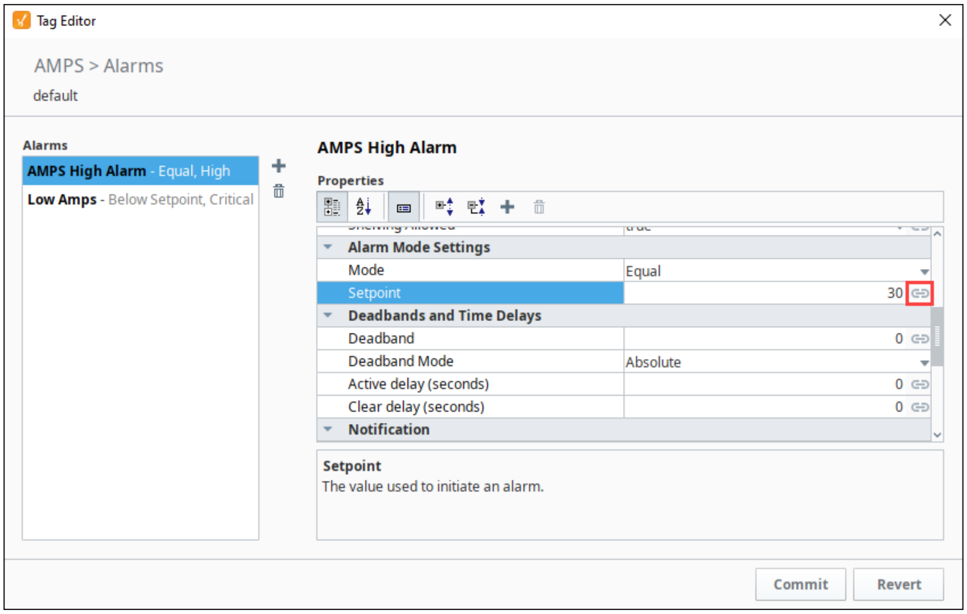
Watch the Video

Alarm Property Bindings

Many alarm properties are bindable, which means they can be bound to other Tags in the system, expressions and even a UDT parameter. For example, you might bind the Enabled property to another Tag which represents whether or not your process is running, thereby disabling the alarm when production is stopped. Or, you might bind the setpoint of an alarm to a Tag that operators can manipulate, thereby letting the setpoint be changed at runtime.

This example shows how to bind an alarm property from the Tag Browser in Perspective.

1. Double click on your Tag to open the **Tag Editor**.
2. Click on the **Edit** ✎ icon next to **Configured Alarms** to see your configured alarms. (If you don't have any alarms, create an alarm using the steps in the Configuring an Alarm on a Tag.)
3. Select the alarm and the screen will refresh with all the alarm properties.
4. From here, find the alarm property you want to add a binding to and click on the binding 🔗 icon. The binding UI will slide in from the right.



AMPS > Alarms
default

Alarms
- **AMPS High Alarm** - Equal, High
- **Low Amps** - Below Setpoint, Critical

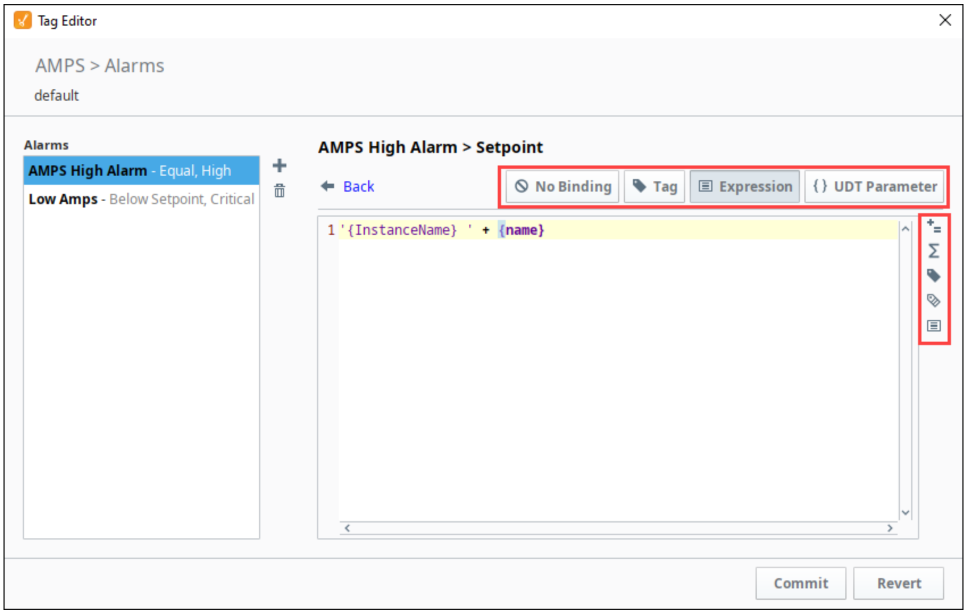
AMPS High Alarm

Properties

Alarm Mode Settings	
Mode	Equal
Setpoint	30
Deadbands and Time Delays	
Deadband	0
Deadband Mode	Absolute
Active delay (seconds)	0
Clear delay (seconds)	0
Notification	

Setpoint
The value used to initiate an alarm.

5. Select the binding type (**No Binding**, **Tag**, **Expression**, or **UDT Parameter**, if applicable). The image below shows an example of an Expression binding. Note that the expression can reference many useful values such as the Tag's value and other settings of the alarm. Enter your expression.
6. Once you configured the binding to your liking, click **Commit**, or **Revert** if you decide to cancel.



AMPS > Alarms
default

Alarms
- **AMPS High Alarm** - Equal, High
- **Low Amps** - Below Setpoint, Critical

AMPS High Alarm > Setpoint

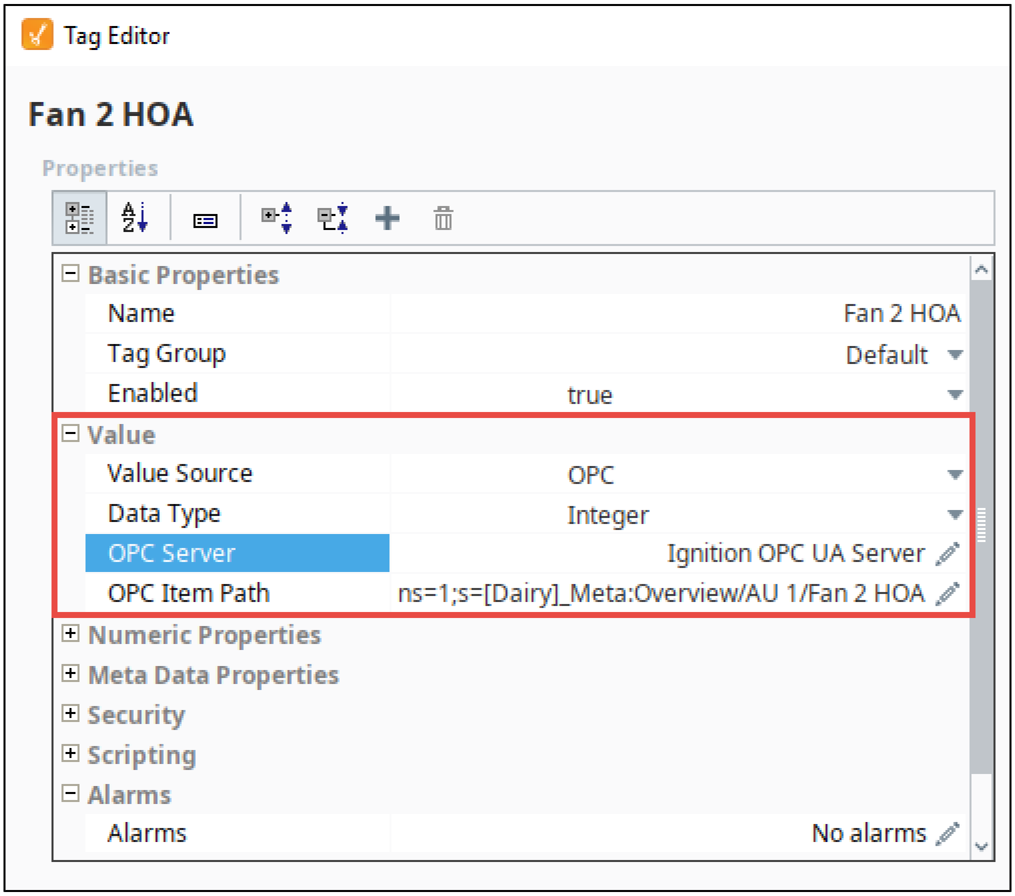
```
'{InstanceName} ' + {name}
```

7. Click **OK** to save the changes to the Tag.

For more information on property bindings see, Property Bindings in Perspective and Property Bindings in Vision.

Alarms in Transaction Groups

Alarms can also be added to OPC items in Transaction Groups. This means alarms can be used without ever creating a Tag in Ignition. Simply edit an OPC item, and an Alarming section will appear in the Tag Editor window. From here, adding an alarm to the item is similar to adding an alarm to an Ignition Tag.



Tag Editor

Fan 2 HOA

Properties

Basic Properties

Name	Fan 2 HOA
Tag Group	Default
Enabled	true

Value

Value Source	OPC
Data Type	Integer
OPC Server	Ignition OPC UA Server
OPC Item Path	ns=1;s=[Dairy]_Meta:Overview/AU 1/Fan 2 HOA

Numeric Properties

Meta Data Properties

Security

Scripting

Alarms

Alarms	No alarms

Dynamic Alarm Attributes

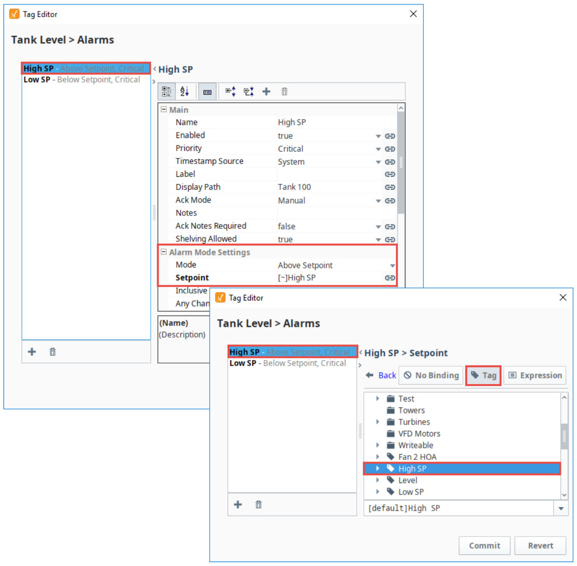
Dynamic Setpoints

You can configure alarms with dynamic setpoints inside of a Tag definition. It is similar to configuring an alarm, however, configuring an alarm with a dynamic setpoint requires additional Tags to serve as these setpoints. A good example of this is when an operator changes a high or low setpoint, it also changes how the alarm is evaluated.

To Configure Dynamic Setpoints on an Alarm

Suppose you want to alarm a Tank Level if it goes above or below a certain setpoint. Rather than hardcoding a value in the high and low Setpoint properties, you can bind them to either a Tag in the system or an Expression to make them dynamic. This example uses an OPC Tag called Tank Level, and two Memory Tags called High SP and Low SP. The Tank Level will use the values in the Memory Tags as the high and low setpoints. Let's alarm the tank level when the setpoint goes above 85 and below 20.

1. Create two Memory Tags calling one **High SP** with value of 85, and another one called **Low SP** with a value of 20.
2. Use an OPC Tag and rename it to **Tank Level**. Click on the Tag to open the **Tag Editor**.
3. Scroll down to Alarms, and click on the **Edit** ✎ icon next to your alarm to open the Alarm UI.
4. If you don't have a configured alarm, you will need to create one. Select the configured alarm that you want associate a high setpoint with. (This examples uses the **High SP** alarm).
5. Under Alarm Mode Settings, set the Mode to **Above Setpoint**.
6. Click on the **binding** 🔗 icon for the **Setpoint** property. You can select a **Tag** or **Expression**. (This examples binds the **Setpoint** property to a Memory Tag that was created called **High SP**). Click the **Commit** to commit your changes.



Name: High SP
Enabled: true
Priority: Critical
Timestamp Source: System
Label:
Display Path: Tank 100
Ack Mode: Manual
Notes:
Ack Notes Required: false
Shelving Allowed: true
Mode: Above Setpoint
Setpoint: [~]High SP

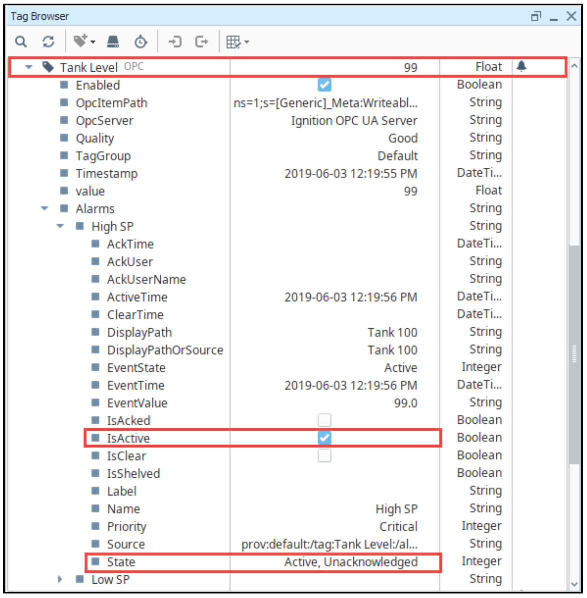
[default]High SP

7. Now let's configure the low setpoint. Create another alarm to associate with the low setpoint if you don't have one.
8. Under Alarm Mode Settings, set the Mode to **Below Setpoint**.
9. Click on the **binding** 🔗 icon for the **Setpoint** property. You can select a **Tag** or **Expression**. (This examples binds the **Setpoint** property to a Memory Tag that was created called **Low SP**). Click the **Commit** to commit your changes.
10. Click **OK**. Now, the tank alarm will be evaluated based upon the high and low setpoints of each Tag.

On this page

...

11. To test it, change the value of the Tank Level to 99, and you'll notice it actives the alarm.



Tank Level OPC	99	Float	🔔
Enabled	☑	Boolean	
OpcItemPath	ns=1;s=[Generic]_Meta:Writeabl...	String	
OpcServer	Ignition OPC UA Server	String	
Quality	Good	String	
TagGroup	Default	String	
Timestamp	2019-06-03 12:19:55 PM	DateTi...	
value	99	Float	
Alarms		String	
High SP		String	
AckTime		DateTi...	
AckUser		String	
AckUserName		String	
ActiveTime	2019-06-03 12:19:56 PM	DateTi...	
ClearTime		DateTi...	
DisplayPath	Tank 100	String	
DisplayPathOrSource	Tank 100	String	
EventState	Active	Integer	
EventTime	2019-06-03 12:19:56 PM	DateTi...	
EventValue	99.0	String	
IsAcked	□	Boolean	
IsActive	☑	Boolean	
IsClear	□	Boolean	
IsShelved		Boolean	
Label		String	
Name	High SP	String	
Priority	Critical	Integer	
Source	prov:default:/tag:Tank Level:/al...	String	
State	Active, Unacknowledged	Integer	
Low SP		String	

While you can configure dynamic values on any property that is showing the binding icon on the right, some other properties (like Name) may also accept dynamic properties using the {myParam} notation. We recommend against using Dynamic setpoints on these static alarm properties because they will only evaluate on startup, not while the Gateway is running.

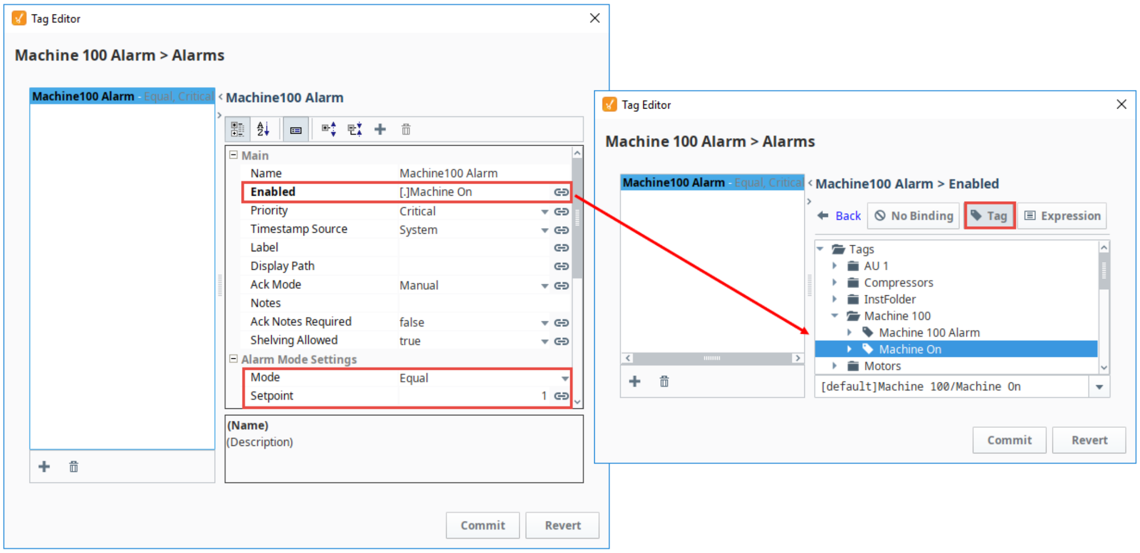
Dynamic Enabling and Disabling

Enabling Based on Machine State

Allowing a dynamic condition to determine if an alarm is enabled or disabled is possible inside the alarm's properties, like whether or not a machine is on. This example uses two Tags: one for the Alarm, and one to determine the running state of the machine. Two conditions must be true for the alarm to become active. The value on the alarm Tag must match the setpoint ('true' in the example below), and the **Enabled** property must evaluate to 'true' driven by the value of the 'Machine On' Tag.

1. Select a **Tag** that has the alarm you want to configure.
2. To access the alarm properties, click on the **Edit** icon next to your alarm to open the Alarm UI. Select an alarm or create a alarm if one doesn't exist. The alarm in this example is called Machine 100 Alarm.

3. Click the **binding** icon for the **Enabled** property and bind it to a **Boolean Tag** or an **Expression** that evaluates a 'true' or 'false' condition. This example uses a Tag called Machine On. Enter the **Mode** to **'Equal'** and the **Setpoint** to **'1.'** Click **Commit**.



Tag Editor

Machine 100 Alarm > Alarms

Machine100 Alarm Equal, Critical

Machine100 Alarm

Main

Name	Machine100 Alarm
Enabled	[.]Machine On
Priority	Critical
Timestamp Source	System
Label	
Display Path	
Ack Mode	Manual
Notes	
Ack Notes Required	false
Shelving Allowed	true

Alarm Mode Settings

Mode	Equal
Setpoint	1

(Name)
(Description)

Tag Editor

Machine 100 Alarm > Alarms

Machine100 Alarm Equal, Critical

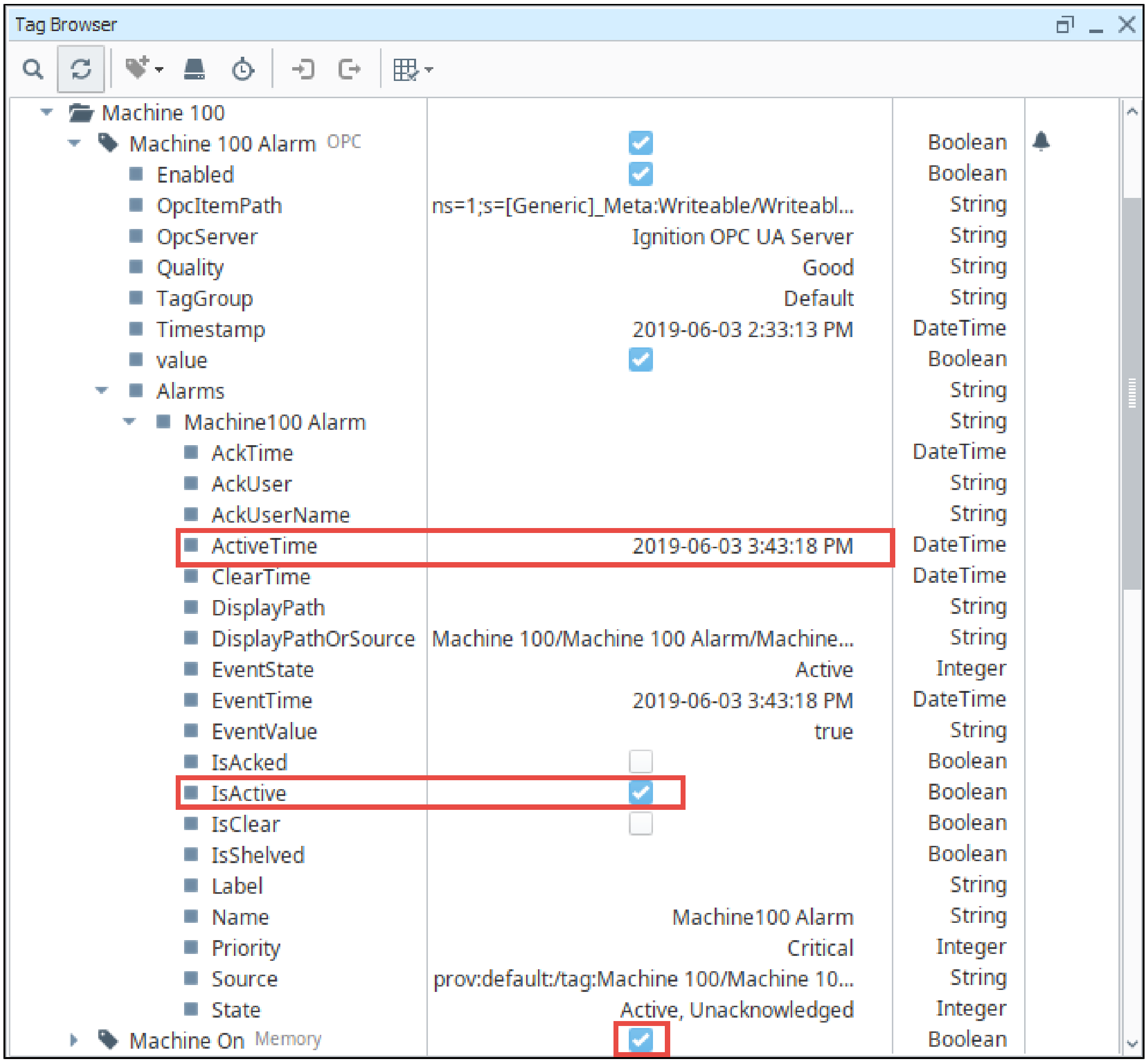
Machine100 Alarm > Enabled

- Tags
  - AU 1
  - Compressors
  - InstFolder
  - Machine 100
    - Machine 100 Alarm
    - Machine On
  - Motors

[default]Machine 100/Machine On

4. Click **Commit** to save your changes. Now, the alarm will only be evaluated based upon the state of the Machine On Tag. Set the Machine to **On** or **'true.'**

5. In the Tag Browser, expand your Tag (i.e., Machine 100 Alarm), and you'll notice the **'IsAlert'** is now active. As you can see, making the Enable property dynamic based on another Tag in the system is extremely easy.



Tag Browser

Machine 100		
Machine 100 Alarm OPC	☑	Boolean
Enabled	☑	Boolean
OpcItemPath	ns=1;s=[Generic]_Meta:Writeable/Writeabl...	String
OpcServer	Ignition OPC UA Server	String
Quality	Good	String
TagGroup	Default	String
Timestamp	2019-06-03 2:33:13 PM	DateTime
value	☑	Boolean
Alarms		String
Machine100 Alarm		String
AckTime		DateTime
AckUser		String
AckUserName		String
ActiveTime	2019-06-03 3:43:18 PM	DateTime
ClearTime		DateTime
DisplayPath		String
DisplayPathOrSource	Machine 100/Machine 100 Alarm/Machine...	String
EventState	Active	Integer
EventTime	2019-06-03 3:43:18 PM	DateTime
EventValue	true	String
IsAcked	☐	Boolean
IsActive	☑	Boolean
IsClear	☐	Boolean
IsShelved		Boolean
Label		String
Name	Machine100 Alarm	String
Priority	Critical	Integer
Source	prov:default:/tag:Machine 100/Machine 10...	String
State	Active, Unacknowledged	Integer
Machine On Memory	☑	Boolean

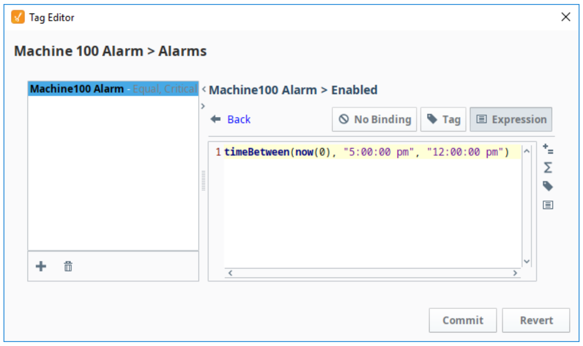
Enabling Based on Time of Day

There are occasions when you only want alarms to be evaluated at certain times of the day. In Ignition, you can automatically enable and disable alarms for specific times of the day. This is typically achieved by binding the alarm's Enabled property to a Tag or an Expression. (This example uses an Expression).

1. From the **Tag Browser**, select the **Tag** on which you want to configure the alarm.
2. To access the alarm properties, click on the **Edit** ✏ icon next to your alarm to open the Alarm UI.
3. Select an alarm you want to place the binding on, or create a alarm if one doesn't exist. The alarm in this example is called Machine 100 Alarm.
4. Bind the **Enabled** property to an expression that evaluates a 'true' or 'false' condition based upon the current time. Click the binding button to the right of the Enabled property, and click on **Expression** tab on the top right of the screen, and enter your expression or copy and paste from the code block below, then click **Commit**.

   In the expression language, there are various functions for dates that can be used. For example, the following expression would return 'true' if the time is between the hours of 5pm and 12pm, and return 'false' if it is not.

```
timeBetween(now(0), "5:00:00 pm", "12:00:00 pm")
```



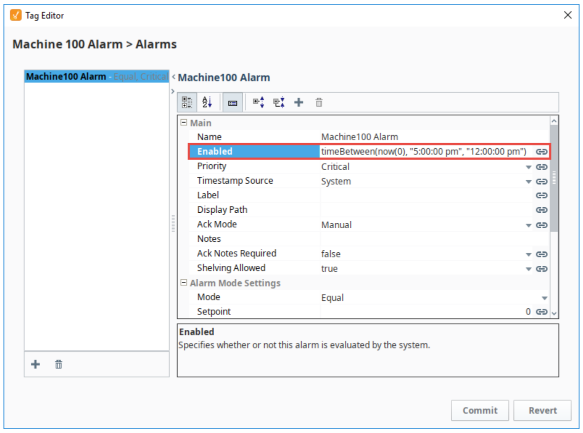
Machine 100 Alarm > Alarms

Machine100 Alarm - Equal, Critical

Machine100 Alarm > Enabled

```
timeBetween(now(0), "5:00:00 pm", "12:00:00 pm")
```

5. Click **Commit** again to save your edits. The alarm will only be evaluated when the system clock falls between the specified time.



Machine 100 Alarm > Alarms

Machine100 Alarm - Equal, Critical

Machine100 Alarm

Main	
Name	Machine100 Alarm
Enabled	timeBetween(now(0), "5:00:00 pm", "12:00:00 pm")
Priority	Critical
Timestamp Source	System
Label	
Display Path	
Ack Mode	Manual
Notes	
Ack Notes Required	false
Shelving Allowed	true
Alarm Mode Settings	
Mode	Equal
Setpoint	0

Enabled
Specifies whether or not this alarm is evaluated by the system.

Alarms in UDTs

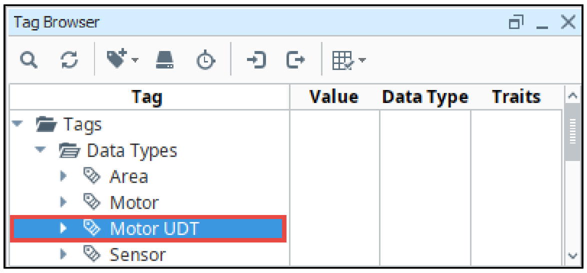
Configuring an Alarm on a UDT Member

The great thing about UDTs is that you configure it in one place, inside of the UDT definition, and every instance of that UDT will automatically inherit that same configuration. The same concept works for alarms on a UDT. If an alarm is configured inside a UDT, every instance of that UDT will automatically have that same alarm configuration. Even if a new instance is created, it will automatically get that same alarm configuration.

Configuring an Alarm on a UDT

This example uses a Motor UDT. The Motor UDT contains two OPC Tags: Amps and HOA. Let's configure an alarm on the Motor UDT when the Amps setpoint goes under 25.

1. In the **Tag Browser**, go to the **Data Types** folder, and double click on a UDT (i.e., Motor UDT) to edit the definition.



2. Select a Tag (i.e., Amps) and scroll down to Alarms, and click on the **Edit** ✎ icon next to your alarm to open the Alarm UI.
3. Click on the **Add** ✚ icon on the lower left side of the screen to create a new alarm. Enter an **Alarm Name, Display Path**, **Priority**, **Mode**, and **Setpoint**. UDT configurations are setup the same way as the normal alarm configuration on a Tag.

> **Display Path**
>
> Since you have multiple instances of a Motor, as in this example, you need to make sure that the Display Path is unique for every instance, otherwise, every instance of the UDT will have the same Display Path name, and the operator won't know which Motor alarm to respond too. There are a couple of ways to address this. You can leave the Display Path blank, in which case, Ignition will use the full Display Path to the instance of that Tag, or you can bind a property to an expression involving a UDT parameter (i.e., "Motor" + {MotorNumber}).

4. Click **Commit** to save your alarm edits.
5. Click **OK** to save your Motor UDT edits. Now, all instances of the UDT will have a similar alarm setup, but each alarm can be unique to that UDT instance by changing the **Below Setpoint** value.

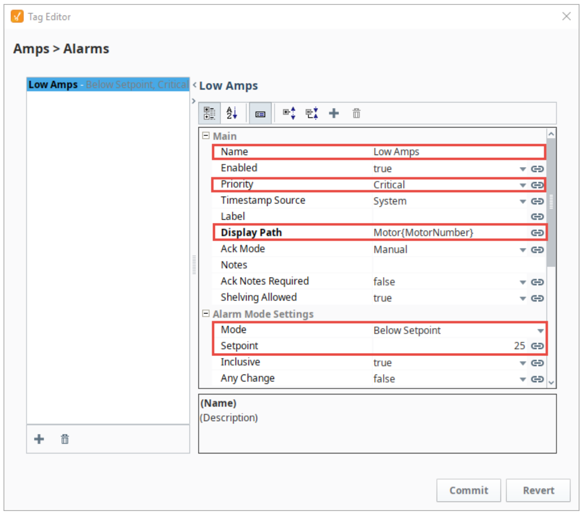
On this page

...

INDUCTIVE UNIVERSITY

Configure Alarm in UDT

Watch the Video



Tag Editor

Amps > Alarms

Low Amps - Below Setpoint, Critical

Low Amps

Main

- Name: Low Amps
- Enabled: true
- Priority: Critical
- Timestamp Source: System
- Label:
- Display Path: Motor{MotorNumber}
- Ack Mode: Manual
- Notes:
- Ack Notes Required: false
- Shelving Allowed: true

Alarm Mode Settings

- Mode: Below Setpoint
- Setpoint: 25
- Inclusive: true
- Any Change: false

(Name)
(Description)

Dynamic Setpoints in UDTs

Instead of using a hardcoded setpoint, you can configure alarms with dynamic setpoints inside of a UDT definition. It is similar to configuring a UDT alarm, however, configuring a UDT alarm with a dynamic setpoint requires additional Tags to serve as these setpoints in all the deployed UDTs.

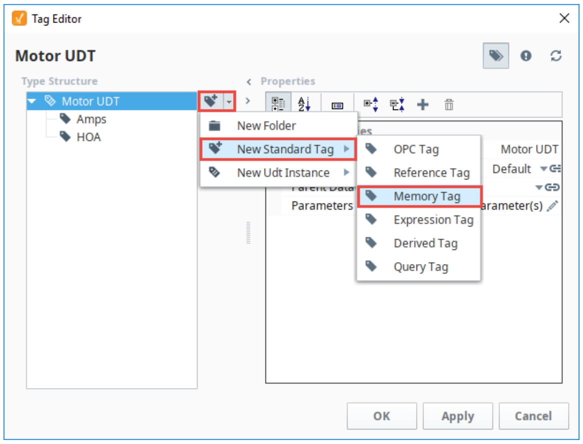
UDT Alarm Dynamic Setpoints

Watch the Video

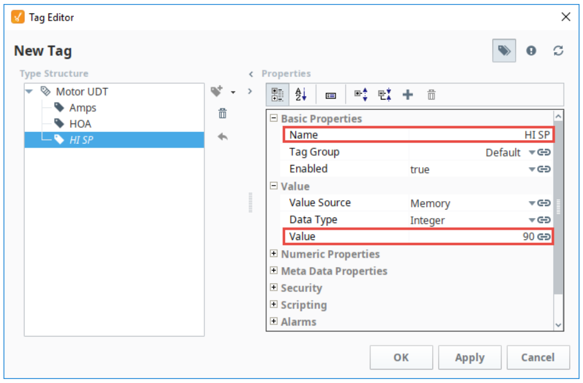
Configuring Dynamic Setpoints Inside a UDT

In the this example, we'll use the Motor UDT in the above section to create a **Memory Tag** inside the UDT definition to serve as the setpoint.

1. In the **Tag Browser**, go to the **Data Types** folder, and click on your UDT to edit the definition.
2. Click the **Add Tag** icon and select **New Standard Tag > Memory Tag** from the dropdown.



3. Enter the **Name** of the Tag (i.e., HI SP) and a **Value** (i.e., 90). Click **Apply.**



Basic Properties

Name	HI SP
Tag Group	Default
Enabled	true

Value

Value Source	Memory
Data Type	Integer
Value	90

Numeric Properties

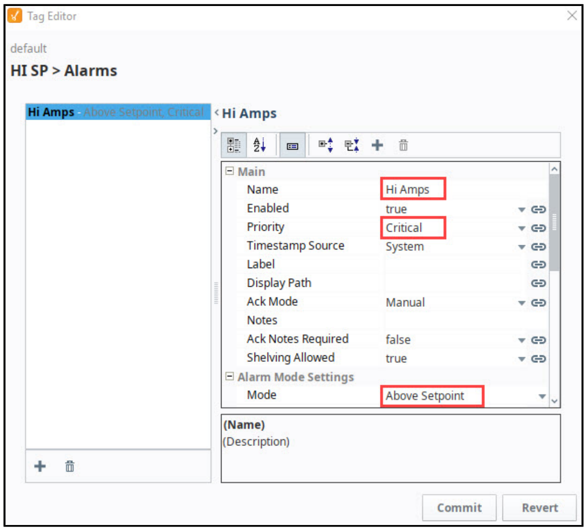
Meta Data Properties

Security

Scripting

Alarms

4. In the **Type Structure** area, select the Tag you want to associate with the dynamic setpoint (i.e., Amps)
5. Under Properties, scroll down to Alarms. Click on the **Edit** icon next to Alarms.
6. Click on the Add icon to create a new alarm for the high setpoint and call it "HI Amps".
7. Set the **Priority** as **Critical,** and the **Alarm Mode** as **Above Setpoint.** Click **Commit**.



Main

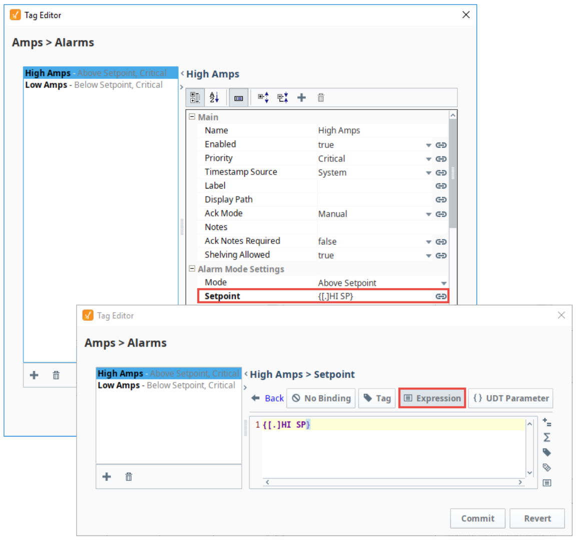
Name	Hi Amps
Enabled	true
Priority	Critical
Timestamp Source	System
Label	
Display Path	
Ack Mode	Manual
Notes	
Ack Notes Required	false
Shelving Allowed	true

Alarm Mode Settings

Mode	Above Setpoint

(Name)
(Description)

8. Bind the **Setpoint** property to the new UDT Memory Tag (i.e., HI SP). Click on the **binding** icon and you can either create an expression or use the UDT Tag Tab to select the Memory Tag (i.e., HI SP). This example uses an expression as shown in the image below.
9. Click **Commit** to save the expression.
10. Click **Commit** again to save your alarm edits.


Tag Editor
Amps > Alarms
High Amps - Above Setpoint, Critical
Low Amps - Below Setpoint, Critical
High Amps

Main	
Name	High Amps
Enabled	true
Priority	Critical
Timestamp Source	System
Label	
Display Path	
Ack Mode	Manual
Notes	
Ack Notes Required	false
Shelving Allowed	true
Alarm Mode Settings	
Mode	Above Setpoint
Setpoint	{[.]HI SP}

Tag Editor
Amps > Alarms
High Amps - Above Setpoint, Critical
Low Amps - Below Setpoint, Critical
High Amps > Setpoint
Back
No Binding
Tag
Expression
UDT Parameter
Commit
Revert

```
{[.]HI SP}
```

11. Click **OK** to save all your UDT updates. Now you're ready to create instances of your UDT. All of the UDT instances will now have Memory Tags that serve as alarm setpoints.
12. Test it out by dragging an Alarm Status Table component in a window and locating your alarm.

Related Topics ...

- Configuring Alarms
- UDT Definitions
- UDT Instances
- Dynamic Alarm Attributes
- UDT Multi-Instance Wizard

Alarm Associated Data

What is Alarm Associated Data

On this page ...

Every alarm in Ignition has alarm properties associated with it as it moves throughout the system, like active time, clear time, who acknowledged the alarm, time the alarm was acknowledged, priority, and display path. You can extend the list of alarm properties by adding your own associated data or custom properties to an alarm you already configured. Values can either be static or dynamic. Static properties are great for filtering alarms. Dynamic properties will often be bound to other Tags or an expression. Regardless of the whether a value is static or dynamic, these values will be attached to the alarm event as it moves through the system, and the values will be available from the Alarm Status system, the Alarm Journal system, and the Alarm Notification system.

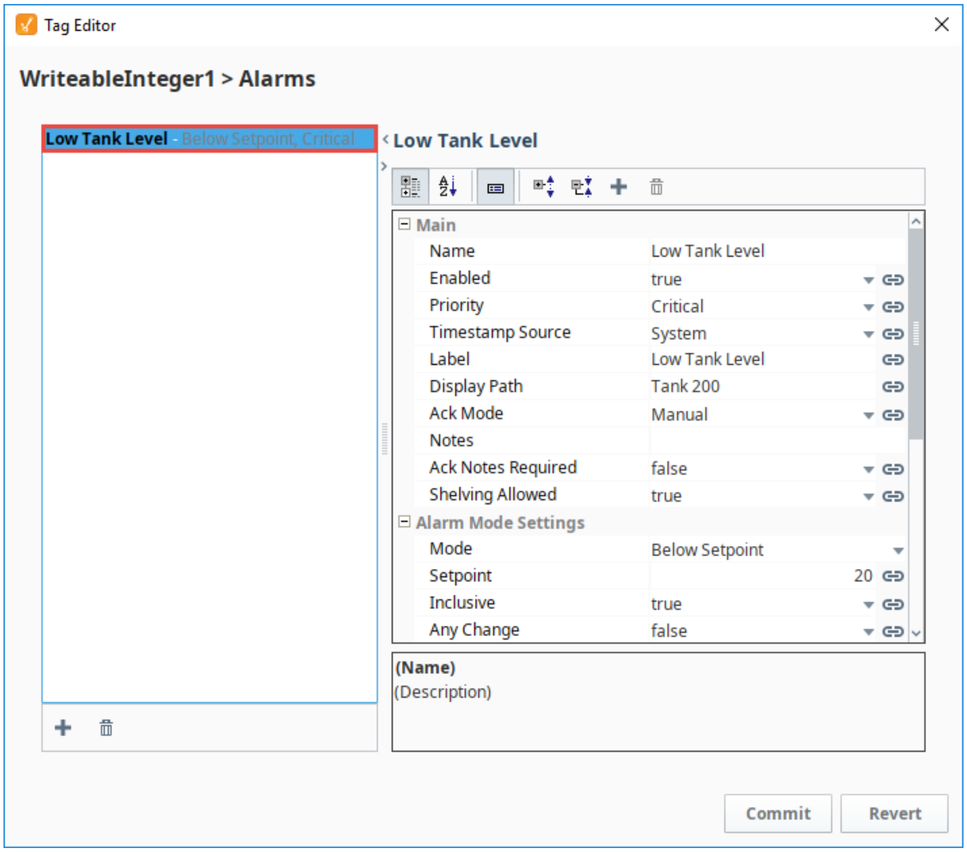
It's easy to go back to an existing alarm and add associated data.

Creating Associated Data for both Static and Dynamic Properties

For this example you can use an existing alarm you already configured or create a new alarm if you don't have one.

1. Right-click on your Tag and select **Edit Tag**. This example uses a WriteableInteger OPC Tag, but any alarm-capable Tag will suffice.

2. In the **Tag Editor**, scroll down to Alarms, and next to 'No alarms,' click on the **Edit** icon and click on your alarm. The Tag below already has an alarm named ' Low Tank Level.' Select the alarm to open the Alarm UI.



WriteableInteger1 > Alarms

Low Tank Level - Below Setpoint, Critical

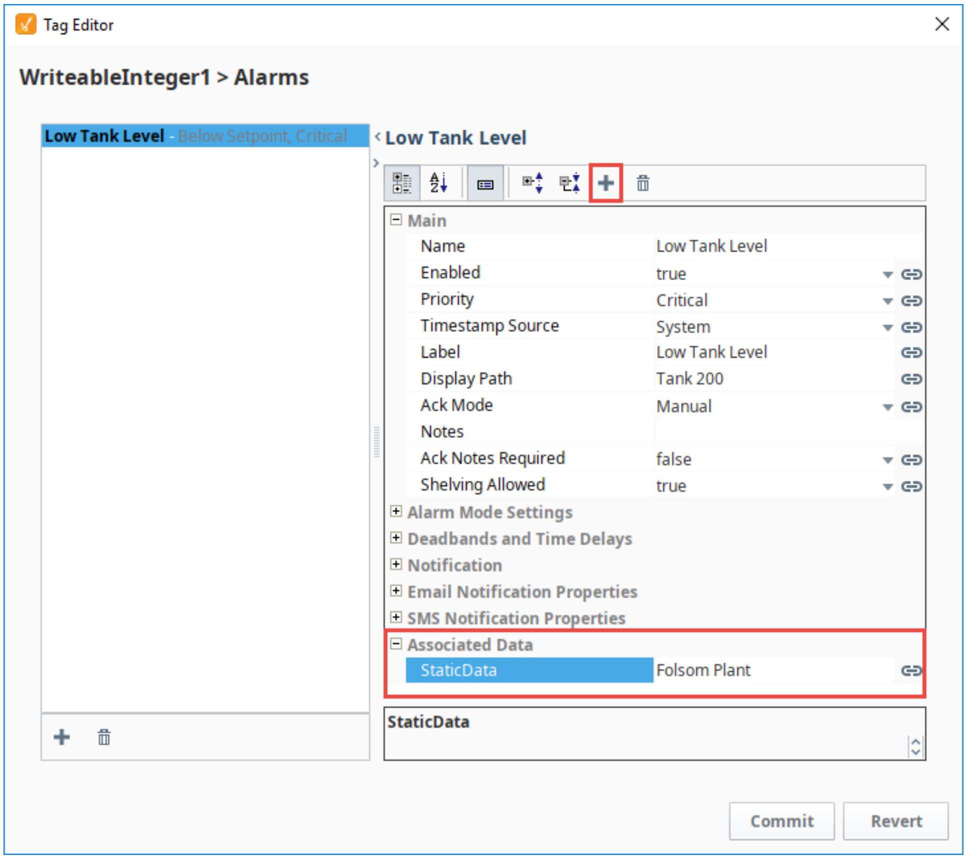
Low Tank Level

Main	
Name	Low Tank Level
Enabled	true
Priority	Critical
Timestamp Source	System
Label	Low Tank Level
Display Path	Tank 200
Ack Mode	Manual
Notes	
Ack Notes Required	false
Shelving Allowed	true
Alarm Mode Settings	
Mode	Below Setpoint
Setpoint	20
Inclusive	true
Any Change	false

(Name)
(Description)

3. Click the **Add** icon above the alarm properties to add new associated data for the alarm. Scroll to the bottom of the property list, and you'll see a new associated data property was added. By default, the new property name is called **'New Data.'** Double-click the property name to rename it to something more meaningful. In this example, make the associated data property a static value. We

renamed ours to **'StaticData'** with a value of **'Folsom Plant'** and press enter to submit.



Tag Editor

WriteableInteger1 > Alarms

Low Tank Level - Below Setpoint, Critical

Low Tank Level

Main

Name	Low Tank Level
Enabled	true
Priority	Critical
Timestamp Source	System
Label	Low Tank Level
Display Path	Tank 200
Ack Mode	Manual
Notes	
Ack Notes Required	false
Shelving Allowed	true

Alarm Mode Settings

Deadbands and Time Delays

Notification

Email Notification Properties

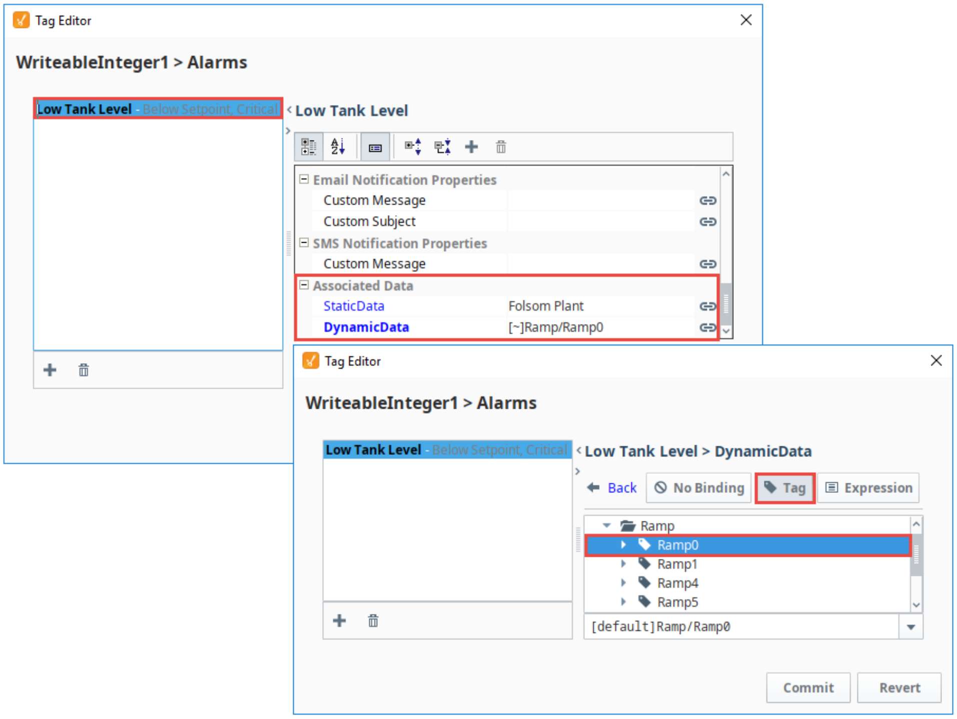
SMS Notification Properties

Associated Data

StaticData	Folsom Plant

StaticData

4. Now, let's create another Associated Data property with a dynamic value. Click the **Add** + icon above the alarm properties.
5. Scroll down to the bottom of the property list and rename your **'New Data'** property to **'DynamicData'** and bind it to a Tag using the **binding** 🔗 icon on the right side of the window. This example uses a Ramp0 OPC Tag.
6. Click the **'Commit'** to submit the Tag selection, and then click **OK** to save your Tag edits.



Tag Editor

WriteableInteger1 > Alarms

Low Tank Level - Below Setpoint, Critical

Low Tank Level

Email Notification Properties

Custom Message	
Custom Subject	

SMS Notification Properties

Custom Message	

Associated Data

StaticData	Folsom Plant
DynamicData	[~]Ramp/Ramp0

Tag Editor

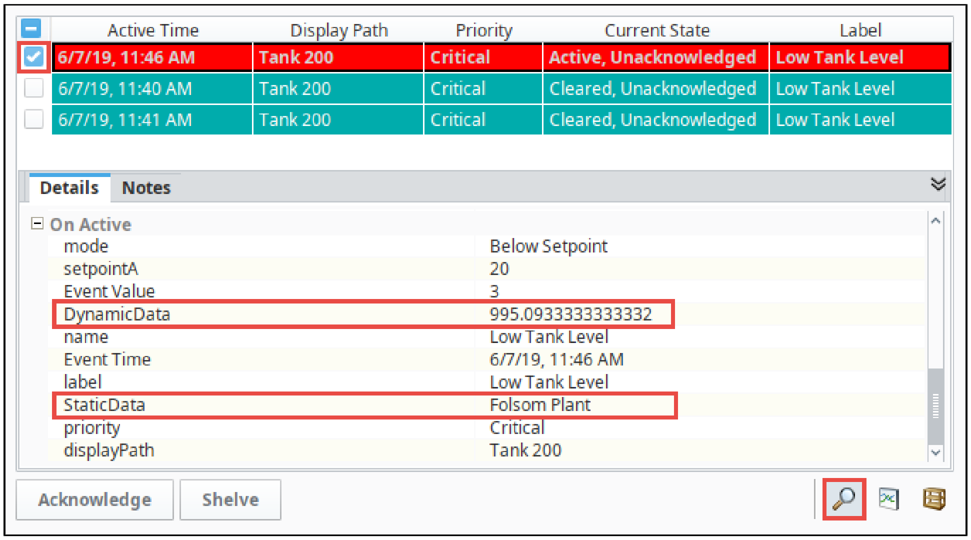
WriteableInteger1 > Alarms

Low Tank Level - Below Setpoint, Critical

Low Tank Level > DynamicData

[default]Ramp/Ramp0

7. When the alarm goes active, Ignition stores the values of the associated data with the alarm. You can view the details of the alarm in the Alarm Status Component along with the new associated data properties. To view the **Details** of the alarm, check the box of the alarm you want to see, and click on the magnifying glass at the bottom of the window to open the Details window.



☐	Active Time	Display Path	Priority	Current State	Label
☑	6/7/19, 11:46 AM	Tank 200	Critical	Active, Unacknowledged	Low Tank Level
☐	6/7/19, 11:40 AM	Tank 200	Critical	Cleared, Unacknowledged	Low Tank Level
☐	6/7/19, 11:41 AM	Tank 200	Critical	Cleared, Unacknowledged	Low Tank Level

Details | Notes

On Active

mode	Below Setpoint
setpointA	20
Event Value	3
DynamicData	995.0933333333332
name	Low Tank Level
Event Time	6/7/19, 11:46 AM
label	Low Tank Level
StaticData	Folsom Plant
priority	Critical
displayPath	Tank 200

Alarm Grouping

Grouping alarms in Ignition is an important concept. Instead of seeing every alarm in a single view, often times, you may want to view alarms for a particular area of your plant, or for a specific set of alarms.

There are a several ways to group alarms. One way is to use a folder structure which requires you to organize your Tags into a hierarchy. Another way is to use the Display Path field in the alarm configuration. The third way, and the most recommended way, is to use associated data. It's a common design practice to associate alarm groupings on the associated data of the alarm.

Alarm Grouping

Watch the Video

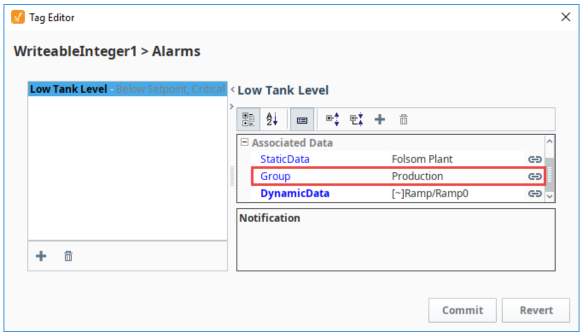
Creating an Alarm Grouping

This example uses the WriteableInteger Tag that we used in the example above.

1. Use an existing alarm that you already configured. If you don't have one, create a new alarm.

2. Right-click on your **Tag** and select **Edit Tags**. In the **Tag Editor,** scroll down to Alarms, and next to 'No alarms,' click on the **Edit** ✎ icon and click on your alarm. Click the **Add** ➕ icon above the alarm properties to add new associated data for the alarm.

3. Scroll to the bottom of the property list, and you'll see a new associated data property was added. In this example, make the associated data property a static value, and rename **'New Data'** to **'Group'** with a value that represents your Group (i.e., Production).



Tag Editor

WriteableInteger1 > Alarms

Low Tank Level - Below Setpoint, Critical

Low Tank Level

Associated Data

StaticData	Folsom Plant
Group	Production
DynamicData	[~]Ramp/Ramp0

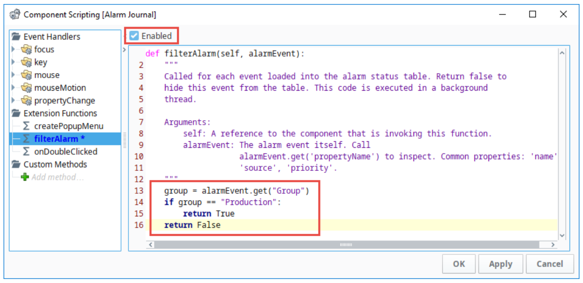
Notification

4. Click **Commit** to save your alarm edits, and then press **OK**. The alarm will now always have the **'Group'** designation associated with it. This can be used for filtering or part of your alarm pipeline notifications.

5. Before you can use the Alarm Journal to filter on your new associated data property, edit the **'filterAlarm'** Extension Function to include your new associated data property by selecting the **Alarm Journal component**, and right clicking on **Scripting**. Under the **Extension Functions** folder, click on **filterAlarm**, then click on the **Enabled** checkbox to enable the script. Edit the script to add your new associated property and property value. You can copy the new code below to add to your filterAlarm script.

filterAlarm scripting function

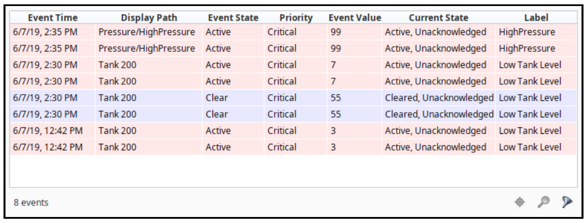
```
group = alarmEvent.get("Group")
if group == "Production":
        return True
return False
```



☑ Enabled

```
def filterAlarm(self, alarmEvent):
    """
    Called for each event loaded into the alarm status table. Return false to
    hide this event from the table. This code is executed in a background
    thread.

    Arguments:
        self: A reference to the component that is invoking this function.
        alarmEvent: The alarm event itself. Call
                    alarmEvent.get('propertyName') to inspect. Common properties: 'name'
                    'source', 'priority'.
    """
    group = alarmEvent.get("Group")
    if group == "Production":
        return True
    return False
```

6. Now, you're good to go! Open the Alarm Journal component and filter on your associated data. The Alarm Journal Table shown below displays all the alarms that are associated with the group, Production.



Event Time	Display Path	Event State	Priority	Event Value	Current State	Label
6/7/19, 2:35 PM	Pressure/HighPressure	Active	Critical	99	Active, Unacknowledged	HighPressure
6/7/19, 2:35 PM	Pressure/HighPressure	Active	Critical	99	Active, Unacknowledged	HighPressure
6/7/19, 2:30 PM	Tank 200	Active	Critical	7	Active, Unacknowledged	Low Tank Level
6/7/19, 2:30 PM	Tank 200	Active	Critical	7	Active, Unacknowledged	Low Tank Level
6/7/19, 2:30 PM	Tank 200	Clear	Critical	55	Cleared, Unacknowledged	Low Tank Level
6/7/19, 2:30 PM	Tank 200	Clear	Critical	55	Cleared, Unacknowledged	Low Tank Level
6/7/19, 12:42 PM	Tank 200	Active	Critical	3	Active, Unacknowledged	Low Tank Level
6/7/19, 12:42 PM	Tank 200	Active	Critical	3	Active, Unacknowledged	Low Tank Level

8 events

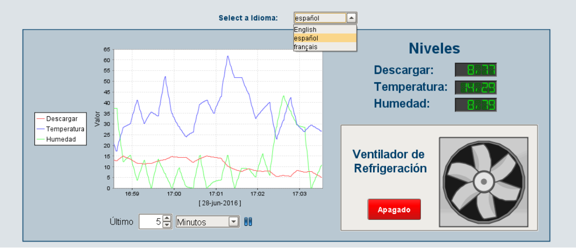
Related Topics ...

- Configuring Alarms
- Extension Functions
- Alarm Status Table
- Alarm Journal Table

Localization and Languages

Translating Ignition

Localization allows you to translate text into multiple languages in a project for display on client screens. The localization feature allows users located in different countries to set their default language so client screens can be displayed in their native language. The user can easily choose which language their Client displays with a click of a button! Text contained in components, that have their properties correctly mapped with localization, will change to reflect the language the user has selected. There is a special Language Selector dropdown list that displays the available translation mapping options for these translations.



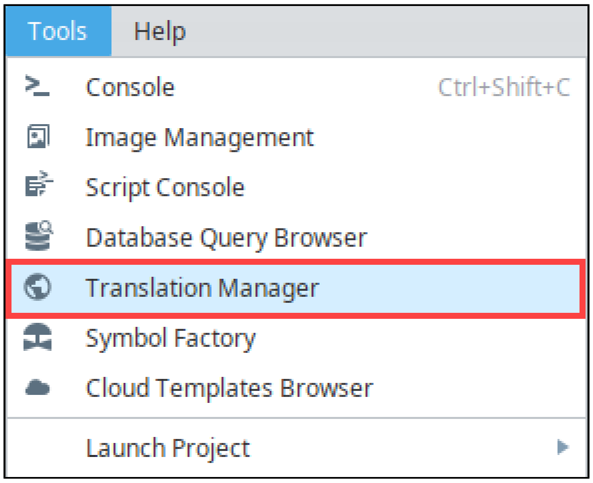
On this page

...

Store All Translations in One Central Location

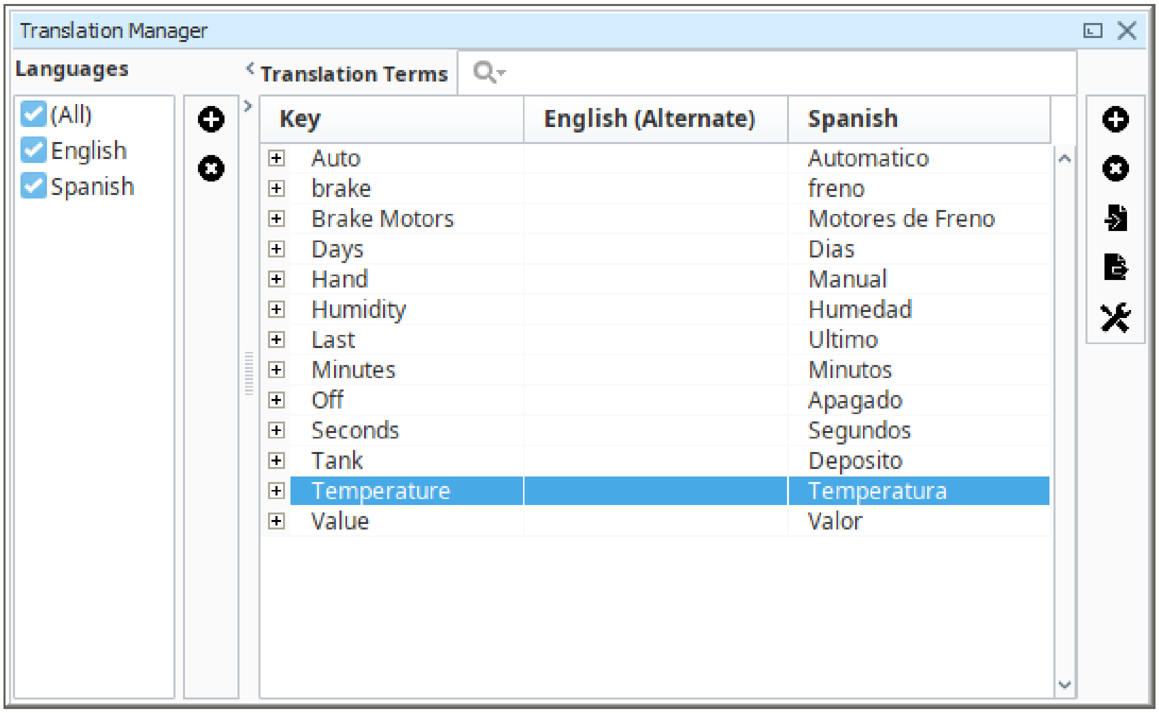
Translation Manager

All translations are stored centrally in the Gateway and are distributed to each Client and Designer. All projects share the same translations, and those translations can be used in other locations, such as Gateway scripts, and alarm messages. The full translation database can be viewed and edited in the **Translation Manager** tool located on the Tools menu the Designer.



You can add new terms and phrases to the Transaction Manager. Make your phrases as long or short as you like with single words or whole paragraphs. It allows users to easily import and export translatable components and share them across projects and with other users. It also has the ability to define new languages.

A central term database and support are automatically built in to all component text properties, as well as other text-based properties, such as alarm messages. Aside from defining translations for terms, there is usually no other work needed to take advantage of the translation system.



☑ (All)
☑ English
☑ Spanish

Key	English (Alternate)	Spanish
Auto		Automatico
brake		freno
Brake Motors		Motores de Freno
Days		Dias
Hand		Manual
Humidity		Humedad
Last		Ultimo
Minutes		Minutos
Off		Apagado
Seconds		Segundos
Tank		Deposito
Temperature		Temperatura
Value		Valor

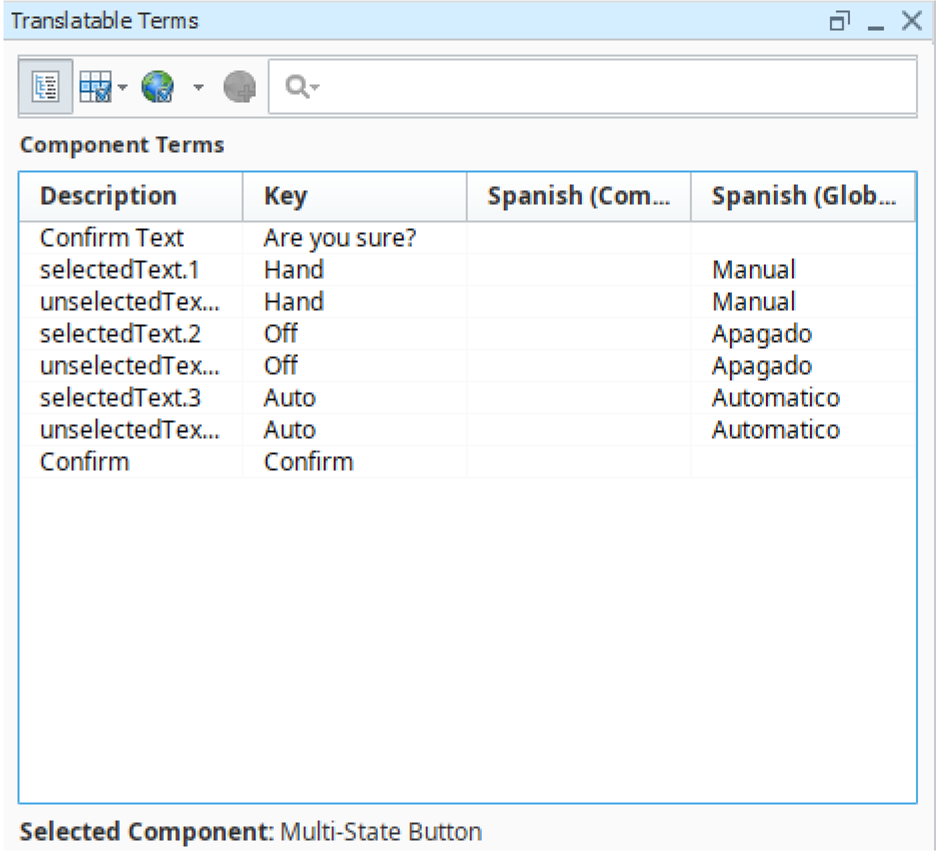
Translation Database and Term Lookup

Terms are referenced in the translation database using direct string comparison. Ignition has special term **Keys** that are used to identify when to translate text. The base term may be a user-readable string such as "Start", or any special code you want such as "START_COMMAND" or "#start", which would have an alternate translation for English defined as "Start." This way you can quickly identify which terms will be translated when working in the Designer. To keep simple, the English key is automatically used if no match is found for the requested language.

Quickly Identify Fields to Translatable Terms

You can quickly and easily identify, and translate all terms by selecting a container from your Project Browser and opening the Translatable Terms panel.

⚠ If you don't see the Translatable Terms panel, go to the menubar and select **View > Panels** and check **Translatable Terms**. Here, you can see all terms for each component in the container, and if each term has an associated translation for the requested language.



Translatable Terms

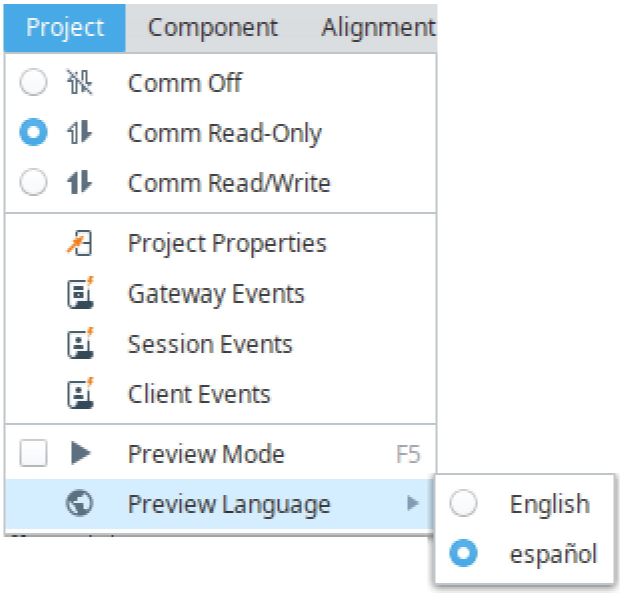
Component Terms

Description	Key	Spanish (Com...	Spanish (Glob...
Confirm Text	Are you sure?		
selectedText.1	Hand		Manual
unselectedTex...	Hand		Manual
selectedText.2	Off		Apagado
unselectedTex...	Off		Apagado
selectedText.3	Auto		Automatico
unselectedTex...	Auto		Automatico
Confirm	Confirm		

Selected Component: Multi-State Button

Previewing Translations

You can easily preview your translations in the Designer. To change the preferred language for Preview Mode, use the **Preview Language** menu item under the **Project** menu in the Designer. When you put the Designer into Preview Mode, everything on your windows will be automatically translated. Using the Language Selector Component or scripting can temporarily change the current language, and the language will revert back any time Preview Mode is disabled. The Designer will remember your choice, even when the Designer is restarted.



○ Comm Off
● Comm Read-Only
○ Comm Read/Write

Project Properties
Gateway Events
Session Events
Client Events

□ Preview Mode F5
Preview Language

○ English
● español

In This Section ...

Creating Translation Lists

Project Translation Overview

Project translation is also known as Client localization. This is how the user can choose their language-of-choice for the project they are working with. When the user selects a language, the Client will change its text in accordance with the project's translation settings. Translations work with all components (i.e., Labels, Charts, Multi-State Buttons, etc.,).

Translation Notes

- Text within Vision Table components (those under the "Table" group) can not be translated. However the Alarm Status Table and Alarm Journal Table components do support translations.
- For Text elements that have bindings, the translation occurs after the binding is evaluated.

The Translation Manager allows you to view all the terms from all your projects in the global translation database. You can add, edit, and remove terms, and provide translations, as well as allow users to easily import and export translatable components. The translation database shares terms across all projects and with other users.

On this page ...

Setting up New Languages, Terms, and Translations

Setting up new languages and adding terms for translation is pretty easy. For every element of text on the screen, you can provide a translation in the desired language. When a user logs into a project, they can choose their preferred language, and all the text will be displayed in their preferred language. They can also have their preferred language defined in user settings so when a user logs into the Client, their preferred language is automatically enabled.

Component vs Global Terms

There are two types of terms used in the translation system: Component and Global. Both behave a little differently, and it's important to know the difference.

- **Component Terms** - Component level terms are specific to that one component, and translations are added to the translation system using the Translatable Terms Panel. Any component level translation will automatically take precedence in the event there is a matching global term.

- **Global Terms** - Global terms and translations are available in both the Translatable Terms Panel and in the Translation Manager's global database. The global translation database provides language translations on all components, in all windows, and on all projects. You can setup global translations so that anywhere a term is used within your project, it will get replaced with that global translation. Although, there are times, when a component might have a different context than the global translation. For this reason, you may want to create a component level translation to take precedence over the global translation. For example, some users may work in a specific window of a project, and what is typically called a 'Tank,' this group of users refer to it as a 'Barrel.' You can create a component level translation for 'Barrel' to take precedence on the root container of a window. This way the users see the term in the context that is familiar to them.

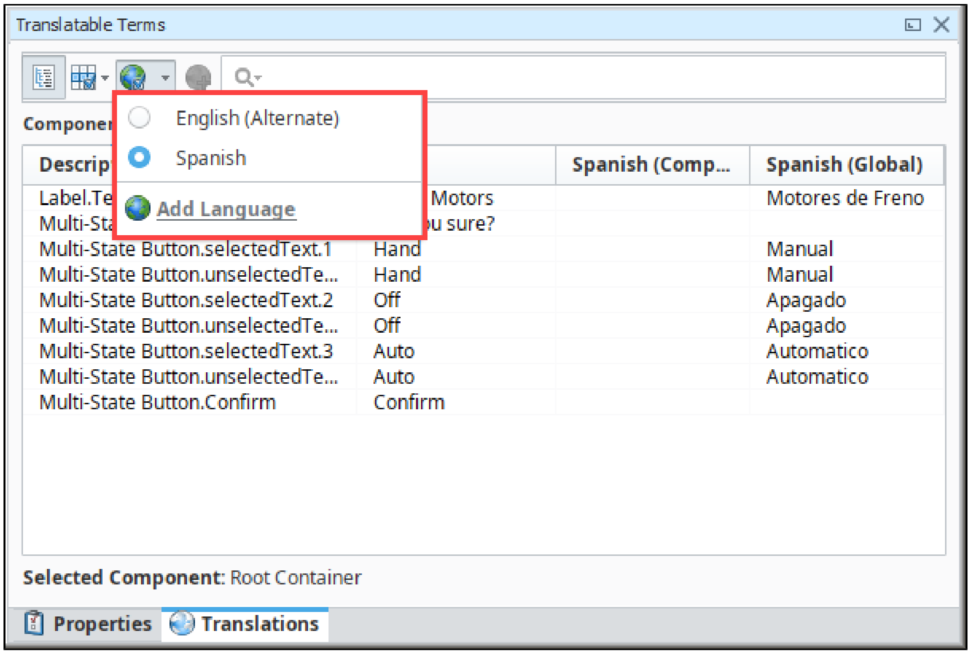
INDUCTIVE UNIVERSITY

Introduction to Project Translation

Watch the Video

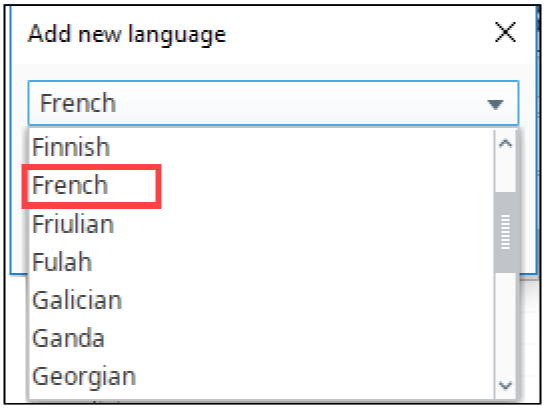
Add a New Language

1. In **Designer,** right click in your workspace and select **Translations** from the dropdown. The **Translatable Terms** window will open.
2. Select the **Earth** icon, and click **Add Language**. By default, the English language is defined.



Descrip		Spanish (Comp...	Spanish (Global)
Label.Te	Motors		Motores de Freno
Multi-Sta	u sure?		
Multi-State Button.selectedText.1	Hand		Manual
Multi-State Button.unselectedTe...	Hand		Manual
Multi-State Button.selectedText.2	Off		Apagado
Multi-State Button.unselectedTe...	Off		Apagado
Multi-State Button.selectedText.3	Auto		Automatico
Multi-State Button.unselectedTe...	Auto		Automatico
Multi-State Button.Confirm	Confirm		

3. Select a new language from the dropdown. That's all there is to it! You can create as many languages as you need for your project.

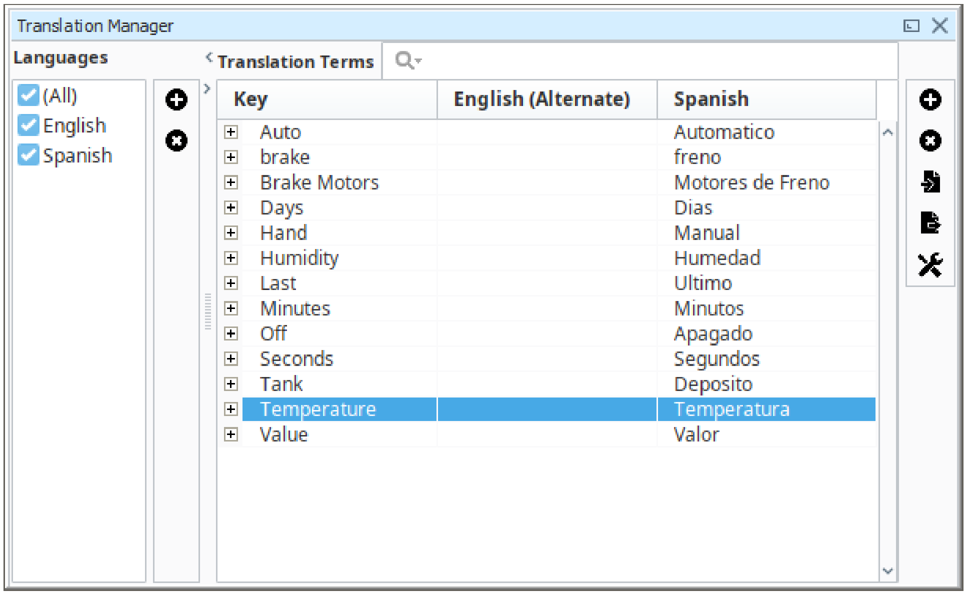


Add Global Translation Terms

Creating global translations is always a good first choice over the component level translation because most of your translated terms will be used across all projects unless you want one particular term to be different from the global translated term.

There are a couple of ways to add terms to the Translation Manager. You can add new terms directly to the Translation Manager or by selecting terms from the Translation Terms Panel.
Let's add some terms directly to the Translation Manager.

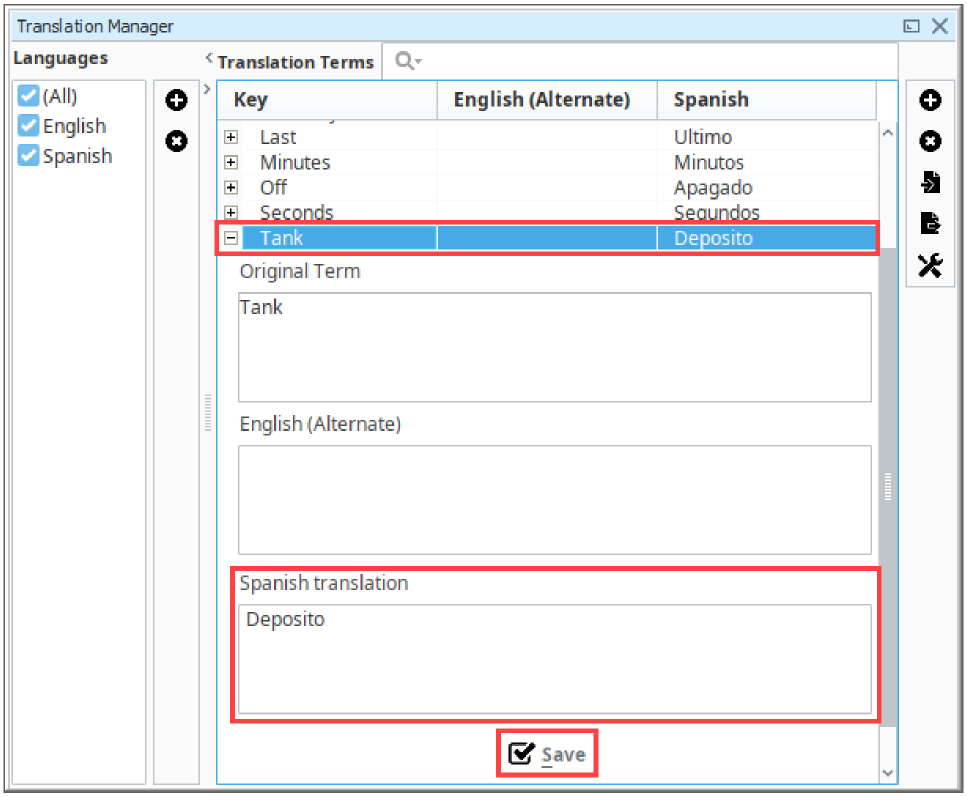
1. From the menu bar at the top, go to **Tools > Translation Manager**.
2. The Translation Manager window will open, and you can view all the translatable terms. This example has terms that were already added and translated.
3. On the right side of the window, there is a list of icons where you can add, delete, import, export, and edit the Translation settings.



☑ (All)
☑ English
☑ Spanish

Key	English (Alternate)	Spanish
Auto		Automatico
brake		freno
Brake Motors		Motores de Freno
Days		Dias
Hand		Manual
Humidity		Humedad
Last		Ultimo
Minutes		Minutos
Off		Apagado
Seconds		Segundos
Tank		Deposito
Temperature		Temperatura
Value		Valor

4. To add a new global term, click on the **Add** ⊕ icon. Enter the term (Tanks), and click **Save**.
5. Double click on the row of the new term, and under **Spanish translation**, enter the word "Deposito." Click **Save**.



☑ (All)
☑ English
☑ Spanish

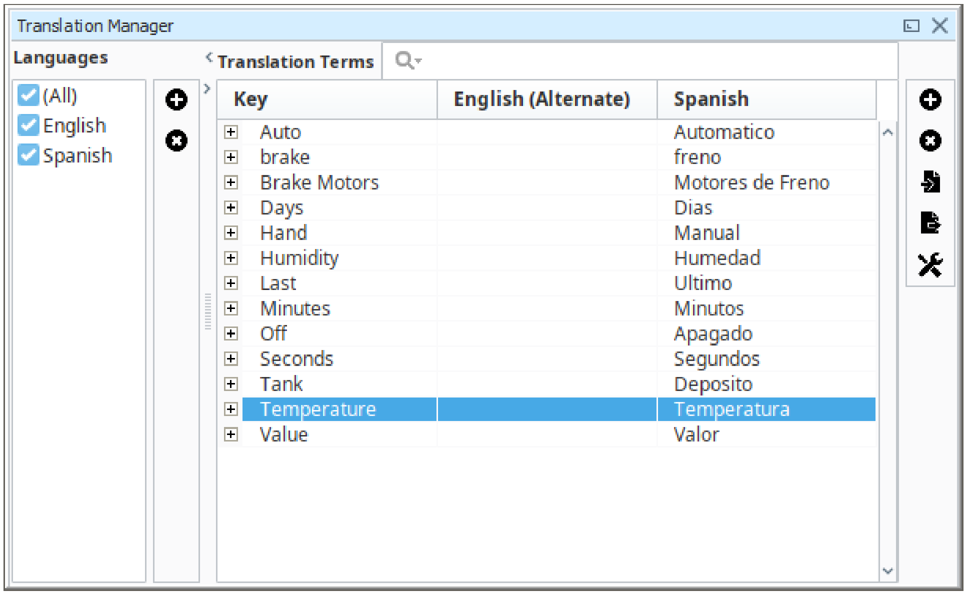
Key	English (Alternate)	Spanish
Last		Ultimo
Minutes		Minutos
Off		Apagado
Seconds		Segundos
Tank		Deposito

Original Term: Tank

English (Alternate):

Spanish translation: Deposito

6. To view all the global terms added to a project, go to **Tools > Translation Manager.** If you want to provide a translation to a term that has not been translated, double click on the row of the term, enter the translation, and click **Save**.



Translation Manager

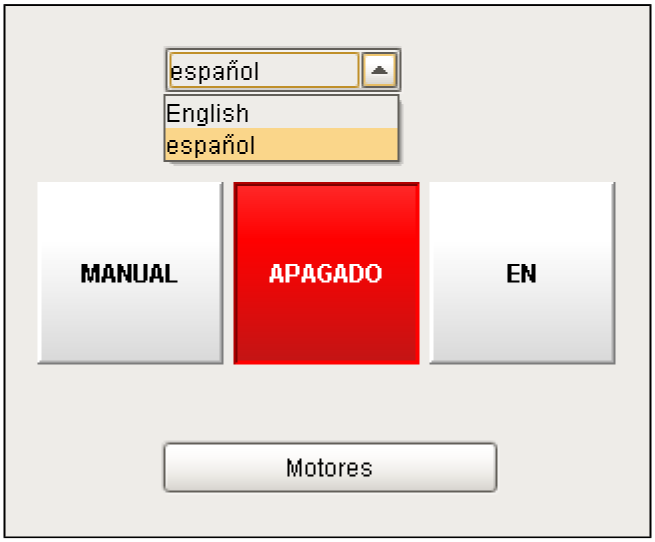
Languages

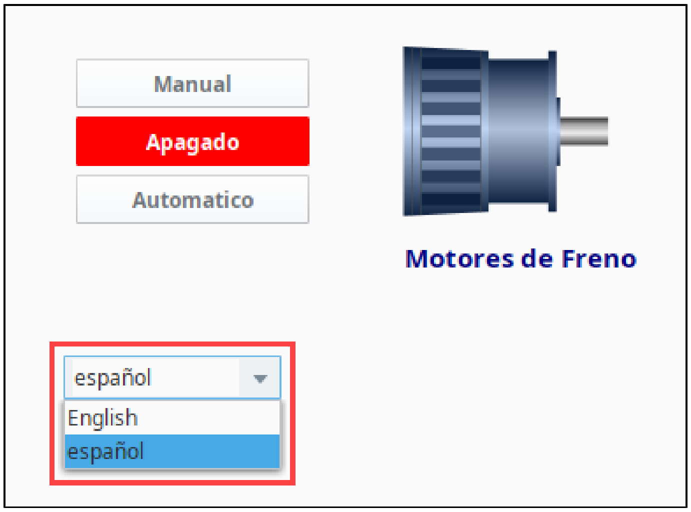
☑ (All)
☑ English
☑ Spanish

Translation Terms

Key	English (Alternate)	Spanish
Auto		Automatico
brake		freno
Brake Motors		Motores de Freno
Days		Dias
Hand		Manual
Humidity		Humedad
Last		Ultimo
Minutes		Minutos
Off		Apagado
Seconds		Segundos
Tank		Deposito
Temperature		Temperatura
Value		Valor

7. Now test it out. Drag a **Language Selector** component to your window. Go into **Preview Mode,** and toggle the languages. You will see the languages for the text switch between English and Spanish for these two components.



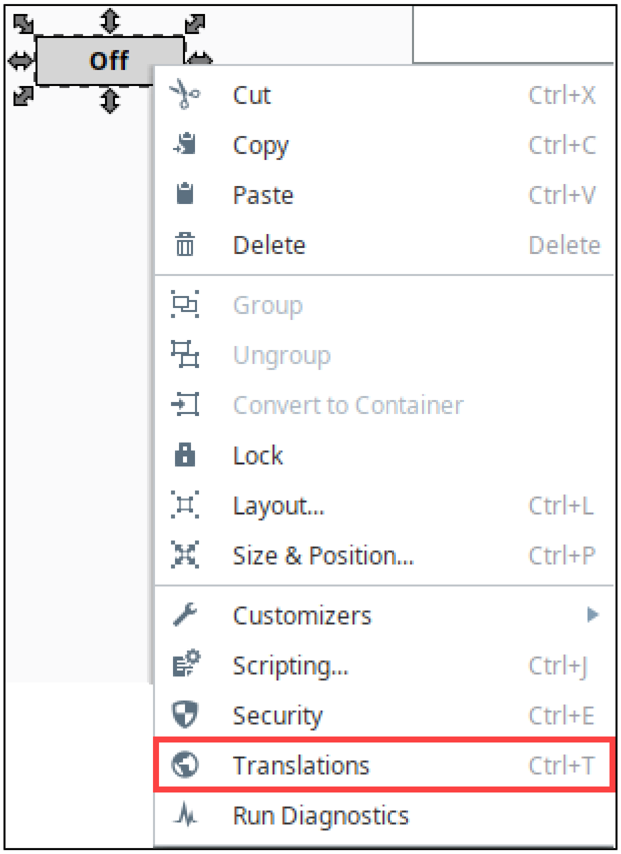


Add Component Level Translation Terms

Component level translation terms are only available in Vision.

What's nice about adding component translation terms is that the Translatable Terms panel provides all the terms for that component on the window. All you have to do is add the translations. The Translatable Terms Panel is a central place to manage all the translations associated with a specific component. One important reason to use a component level term is if it is going to be different from the global term. Just remember, that because a term is translated at the component level doesn't mean that other components get to use it. Typically, it is used for that component only.

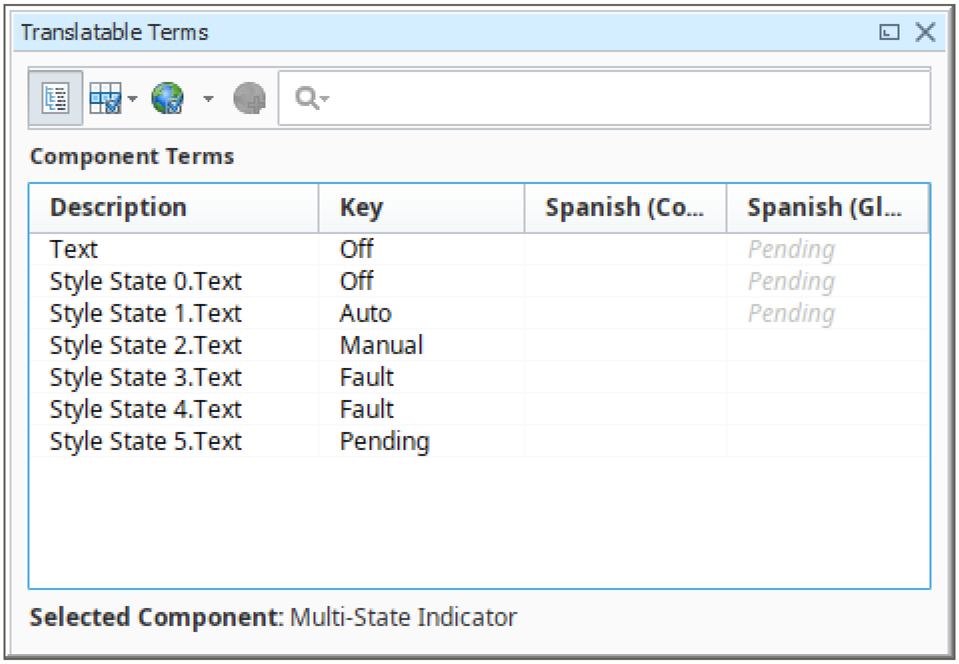
1. In **Designer**, drag a **Multi-State Indicator** from the Component Palette in to your workspace.
2. Right click on the component and scroll down to **Translations**. The Translatable Terms panel will open and display all the terms for the specific component. Alternatively, you can right-click on the Root Container of a window and select **Translations**. The **Translatable Terms Panel** will display all the terms for all the components on the window.



Translatable Term Options

In the Translatable Terms Panel, you have the option of adding your translation to the Component, Translation Manager global database, or marking it for translation at a later time. You'll notice under the Component Terms section of the Translatable Terms window that there are four columns; Component Description, Key, Component translation, and Global translation.

You'll see below, all the terms for the Multi-State Button, and none of the terms have translations added. If you don't see the language you want your terms translated to, click the Earth icon 🌐, and select the language (i.e., Spanish). You will then see a column for your component translation and global translation.



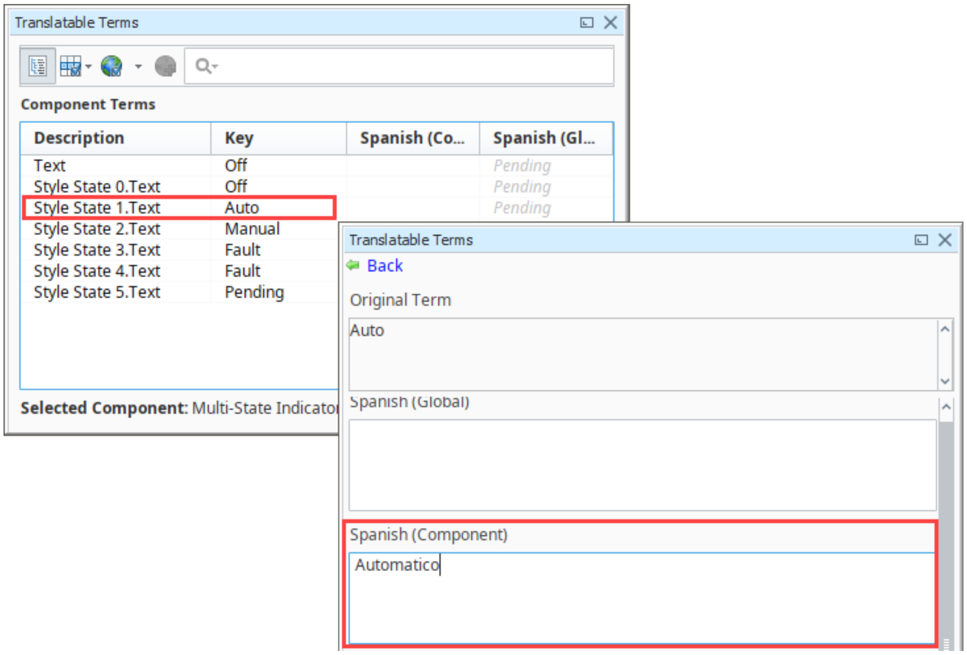
Translatable Terms

Component Terms

Description	Key	Spanish (Co...	Spanish (Gl...
Text	Off		Pending
Style State 0.Text	Off		Pending
Style State 1.Text	Auto		Pending
Style State 2.Text	Manual		
Style State 3.Text	Fault		
Style State 4.Text	Fault		
Style State 5.Text	Pending		

Selected Component: Multi-State Indicator

3. To add translations for your component terms, double click the row of your term (i.e. ON). In the **Spanish Component box**, enter the translation (i.e, EN). You also have the option of creating a global translation on the Translatable Terms Panel as well. Then, click the **Back** link, and repeat this step for all your terms.



Translatable Terms

Component Terms

Description	Key	Spanish (Co...	Spanish (Gl...
Text	Off		Pending
Style State 0.Text	Off		Pending
Style State 1.Text	Auto		Pending
Style State 2.Text	Manual		
Style State 3.Text	Fault		
Style State 4.Text	Fault		
Style State 5.Text	Pending		

Selected Component: Multi-State Indicator

Translatable Terms

Back

Original Term

Auto

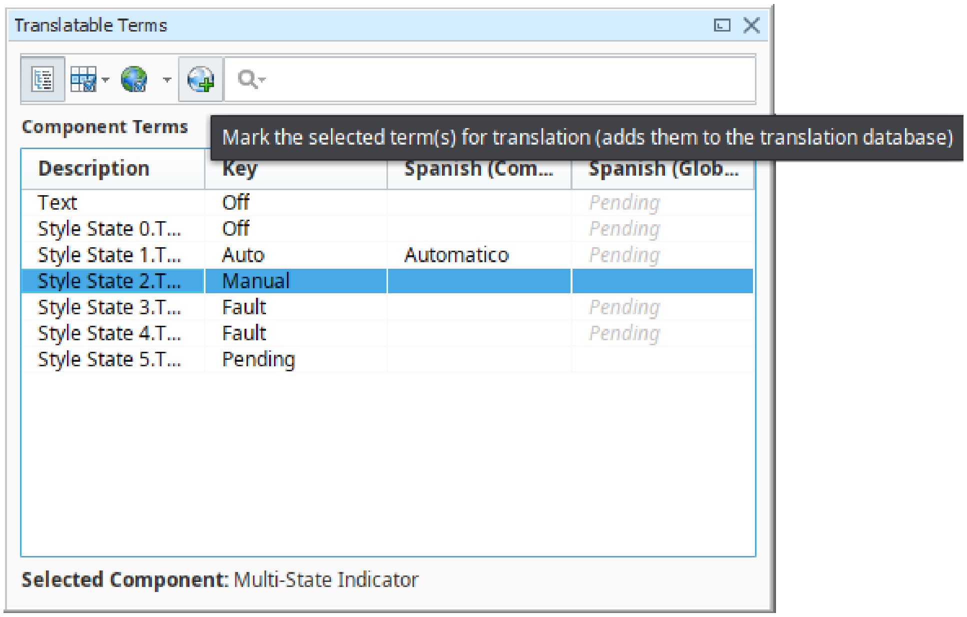
Spanish (Global)

Spanish (Component)

Automatico

4. Once you entered the translations for all your terms, you will see them on the Translatable Terms Panel in the Spanish (Component) column. If a term doesn't have a translation, you can select the term and mark it for translation by clicking the **Earth with a green plus icon**. This adds the term(s) to the translation database for someone else to add the translation.

   In this example, you'll notice several things about each of the terms so look closely at them. You'll find examples of both component and global level terms, a component level term, a global term, a component term that overrides a global term, and terms marked for translation.



Translatable Terms

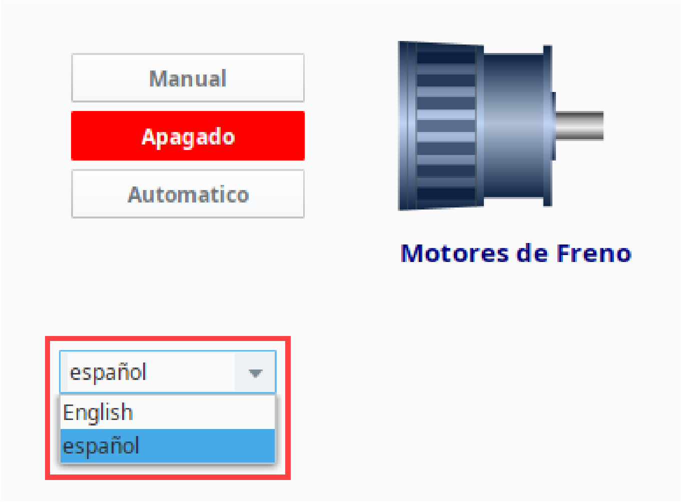
Component Terms

Mark the selected term(s) for translation (adds them to the translation database)

Description	Key	Spanish (Com...	Spanish (Glob...
Text	Off		*Pending*
Style State 0.T...	Off		*Pending*
Style State 1.T...	Auto	Automatico	*Pending*
Style State 2.T...	Manual		
Style State 3.T...	Fault		*Pending*
Style State 4.T...	Fault		*Pending*
Style State 5.T...	Pending		

Selected Component: Multi-State Indicator

5. To test it out, drag a **Language Selector** component from the Component Palette to your workspace. Go to **Preview Mode**, select Spanish from the dropdown to see the text change from English to Spanish for these two components.



Translation List Import and Export

The ability to import and export terms and translations can be extremely important particularly to a large project. It allows you disseminate terms and translations rapidly, send them to a third party for translation, and keeps all related projects up-to-date.

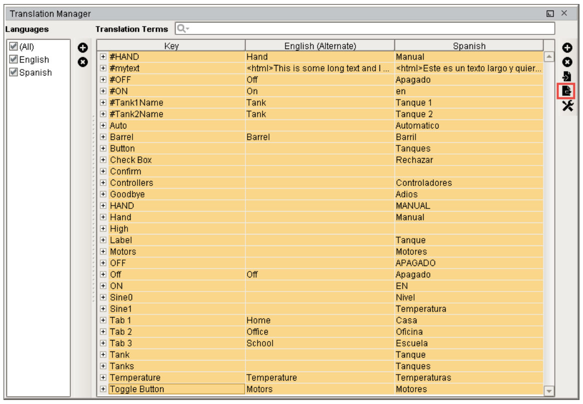
Import and Export Translations

Watch the Video

Export Terms from a Window

Let's export some terms from the Translation Manager. In this example, you can see all the terms that were added to the Translation Manager. You can export selected terms, or all terms from the Translation Manager.

1. Go to **Tools > Translation Manager**. Select all terms or only specific terms to export. Click on the **Export Terms** icon on the right side of the **Translation Manager** window.



Translation Manager

Languages

☑ (All)
☑ English
☑ Spanish

Translation Terms

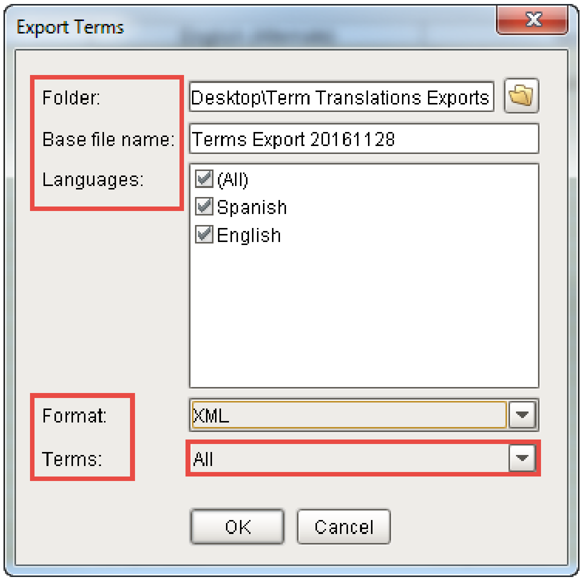
Key	English (Alternate)	Spanish
#HAND	Hand	Manual
#mytext	<html>This is some long text and I ...	<html>Este es un texto largo y quier...
#OFF	Off	Apagado
#ON	On	en
#Tank1Name	Tank	Tanque 1
#Tank2Name	Tank	Tanque 2
Auto		Automatico
Barrel	Barrel	Barril
Button		Tanques
Check Box		Rechazar
Confirm		
Controllers		Controladores
Goodbye		Adios
HAND		MANUAL
Hand		Manual
High		
Label		Tanque
Motors		Motores
OFF		APAGADO
Off	Off	Apagado
ON		EN
Sine0		Nivel
Sine1		Temperatura
Tab 1	Home	Casa
Tab 2	Office	Oficina
Tab 3	School	Escuela
Tank		Tanque
Tanks		Tanques
Temperature	Temperature	Temperaturas
Toggle Button	Motors	Motores

2. The Exported Terms window will open. Enter the following information:

    - Select the **Folder** location where you want to put your exported terms.
    - Enter the **Base file name**. (File name of your exported terms).
    - Select the appropriate **Languages**. You can have multiple languages selected depending on the number of language translations you are using (i.e., English, Spanish, and French). This example exports two files: one for the English translation, and one for the Spanish translation.
    - From the **Format** dropdown list, select either **PROPERTIES** or **XML**. Both formats behave the same, the only difference is that the XML format supports UTF-8 encoding directly. A person can now translate the XML file directly and import it back into Ignition's Translation Manager.
    - From the **Terms** dropdown list, choose either **Selected** or **All**.

    > **For Specific Terms**
    >
    > If you want to select specific terms to export, choose **'Selected,'** otherwise, **'All'** terms will be exported from the dropdown list at the bottom of the window.

    - Press **OK.**



Export Terms

Folder: Desktop\Term Translations Exports

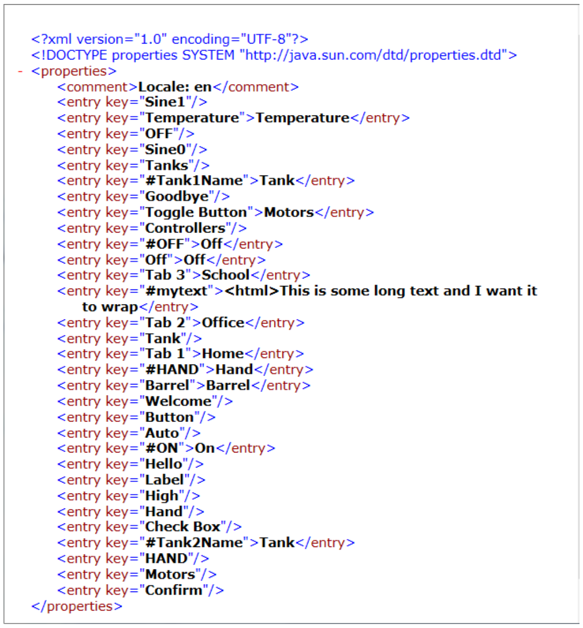
Base file name: Terms Export 20161128

Languages:
☑ (All)
☑ Spanish
☑ English

Format: XML

Terms: All

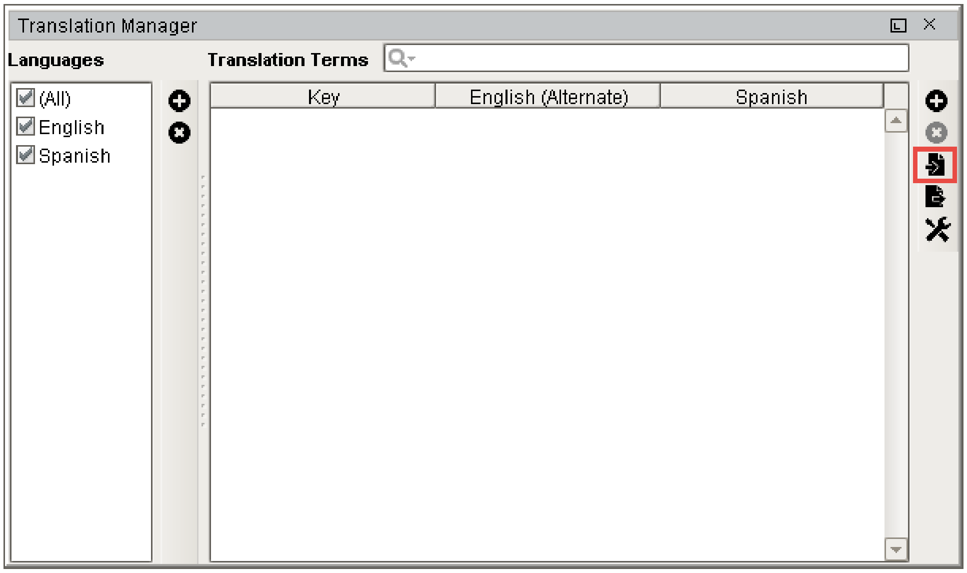
3. Go to the folder location and check to see if your exported files are there. You will see **one file exported file for each language**. Open the English exported file and verify that all the terms you intended to export are there.



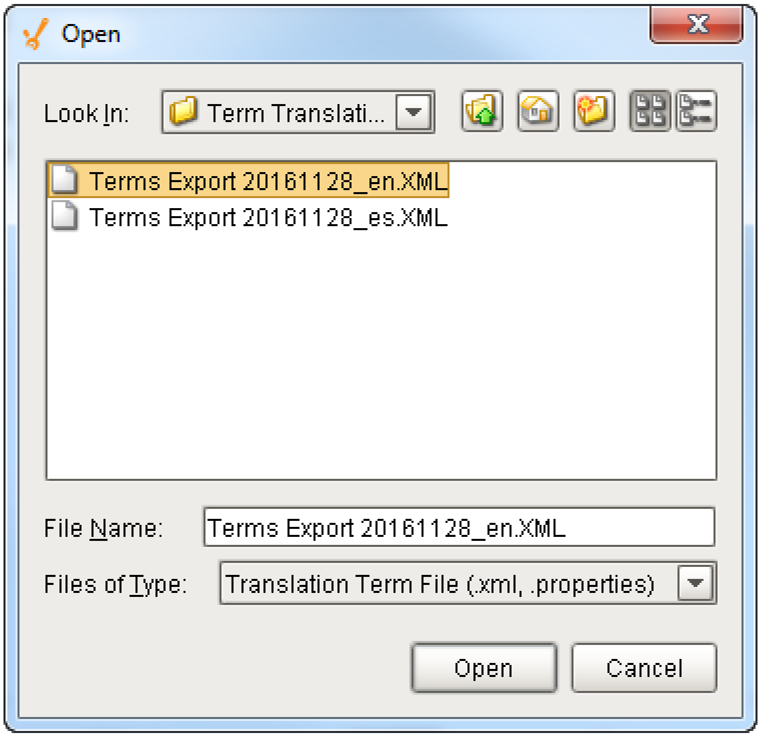
```
  <?xml version="1.0" encoding="UTF-8"?>
  <!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
- <properties>
      <comment>Locale: en</comment>
      <entry key="Sine1"/>
      <entry key="Temperature">Temperature</entry>
      <entry key="OFF"/>
      <entry key="Sine0"/>
      <entry key="Tanks"/>
      <entry key="#Tank1Name">Tank</entry>
      <entry key="Goodbye"/>
      <entry key="Toggle Button">Motors</entry>
      <entry key="Controllers"/>
      <entry key="#OFF">Off</entry>
      <entry key="Off">Off</entry>
      <entry key="Tab 3">School</entry>
      <entry key="#mytext"><html>This is some long text and I want it
          to wrap</entry>
      <entry key="Tab 2">Office</entry>
      <entry key="Tank"/>
      <entry key="Tab 1">Home</entry>
      <entry key="#HAND">Hand</entry>
      <entry key="Barrel">Barrel</entry>
      <entry key="Welcome"/>
      <entry key="Button"/>
      <entry key="Auto"/>
      <entry key="#ON">On</entry>
      <entry key="Hello"/>
      <entry key="Label"/>
      <entry key="High"/>
      <entry key="Hand"/>
      <entry key="Check Box"/>
      <entry key="#Tank2Name">Tank</entry>
      <entry key="HAND"/>
      <entry key="Motors"/>
      <entry key="Confirm"/>
  </properties>
```

Import Terms into Ignition's Translation Manager

1. Go to **Tools > Translation Manager**, and select the **Import Terms** icon from the right side of the Translation Manager window.



2. Navigate to the folder where your exported files reside, and choose the English file from the **Open** window, and click **Open**.



Look In: Term Translati...

Terms Export 20161128_en.XML

Terms Export 20161128_es.XML

File Name: Terms Export 20161128_en.XML

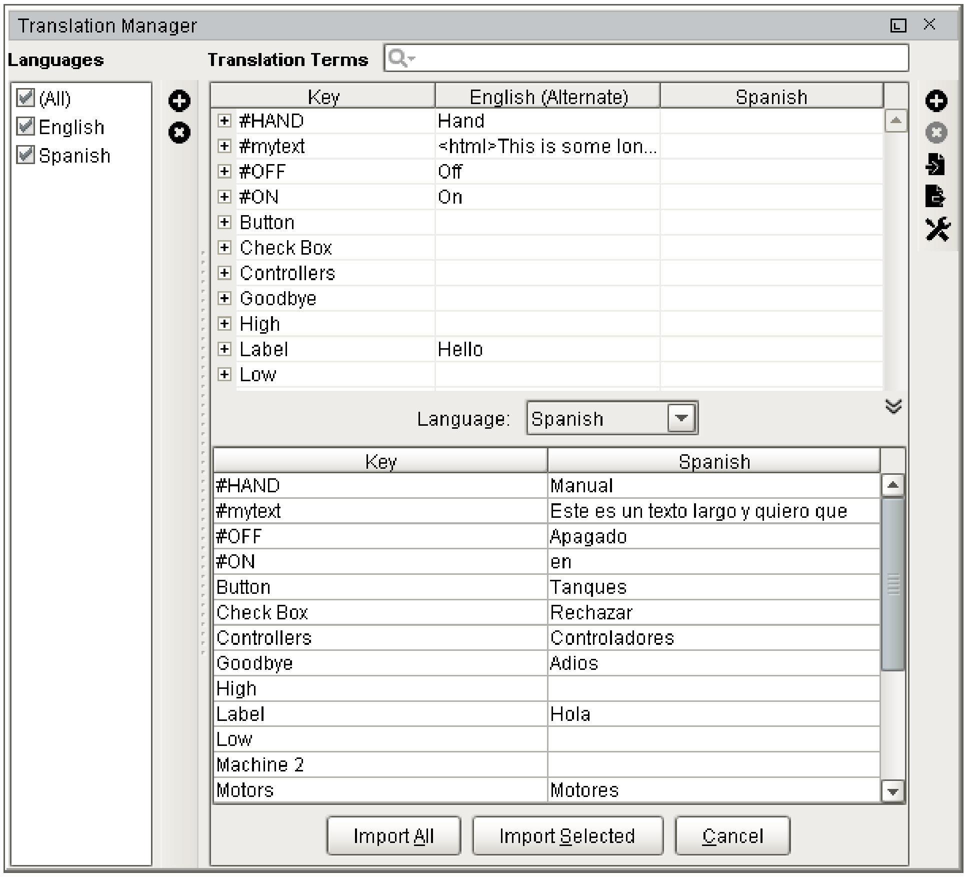
Files of Type: Translation Term File (.xml, .properties)

3. You have the option of importing selected terms or all terms. Click either **Import All** or **Import Selected** depending on what terms you want to import.

   Repeat this step to import each translation file. This example imports the Spanish translation file.

Imported Terms Overwrite Existing Terms

Be aware that existing terms will be overwritten by the imported term.



☑ (All)
☑ English
☑ Spanish

Key	English (Alternate)	Spanish
⊞ #HAND	Hand	
⊞ #mytext	<html>This is some lon...	
⊞ #OFF	Off	
⊞ #ON	On	
⊞ Button		
⊞ Check Box		
⊞ Controllers		
⊞ Goodbye		
⊞ High		
⊞ Label	Hello	
⊞ Low		

Language: Spanish

Key	Spanish
#HAND	Manual
#mytext	Este es un texto largo y quiero que
#OFF	Apagado
#ON	en
Button	Tanques
Check Box	Rechazar
Controllers	Controladores
Goodbye	Adios
High	
Label	Hola
Low	
Machine 2	
Motors	Motores

4. Check to see that all your terms were imported into the Translation Manager by going to **Tools > Translation Manager.**

Related Topics ...

- Switching the Current Language
- Localization Best Practices

Switching the Current Language

On this page

...

Language Selection

Because Ignition has multiple visualization systems, each system has separate ways to switch the current language.

Vision

In Vision, the language can be selected using the language selector dropdown before logging in to the client, or can be added to the client in the form of the language selector component.

Perspective

The following feature is new in Ignition version **8.0.2**
Click here to check out the other new features

In Perspective, the language is selected by changing the locale of the session by modifying the session.props.locale. This can be written to using a script when the session first opens, or something can be bound to it that the user can change.

Expression and Scripting Functions

You can look up translations using the following functions:

- Expression Function
  translate()

- Scripting Functions
  `system.util.translate()`
  `system.util.modifyTranslation()` scripting functions.

Translations are matched by looking for the base language value in the translation database. This is especially useful for message boxes and other warnings or errors that you show in your scripts.

System Considerations During Translation

The inherent operating system may affect Ignition's ability to provide a complete translation in certain circumstances.

For example, suppose a user selected the Spanish option from their Language Selector component. An English to Spanish translation term exist for the word "Information" as well as "Start." The message box script takes two parameters: one for the content while the other parameter is optional. If the second parameter is not included in the function then the resulting title of the message box will include the English word "Information." The following code is executed on a button clicked event handler.

```
message = system.util.translate("Start")
system.gui.messageBox(message)
```

What results is a message box with a translated content while the title remains in English. In addition, the accept button is translated as well even though the English word for "OK" does not exist in the Translation manager. This is due to the system level translations that exist for inherent language support.

A similar occurrence exist with the file open dialog window. The following code is executed on a button clicked event handler.

```
system.file.openFile()
```

The result is a translation for the title as well as the open and cancel buttons while the references to the "File Name" and the "Files of Type" remain in the operating system's language and thus are beyond the scope of Ignition's ability to translate.

Similar occurrences will appear in print and error dialog boxes.

Localization Best Practices

Best Practices

On this page

...

Before you begin using multiple languages in your projects, we thought we would start you off with a few best practices for using project terms and translations.

As you already know, there are global level translations which are available to all components and text elements in both the Translatable Terms Panel and the global database, and component level translations that are only specific to the component and only available in the Translatable Terms Panel.   It's good to understand a little about their behavior before investing a lot of manpower setting up your translations only to find out down the road that you might set them up differently after you become more familiar with them.

Global vs Component Terms

We recommend using global terms over component level terms because this way you will only have to add your term once since global terms are shared across all windows and projects. This will save you a little time from having to add the same term again if you discover that a component level term should extend beyond the specific component and shared across all projects. The only time you would want to use a component level term, is if you want it to be different from the global level term. This ensures that the component level term will override the global term.

Where the component level term takes precedence

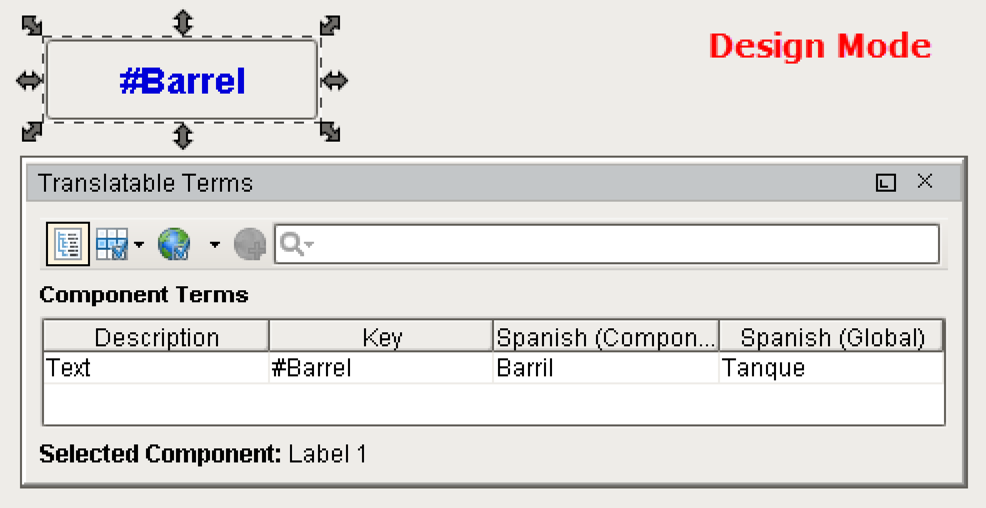
The component level term always takes precedence when there is a matching global term.

Using Codes

It's a good idea to use codes for the Key field of your Terms for any descriptions that you want to translate.(i.e: #introduction or #welcome_screen_info) so your global term won't ever accidentally translate on another window or component. (The **'#'** is just for convention, and is not a special character). Using codes works particularly well for long text strings such as introduction paragraphs. One thing to be very careful of is if you edit the codes in a minor way, it could potentially break the translation.

Here are a couple of examples:

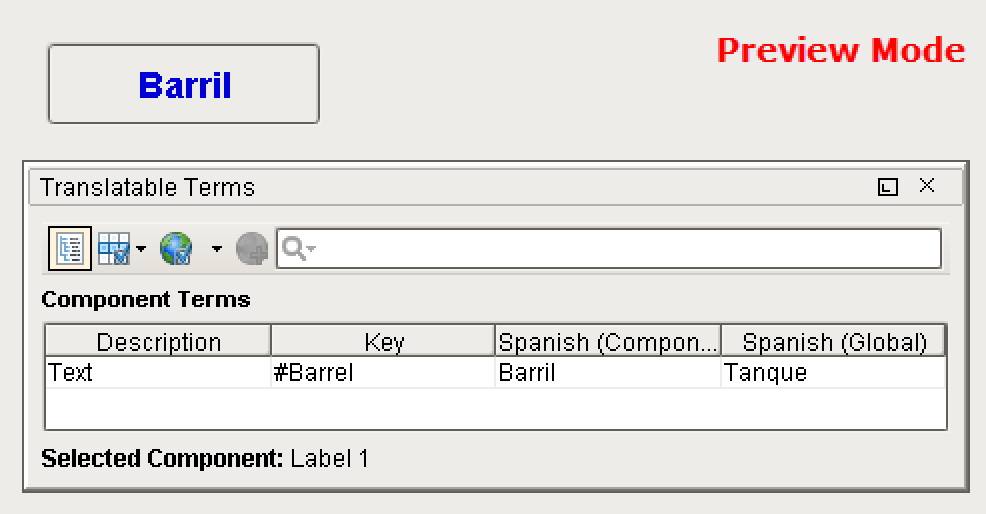
The first example is of a Label component with a Key field of **"#Barrel."** You'll notice the Label has global translation of **"Tanque"** and a component level translation of **"Barril."** By using the **"#,"** the component level translation of **"Barril"** was not overwritten by the global translation of **"Tanque"** by using the **"#."** Here is what the translation looks like in **Design Mode** and **Preview Mode.**



Component Terms

Description	Key	Spanish (Compon...	Spanish (Global)
Text	#Barrel	Barril	Tanque

Selected Component: Label 1

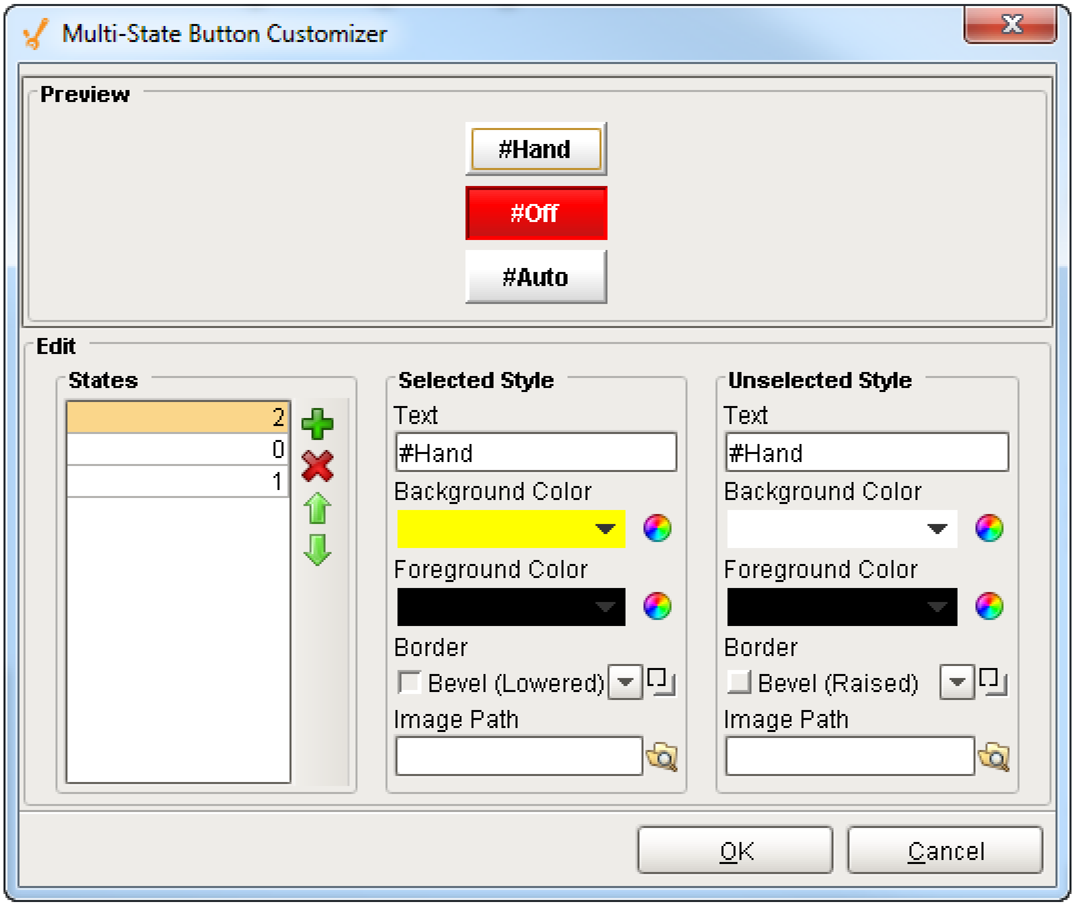


Component Terms

Description	Key	Spanish (Compon...	Spanish (Global)
Text	#Barrel	Barril	Tanque

Selected Component: Label 1

The **"#"** can also be used on components for text that you don't want accidentally translated. This example uses the Multi-State Button component. The only way to change the states / words on the buttons (i.e., Hand, Off, Auto) is to use the Multi-State Button Customizer. Add the **"#"** to the text on each of the three Keys so they don't get translated by global terms that have already been set up.



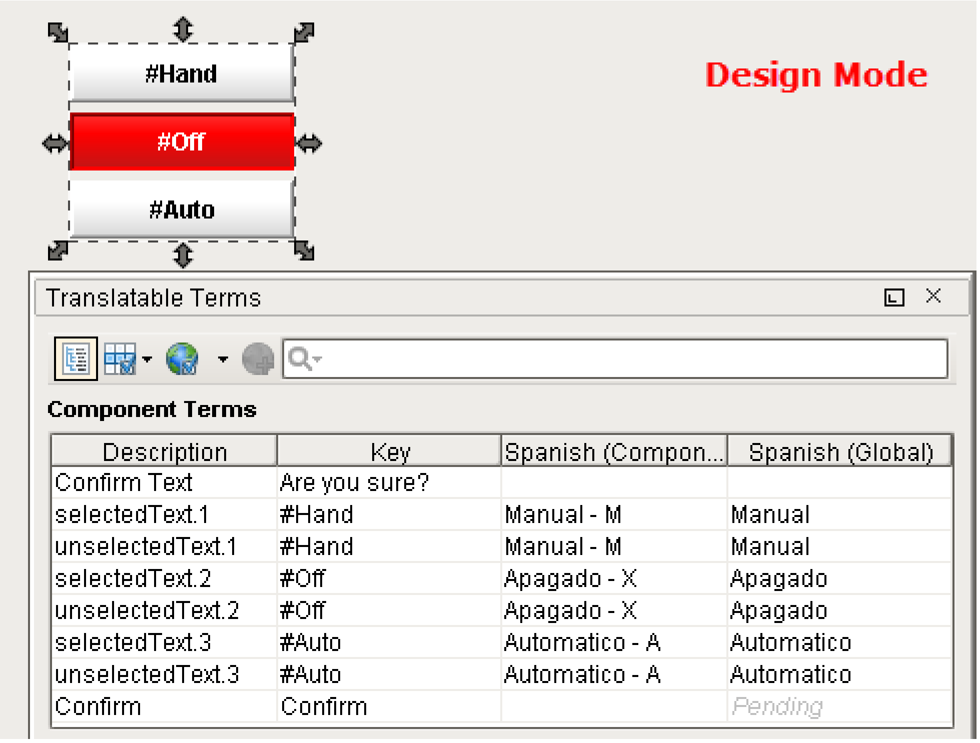
Multi-State Button Customizer

States
2
0
1

Selected Style
Text: #Hand
Background Color
Foreground Color
Border: ☐ Bevel (Lowered)
Image Path

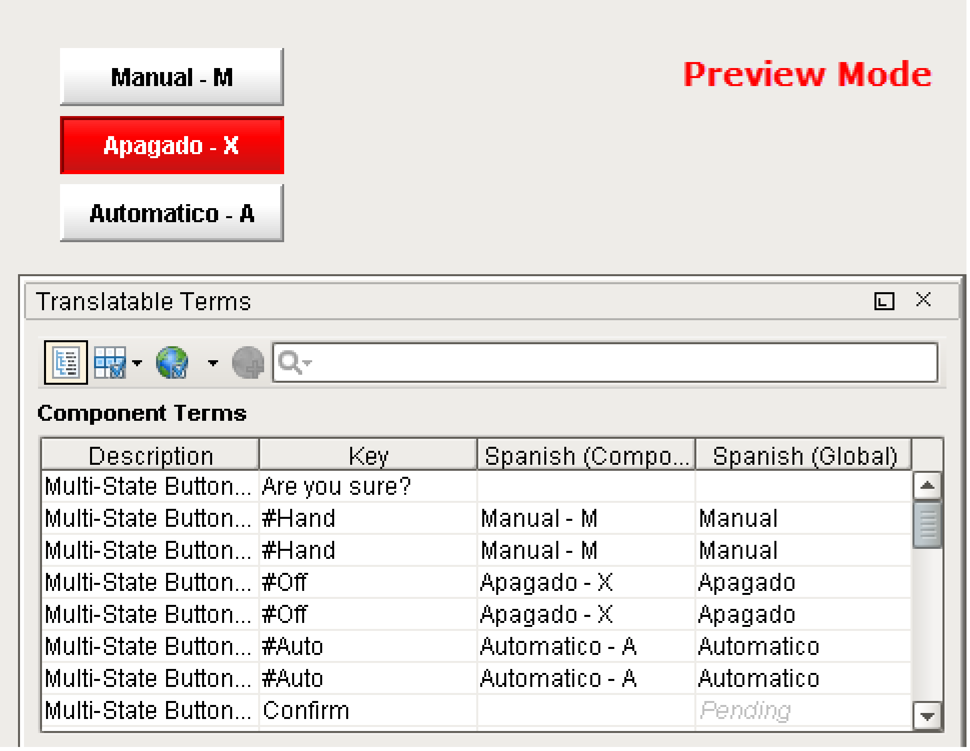
Unselected Style
Text: #Hand
Background Color
Foreground Color
Border: ☐ Bevel (Raised)
Image Path

Here's what the translations look like in **Design Mode** and **Preview Mode.**



Component Terms

Description	Key	Spanish (Compon...	Spanish (Global)
Confirm Text	Are you sure?		
selectedText.1	#Hand	Manual - M	Manual
unselectedText.1	#Hand	Manual - M	Manual
selectedText.2	#Off	Apagado - X	Apagado
unselectedText.2	#Off	Apagado - X	Apagado
selectedText.3	#Auto	Automatico - A	Automatico
unselectedText.3	#Auto	Automatico - A	Automatico
Confirm	Confirm		*Pending*



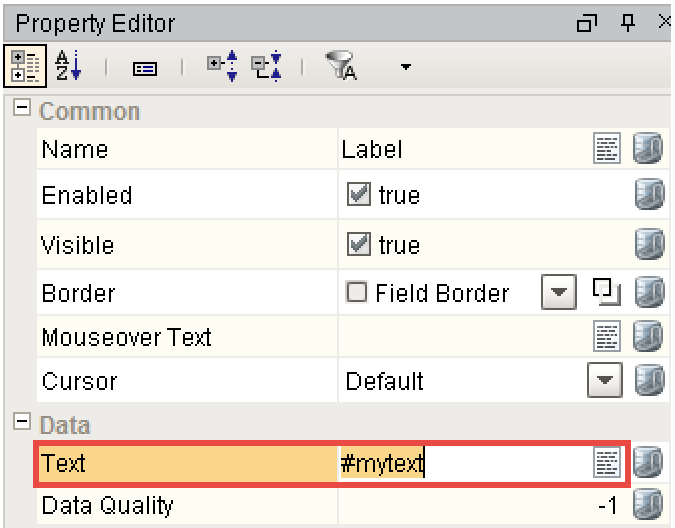
Component Terms

Description	Key	Spanish (Compo...	Spanish (Global)
Multi-State Button...	Are you sure?		
Multi-State Button...	#Hand	Manual - M	Manual
Multi-State Button...	#Hand	Manual - M	Manual
Multi-State Button...	#Off	Apagado - X	Apagado
Multi-State Button...	#Off	Apagado - X	Apagado
Multi-State Button...	#Auto	Automatico - A	Automatico
Multi-State Button...	#Auto	Automatico - A	Automatico
Multi-State Button...	Confirm		*Pending*

Using HTML for Text Wrapping

Another good practice is to use HTML to wrap text within your component's buttons, labels, etc.,. You can use it with translating terms that contain lots of text. HTML is just a good way to wrap text to fit within a label and button components.

1. In the **Property Editor** of a Label component, go to the **Text** property and give the Text a unique name (i.e., #Mytext).

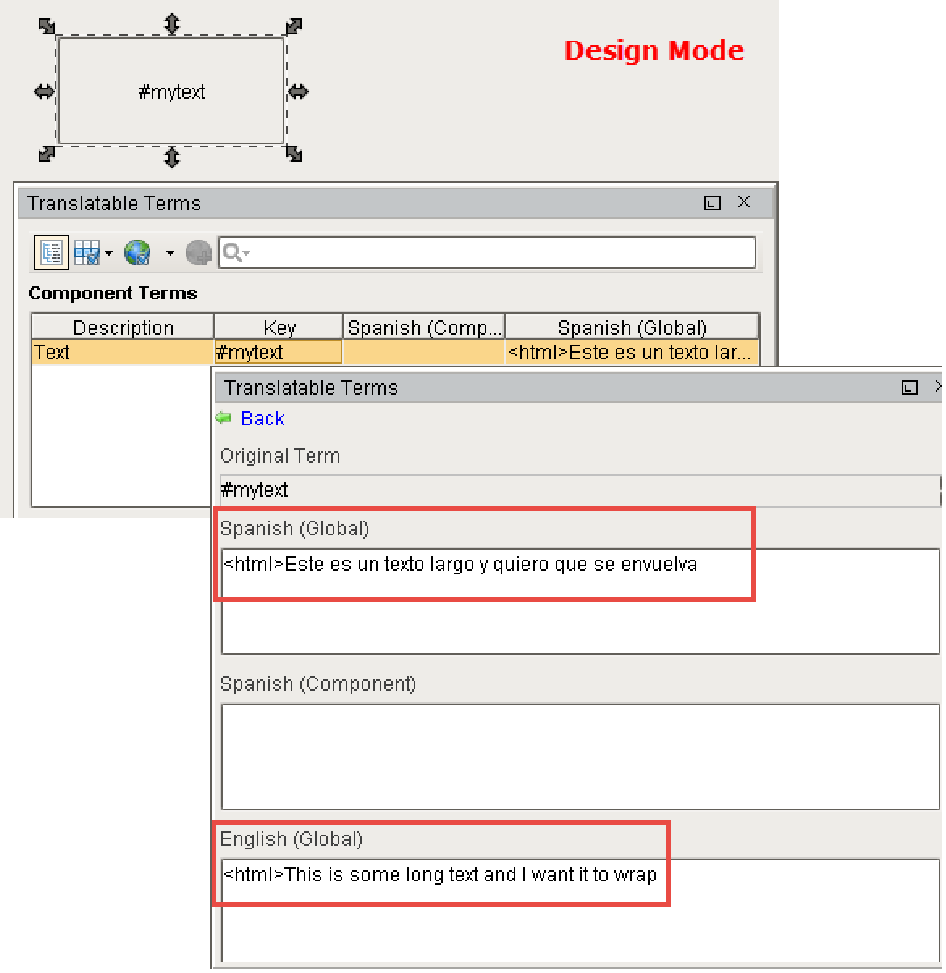


Common	
Name	Label
Enabled	☑ true
Visible	☑ true
Border	☐ Field Border
Mouseover Text	
Cursor	Default
Data	
Text	#mytext
Data Quality	-1

2. Right click on the **Label** component, and click on **Translations**. Double click on the row for Mytext. Enter the text as shown below for both the **Spanish Global** and English **Global translations**. Make sure you precede your text with **"<HTML>."** Click the **Back** link.

Mytext

```
#English
<html>This is some long text and I want to wrap it

#Spanish
<html>Este es un texto largo y quiero que se envuelva
```



Description	Key	Spanish (Comp...	Spanish (Global)
Text	#mytext		<html>Este es un texto lar...

Original Term

#mytext

Spanish (Global)

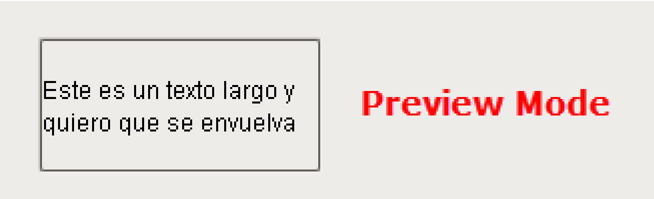
<html>Este es un texto largo y quiero que se envuelva

Spanish (Component)

English (Global)

<html>This is some long text and I want it to wrap

3. Once in **Preview Mode**, your long text will be wrapped!



Translating Built-in Terms

There are several built-in terms used by Ignition that are translatable. This page contains the Keys that can be used to translate these terms.

On this page ...

Built-in Access Denied

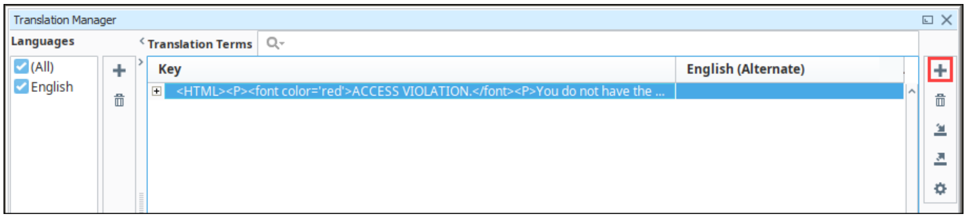
When Security Settings are configured on a window, the only restriction is **Do Not Open**, which prevents the window from opening based on User Roles. The text in this message box is not normally configurable, but can be altered via translations using the following steps.

1. Go to the top menubar in the Designer and select **Tools > Translation Manager**. This opens the Translation Manager window.
2. Copy the **Key** from the code block below, click the plus + icon and paste it in the **Add Item** field. Click **OK** and you'll notice the code will be displayed in the **Key** field.

   The line below should be used as the Key.

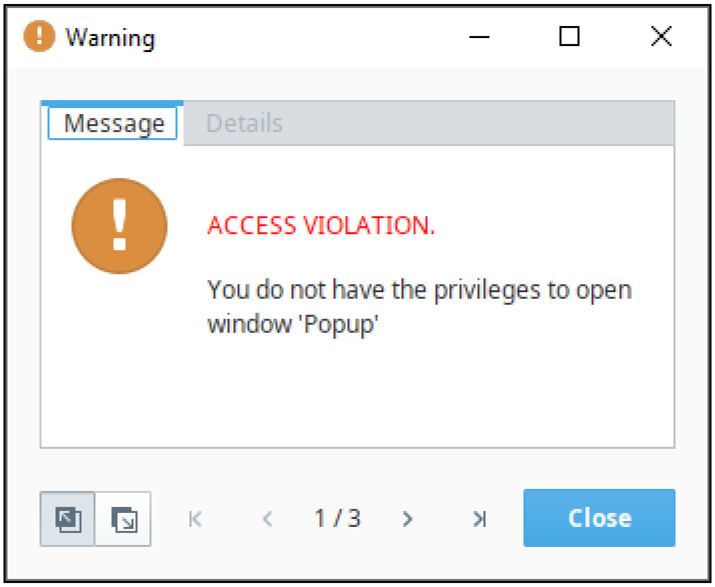
   Access Violation - Key

   ```
   <HTML><P><font color='red'>ACCESS VIOLATION.</font><P>You do not have the privileges to open
   window '%s'
   ```



<HTML><P><font color='red'>ACCESS VIOLATION.</font><P>You do not have the ...

3. Save your project.
4. When a user without the proper permission attempts to access the popup from the client, they will get the following message.



ACCESS VIOLATION.

You do not have the privileges to open window 'Popup'

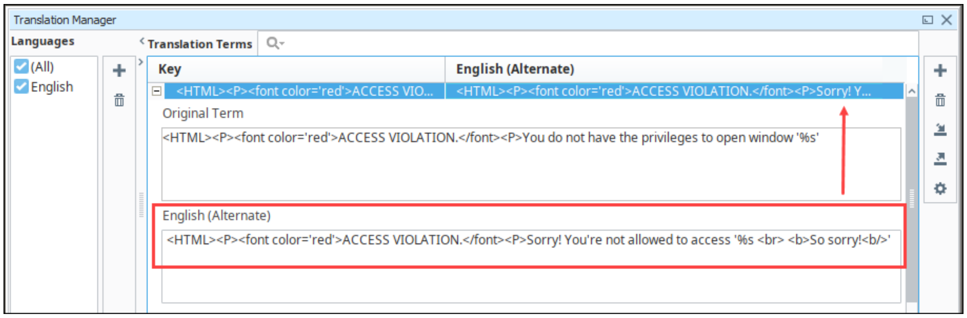
Using the same example from above, we modified a version of the text that appeared in the message box. We used the **English (Alternate)** as the language where the name of the window is passed into the message box via the '**%s**' substring.

1. Copy the new message from the code block below.

Access Violation - Translated Term

```
<HTML><P><font color='red'>ACCESS VIOLATION.</font><P>Sorry! You're not allowed to access '%s
<br> <b>So sorry!<b/>'
```

2. In the Transaction Manager window, double click the row for the **Key** and paste it in the **English (Alternate)** field. Click **Save**.



Key	English (Alternate)
<HTML><P><font color='red'>ACCESS VIO...	<HTML><P><font color='red'>ACCESS VIOLATION.</font><P>Sorry! Y...

Original Term

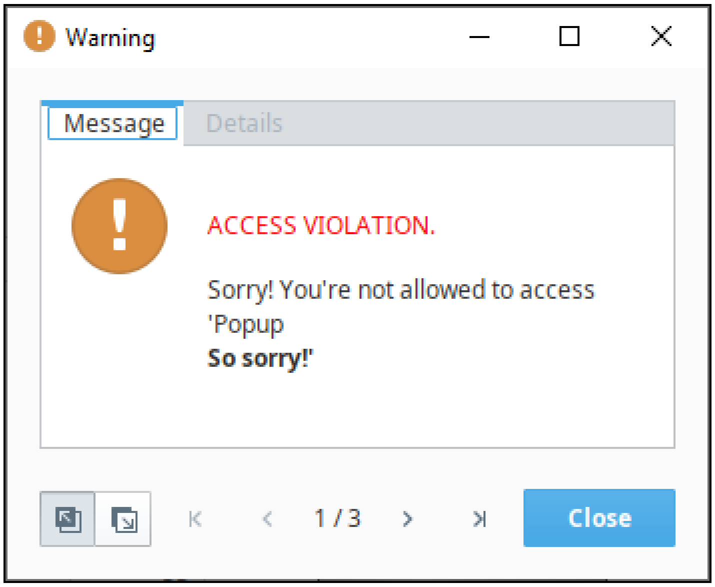
<HTML><P><font color='red'>ACCESS VIOLATION.</font><P>You do not have the privileges to open window '%s'

English (Alternate)

<HTML><P><font color='red'>ACCESS VIOLATION.</font><P>Sorry! You're not allowed to access '%s <br> <b>So sorry!<b/>'

3. Save your project.

Now, our message box appears as the following when a user does not have the proper access.



ACCESS VIOLATION.

Sorry! You're not allowed to access 'Popup

So sorry!'

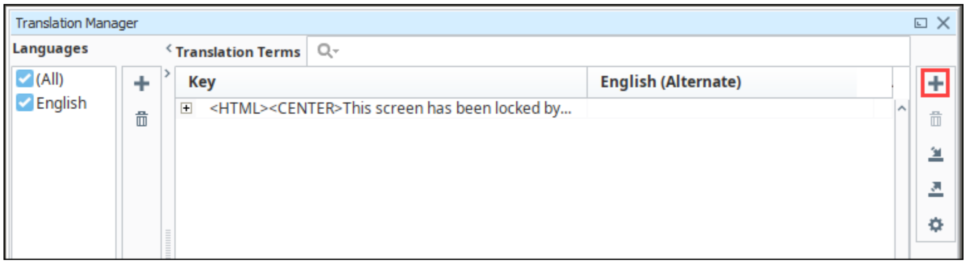
Screen Locked Message

The built-in system.security.lockScreen function in a client timer script can be used to lock the screen. In some cases, you may wish to change the text on the Screen Locked window. Use the following steps to change the text.

1. Open the Transaction Manger window.
2. Copy the **Key** from the code block below. Click the plus ✚ icon and paste it in the **Add Item** field. Click **OK** and you'll notice the code will be displayed in the **Key** field.

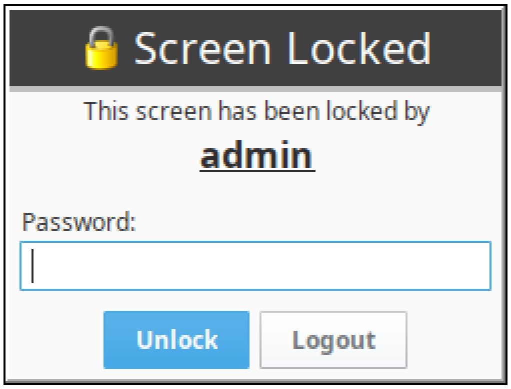
Screen Locked - Key

```
<HTML><CENTER>This screen has been locked by<BR><font size=+1><b><u>%s</u></b>
```



<HTML><CENTER>This screen has been locked by...

3. **Save** your project. Now the Screen Locked window will appear as follows.



This screen has been locked by
admin

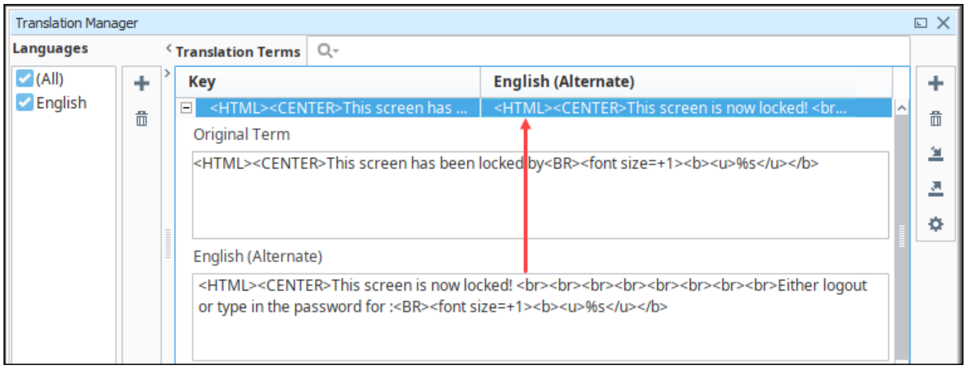
Building on the previous example, we can use the following key to translate the message on the Screen Locked window. The '**%s**' substring will reference the username of the user currently logged in. In this example, we added a few line breaks, which will make the Screen Locked window appear taller.

1. Copy the updated text from the code block below:

Screen Locked - Translated Term

```
<HTML><CENTER>This screen is now locked! <br><br><br><br><br><br><br><br>Either logout or type in
the password for :<BR><font size=+1><b><u>%s</u></b>
```

2. In the Transaction Manager window, click the row for the **Key** and paste it in the **English (Alternate)** field. Click **Save**.



<HTML><CENTER>This screen has ...

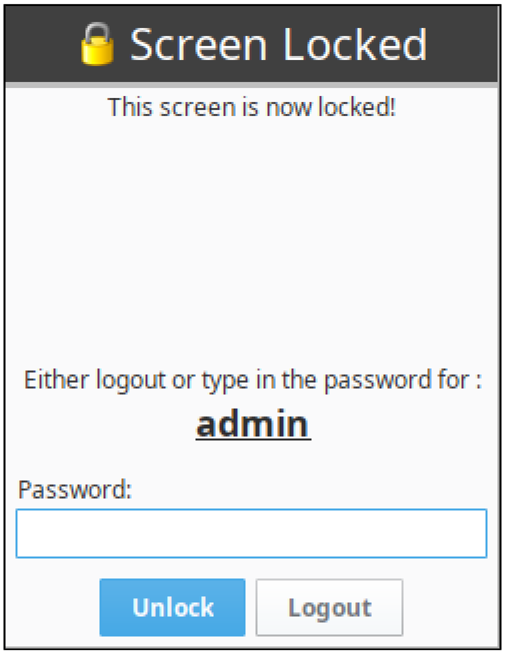
<HTML><CENTER>This screen is now locked! <br...

<HTML><CENTER>This screen has been locked by<BR><font size=+1><b><u>%s</u></b>

<HTML><CENTER>This screen is now locked! <br>

<br>Either logout or type in the password for :<BR><font size=+1><b><u>%s</u></b>

3. **Save** your project. Now, the message on the Screen Locked window appears as follows.



Screen Locked

This screen is now locked!

Either logout or type in the password for :

admin

Expression Overview and Syntax

Overview

The expression language is used to define dynamic values for component properties and expression Tags. Expressions often involve one or more other values that are used to calculate a final value. In most cases, expressions only return a value.

The classic example for an expression is to change a temperature that is stored in Celsius to Fahrenheit in order to display it. Suppose you had a Tag,Tank 6/Temp, that was in Celsius. If you wanted to display that Tag in Fahrenheit on a Label, you would use an Expression Binding on the label's text property using the following expression:

```
1.8 * {Tank 6/Temp} + 32
```

Every time that the temperature Tag changes, the expression will re-calculate the value and push it into the Label's text property. Now lets say that you wanted to append a "°F" to the end of the label so that the user knew the units of the temperature. You could simply use some string concatenation in your expression, like this:

```
(1.8 * {Tank 6/Temp} + 32) + " °F"
```

Lets suppose that you wanted to give the user an option to display the value in Celsius or Fahrenheit, based on checking a checkbox. You could add a Check Box component to the screen called DisplayFahrenheit. Then you could use this expression to dynamically display either unit, based upon the user's selection:

```
if({Root Container.DisplayFahrenheit.selected}, (1.8 * {Tank 6/Temp}
+ 32) + " °F", {Tankf/Temp} +  " °C")
```

On this page

...

Syntax

As its name suggests, everything in the expression language is an "expression". This means that everything returns a value. 5 is an expression. So is 5+1. So are {MyTags/TankLevel} and {MyTags/TankLevel}+1. Expressions can be combined in many powerful ways. Lets take a look at how expressions are written.

More formally, an expression is any one of the following:

- Number
- Boolean
- String
- Bound Tag
- Bound property
- Function call
- Dataset access
- Equation involving any of these

Literal Values

Literal values are things like numbers, booleans, and strings that are represented directly in the language. In the expression language, numbers can by typed in directly as integers, floating point values, or using hexadecimal notation with a 0x prefix. Examples:

```
42
8.456
0xFFC2
```

Strings are represented by surrounding them with double or single quotes. You can use the backslash character to escape quotes that you want to be included in the string. Examples:

```
"This is a regular string"
'This one uses single quotes'
"This string uses \"escaping\" to include quotes inside the string"
```

In addition, the following escape characters are available:

Character	Description
`\n`	New line
`\r`	Tab
`\t`	Carriage return

```
// The words "Hello" and "User" will be placed on separate lines
"Hello\nUser"

// Each "\t" will inject a tab
"Lots\tOf\tSpace"
```

Operators

You can use these arithmetic, logical, and bit-shifting operators to combine expressions.

Operator	Name	Description
//	Comments	Allows for comments following this operator.

-	Unary Minus or Subtraction	If both preceeded by a number, then returns a value by subtracting the operand right from the operand on the left of the operator:
		If preceded by anything else (or nothing, such as the start of the expression) and is followed by a number, this operator will negate the number.
		`// This will return 6`
		`10 - 4`
		
		`// This will return -10`
		`- (15 - 5)`

!	Not	Logical opposite of a boolean.
^	Power	Raises a number to the power of another number.
%	Modulus	Modulus or remainder of two numbers. a%b is the remainder of a÷b.
*	Multiply	Multiplies the number on the left of the operator by the number on the right of the operator.
/	Divide	Divides the number on the left of the operator by the number on the right of the operator.

+	Add or Concatenation	If both operands are numbers, this will add them together.
		Otherwise treats arguments as strings and performs concatenation.
		`// This will return 10`
		`4 + 6`
		
		`// This will return 'FirstSecond'`
		`'First' + 'Second'`
		
		`// This will return '2Alarms'`
		`2 + 'Alarms'`

-	Subtraction	Subtracts the number on the right of the operator from the number on the left of the operator.
~	Bitwise NOT	Examples the bits of an operand, and performs performs logical negation: bits with a value of 0 become 1, and vice versa.

&	Bitwise AND	Examines the bits of two operands, and performs a logical AND to each set, comparing the bits in each position from both sets. Returns true for any position where the bits being compared are true.
		`//                  0101`
		`//         AND 0011`
		`//         =        0001`
		
		`// Performs the bitwise operation above, resulting in the decimal 1`
		`5 & 3`

|	Bitwise OR	Examines the bits of two operands, and performs a logical OR to each set, comparing the bits in each position from both sets. Returns true for any position where either bit being compared is true.
		`//                  0101`
		`//         OR        0011`
		`//         =        0111`
		
		`// Performs the bitwise operation above, resulting in the decimal 7`
		`5 | 3`

xor	Bitwise XOR	Examines the bits of two operands, and performs a logical exclusive OR to each set, comparing the bits in each position from both sets. Returns true for any position where only one of the bits are true.
		`//                  0101`
		`//         XOR        0011`
		`//         =        0110`
		
		`// Performs the bitwise operation above, resulting in the decimal 6`
		`5 xor 3`

<<	Left Shift	A signed bitwise left shift.
>>	Right Shift	A signed bitwise right shift.
>	Greater Than	Logical greater-than test between two numbers. Returns a boolean.
<	Less Than	Logical less-than test between two numbers. Returns a boolean.
>=	Greater Than or Equal To	Tests if the operand on the left is greater or equal to the operand on the right. Returns a boolean.
<=	Less Than or Equal To	Tests if the operand on the left is less than or equal to the operand on the right. Returns a boolean.
=	Equal	Tests for equality between two operands.
!=	Not Equal	Tests for equality, returning true when not equal.
&&	Logical AND	Returns true when both operands are true. Anything non-zero is considered true.
||	Logical OR	Returns true when either operand is true. Anything non-zero is considered true.
like	Fuzzy String Matching	Compares the left-hand value with the pattern on the right side. The pattern may consist of %,*, and ? wildcards.

Bound Values

Bound values are paths to other values enclosed in braces. These will appear red in the expression editor. When you are writing an expression for a Expression Binding in Vision, you can reference Tag values and property values using the brace notation. When you are writing an expression for an Expression Tag, you can only reference other Tag values. You can use the **Insert Property** icon and **Insert Tag** icon to build these references for you.

Dataset Access

If you have an expression that returns a dataset, you can pull a value out of the datatset using the dataset access notation, which takes one of these forms:

```
Dataset_Expression ["Column_Name"]    //returns the value from the first row at the given column name
Dataset_Expression [Column_Index] //returns the value from the given column at the first row
Dataset_Expression [Row_Index, "Column_Name"] //returns the value from the given row at the given column
name
Dataset_Expression [Row_Index, Column_Index] //returns the value from the given row at the given column
index
```

For example, this expression would pull a value out of a Table at row 6 for column "ProductCode":

```
{Root Container.Table.data}[6, "ProductCode"]
```

Note that you'll often have to convince the expression system that what you're doing is safe. The expression language can't tell what the datatype will be for a given column, so you may have to use a type-casting function to convince the expression language to accept your expression, like this:

```
toInt({Root Container.Table.data}[6, "ProductCode"])
```

Expression Functions

The expression language's functions are where much of the real power lies. A function may take various arguments, all of which can themselves be any arbitrary expression. This means that you can use the results of one function as the argument to another function. In general, the syntax for a function call is:

```
functionName(expression1, expression2, ...)
```

Whitespace and Comments

Whitespace, such as spaces, tabs and newlines, are largely ignored in the expression language. It is often helpful to break your expression up onto multiple lines for clarity. Comments are delimited by two forward slashes. This will make the rest of that line be ignored. This example shows an if function spread over 4 lines with comments annotating the arguments.

```
if( {Root Container.UseTagValueOption.selected},
    {MyTags/SomeValue}, // Use the tag value
    "Not Selected"     // Use default value if the user doesn't check the box
)
```

Tag Paths

While referencing a tag path in expressions, you may see some special notation, such as "~" and "[.]". More information on this notation can be found on the Understanding Tags page.

Additional Examples

The following headings demonstrate many simple examples that better demonstrate the Expression Language's syntax.

String Concatenation

You can use an expression binding to concatenate strings resulting in a new string that reflects the concatenation of different strings.

The function starts with `CONCAT()` and inside the function is a list of strings. They can be manually typed like `"42"` or they can come from Tags or properties.

Expression Binding – Concat Strings

Watch the Video

Example 1

```
concat("The answer is: ", "42") //returns "The answer is: 42"
```

Example 2

You have a Date, and need to extract the year, and concatenate the word "Vintage" to the end for a label display. Bind a label's text property to:

```
 dateExtract({Root Container.VintageDate}, 'year') + ' Vintage'
```

Celsius to Fahrenheit

Use an expression binding to convert a temperature from Celsius to Fahrenheit. This is an example of how an expression binding can handle calculations.

```
{celsiusTemp} * 9/5 + 32
```

The reference to the Celsius temperature can come from a property or a Tag. As the property or the Tag changes, so does the expression binding.

Expression Binding - Celsius to Farenheit

Watch the Video

Format Date

You can format a date in an expression binding by using the `dateFormat` and `now` functions.

To make a label that updates to show the current time:

1. Drag a **Label** component onto the window.
2. Select the label's **Text** property binding icon and select **Expression** binding.
3. Enter the following code into the expression, and click **OK**.

   ```
   dateFormat(now(1000), "MMM d, yyyy hh:mm:s a")
   ```

   The `dateFormat` function takes two arguments. The first argument is any date type variable. This can include another function (like now) that returns a date type. The second argument refers to the date format that you want returned. The `now` function returns the current time and in this case it will update every second.
   For more information on the date formatting expression, see the Appendix.

Expression Binding – Format Date

Watch the Video

Date Manipulations

You can manipulate dates in expression bindings such as a date addition/subtraction with the `date Arithmetic` function. This is important when you want to use the expression bindings to select a date that is offset by a certain amount.

The following example will return the time 15 minutes ago by using the dateArithmetic expression function:

```
dateArithmetic(now(), -15, "minute")
```

Expression Binding – Date Manipulations

Watch the Video

Bit Functions

You can use various bit functions in expression bindings like `getBit` to return individual bits of a word.

Example 1

Assuming a tag path 'Folder/BitTag', the following would return the binary representation of the Tag's value at the 0 position

```
getBit({Folder/BitTag}, 0)
```

For more details on the getBit function, please see the getBit page in the Appendix.

Expression Binding - Bit Functions

Watch the Video

Example 2

You have 3 bits in a PLC, only one of which will be on at a time. You want to turn these 3 bits into a single integer (0,1,2) to drive a component's Styles. Bind a custom integer property to:

```
binEnum({MyTags/Bit1}, {MyTags/Bit2}, {MyTags/Bit3})
```

Switch

You can use the switch function in expression bindings to evaluate conditional statements. This function acts like the switch statement in C-like programming languages. It takes the value argument and compares it to each of the case1 through caseN expressions.

The following example returns the string `"Running"` when it is given the value of `1`. Its options are `0`,`1`, and `2`. And when comparing the value to the options the switch statement returns one of the corresponding results. If a result cannot be found, a fail-over option is returned.

```
switch(
1,
0, 1, 2,
"Off","Running","Fault",
forceQuality("!BAD STATE!",0))
```

Expression Binding – Switch

Watch the Video

Checking Conditions

You can use expression bindings to return true or false based on different conditions.

Example 1

Consider the following expression that references the Tag with a path of 'Folder/Machine State':

```
{Folder/Machine State} = 0
```

The above expression simply tests the value of the Machine State Tag. If the value of the Machine State Tag is ever equal to 0 then the above expression would return true. In every case where Machine State is not equal to 0, then the expression would return false.

Example 2

It is possible to check for multiple conditions in the same expression. If you have two boolean Tags and you only want the expression to return true if both Tags are true then the binding would look like:

```
{boolTag1}=True && {boolTag2}=True
```

Example 3

Consider the following expression that references the Tag with a path of 'Folder/Machine State':

```
if({Folder/Machine State} = 0, 1, 0)
```

This also finds the opposite of the Machine State if it's a boolean. It returns whether or not the Machine State is 0.

Expression Binding - Checking Conditions

Watch the Video

SQL in Ignition

On this page

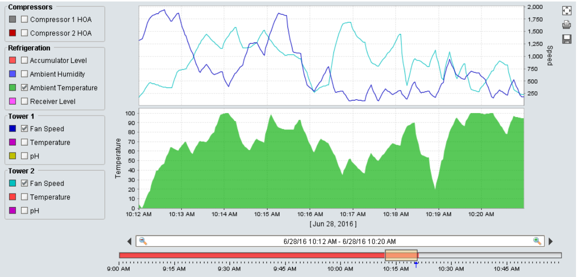
...

Leveraging Databases

In addition to all the normal HMI functionality you would expect, Ignition has the ability to connect to databases, and this can greatly increase the functionality available to you! You can use databases to store history, create easy to search lists and configurations, and retrieve data from ERP or other systems. When you start using SQL, you can expand your project from a simple HMI to a project that brings your whole process together. The best part is that Ignition connects to as many databases as you want, all from one central location. Because the database lives outside of Ignition, we don't maintain any control or rules over what you can do with your data.

Displaying Data

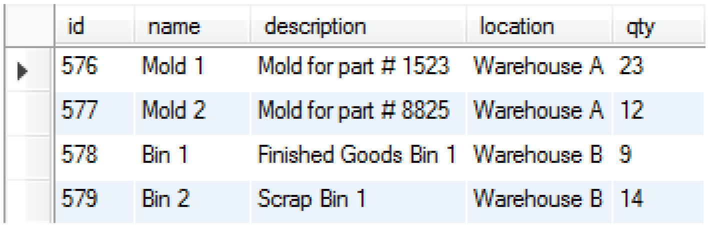
You can easily display information from your databases on a window along with anything else in Ignition. You can show parts lists, step sequences, realtime or historical charts, add the ability to search for inventory, or anything else you can think of.



What is SQL?

SQL stands for "Structured Query Language" and is the backbone of most modern relational databases. It's often referred to as "S.Q.L." or "Sequel," but both are correct and widely recognized. This language (different than the Python Scripting Language) allows you to write out requests or "queries" against the existing data to view, add, edit, or remove the information you want in a simple format.

Everything in a relational database is based around tables. Tables store the basic information for any system and can be combined together to make very efficient queries to retrieve your data.



id	name	description	location	qty
576	Mold 1	Mold for part # 1523	Warehouse A	23
577	Mold 2	Mold for part # 8825	Warehouse A	12
578	Bin 1	Finished Goods Bin 1	Warehouse B	9
579	Bin 2	Scrap Bin 1	Warehouse B	14

```
SELECT *
FROM inventory
JOIN location on location.name = inventory.location;
```

SQL Queries

SQL queries are crucial to Ignition's database-centric model. Queries can show what is available or alter data in the databases, and some companies have positions just dedicated to running databases and creating queries. Anywhere Ignition is fetching data, you can choose to use your own custom queries to get exactly what you want out of the database. You can make your queries as simple or complex as you like. If your database is large, you might have a whole team dedicated to creating these queries for you and Ignition will happily execute them.

SQL - Select Statement - Select Data from a Table

```
SELECT * FROM mytable
```

SQL - Inner Join Statement - Selects Records that have Matching Values

```
SELECT users.id, users.firstname, users.lastname, roles.name as 'rolename'
FROM users
INNER JOIN mapping ON users.id = mapping.userid
INNER JOIN roles ON mapping.roleid = roles.id
WHERE roles.name = 'Administrator'
```

Database Connections

Any SQL query you use needs a Database connection, but Ignition simplifies all that by creating database connections in the Gateway instead of in the clients. This means from one central location you can manage all your database connections, and you don't have to worry about planning around adding clients in the future. Any special rules or connection restrictions are taken care of in the Gateway.

Using SQL in Ignition

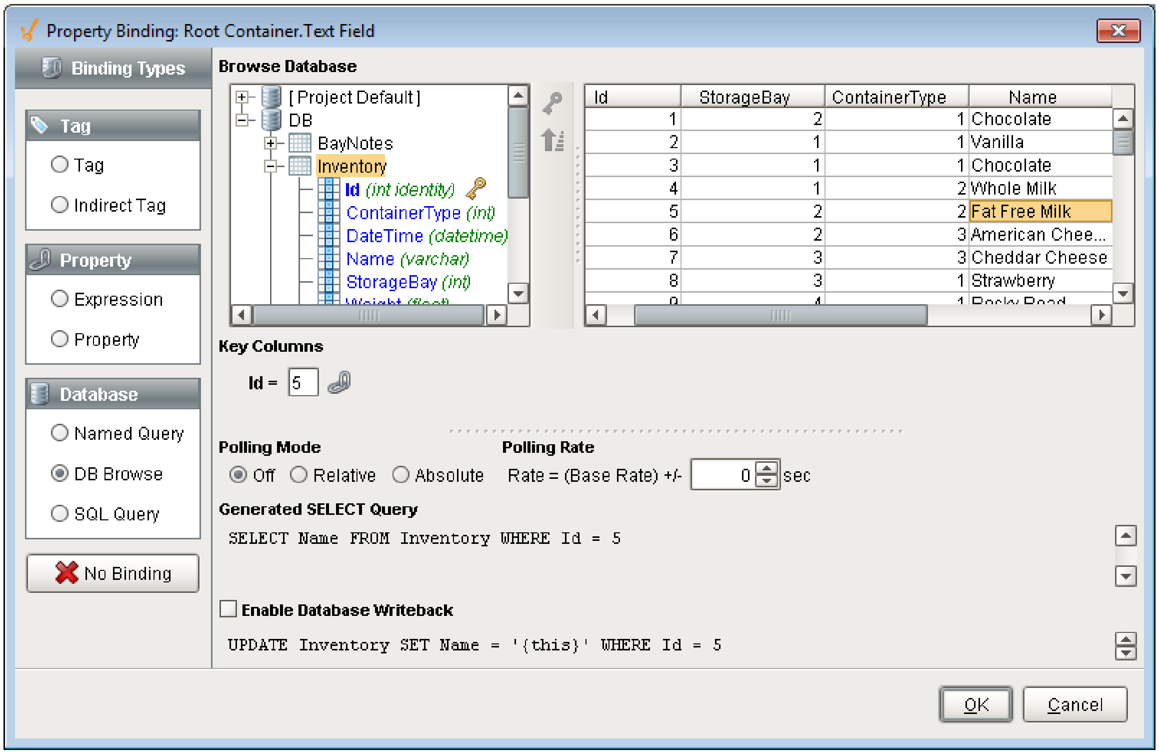
There are many types of queries, and many ways to use them in Ignition. Some provide an easy to use builder to automatically store or fetch data, and some allow you to completely customize your queries.

Querying Data from Database

Watch the Video

Queries in Bindings

Ignition's binding system offers a lot of flexibility in how database data can be used in a binding. The **Named Query** binding allows you to select one of the Named Queries that were previously built for that project, offering a very secure method of pulling data from the database. The **DB Browse** builder provides an interface that will build the query based on the data in the table that was selected. This allows even users with little SQL knowledge to pull data from the database. Finally, the **SQL Query** option will accept a straight query, so that a query specific to that binding can be written directly in the binding. When binding to a basic (non-dataset) data type, you can use the Writeback feature directly to send any changes back to the database.



Id	StorageBay	ContainerType	Name
1	2	1	Chocolate
2	1	1	Vanilla
3	1	1	Chocolate
4	1	2	Whole Milk
5	2	2	Fat Free Milk
6	2	3	American Chee...
7	3	3	Cheddar Cheese
8	3	1	Strawberry
9	4	1	Rocky Road

Key Columns

Id = 5

Polling Mode

● Off ○ Relative ○ Absolute

Polling Rate

Rate = (Base Rate) +/- 0 sec

Generated SELECT Query

```
SELECT Name FROM Inventory WHERE Id = 5
```

□ Enable Database Writeback

```
UPDATE Inventory SET Name = '{this}' WHERE Id = 5
```

Queries in Scripting

Ignition offers a number of built-in scripting functions for you to use to query your databases. This makes it very simple to view data, create dynamic scripts that use real data, and more. You can pull individual pieces of information, return whole tables of data, or update your database directly. Depending on the type of query and the sort of results you want, you will use different functions. The following functions are the ones you will use most, and all of them can use a special placeholder (?) to allow for dynamic query building.

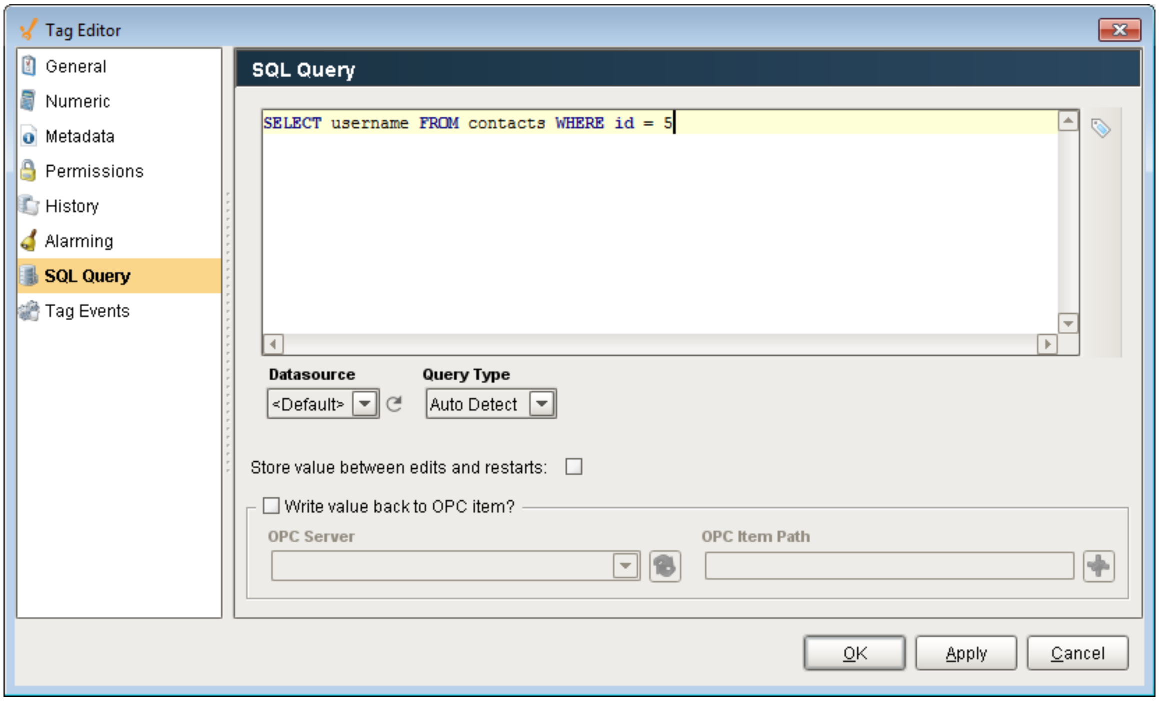
Scripting Function	Description
system.db.runNamedQuery()	Used to run a previously setup Named Query from within a script.
system.db.runPrepQuery()	Used to run basic SELECT queries to fetch whole datasets. This can be used to populate tables, or to sift through the data to do your own calculations.
system.db.runPrepUpdate()	Used to run queries that change the data in the database. Usually used on input form windows to update your database records.
system.db.runScalarPrepQuery()	Used when you want only one value from your results. Perfect for fetching a single value like the highest ID, or first timestamp of a result set.

Each of the different functions takes in different arguments (values) and provides slightly different options and functionality. For example, the runPrepUpdate() can return the auto-generated key from insert queries. This can be extremely helpful and eliminate the need to hit the database multiple times if you are using linked tables.

You can find examples of each of these and all the other database functions in the system.db section of the appendix.

Queries in Tags

Ignition offers Query Tags, which can run queries and return the result as a Tag value, giving all of the projects in the Gateway access to the same Database values.



```
SELECT username FROM contacts WHERE id = 5
```

Datasource: <Default>

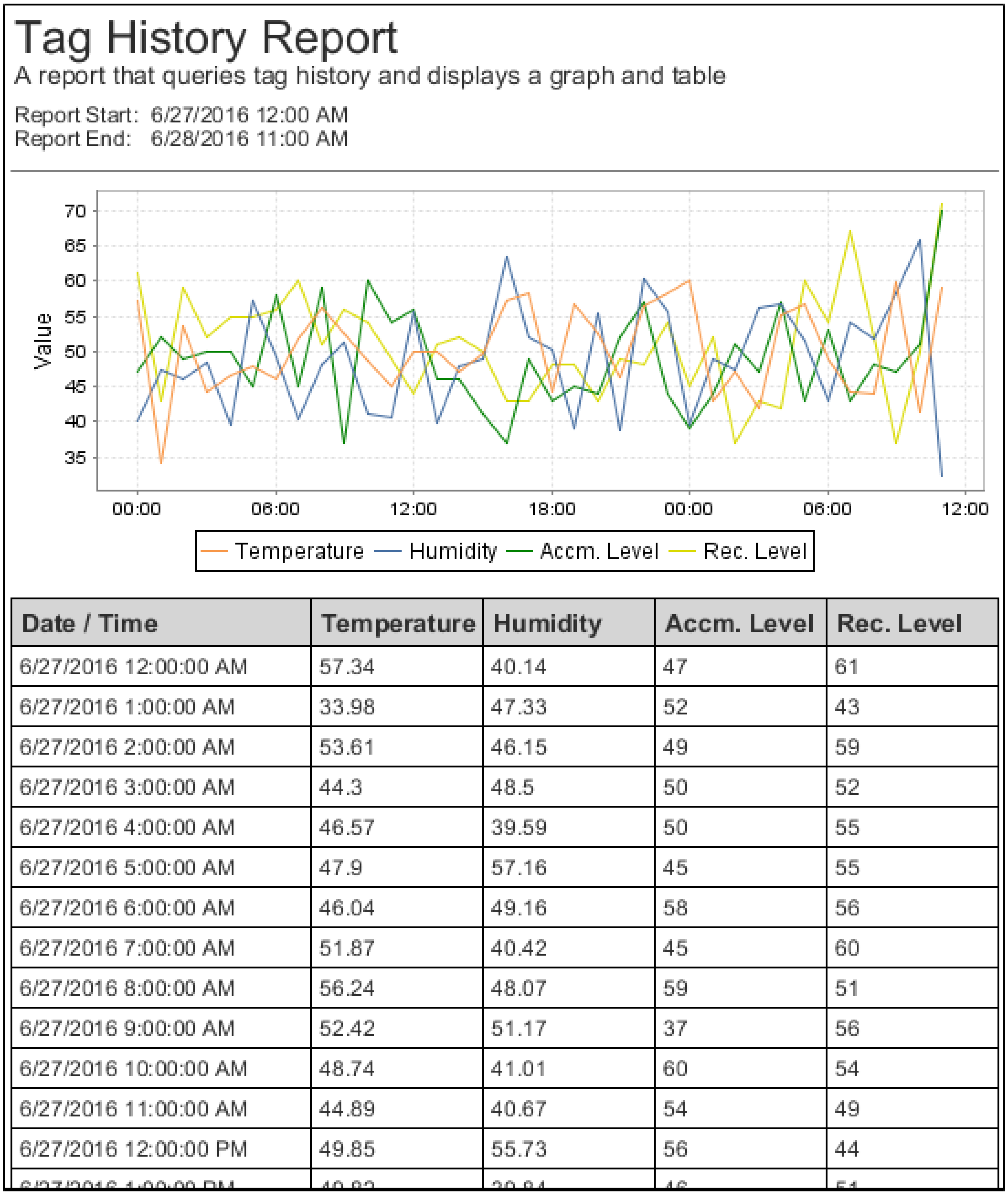
Query Type: Auto Detect

Store value between edits and restarts: □

□ Write value back to OPC item?

Queries in Reports

You can leverage queries to access data from all database connections to create rich reports, from simple data logging to complex reports with grouped charts and datasets.



Tag History Report

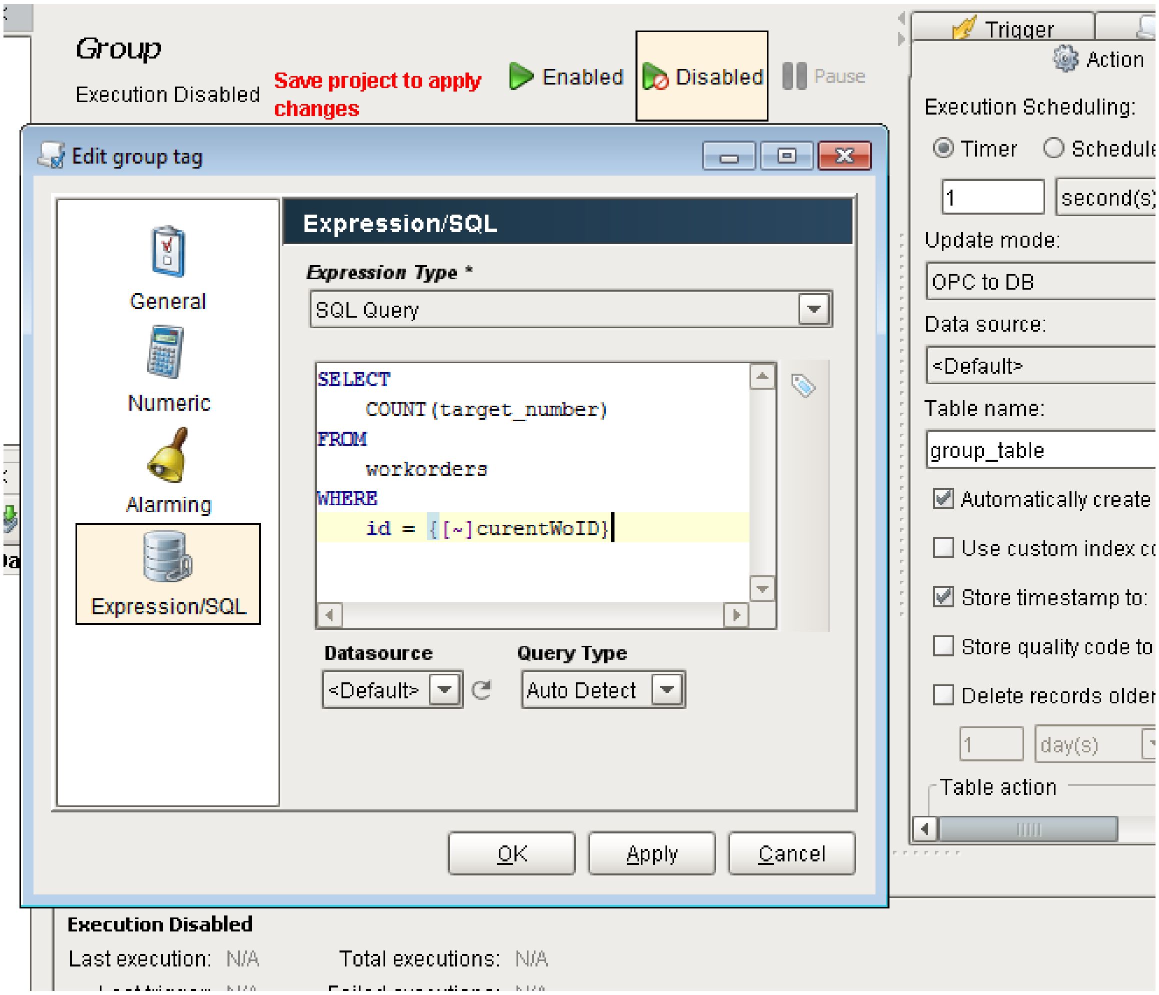
A report that queries tag history and displays a graph and table

Report Start: 6/27/2016 12:00 AM
Report End: 6/28/2016 11:00 AM

Date / Time	Temperature	Humidity	Accm. Level	Rec. Level
6/27/2016 12:00:00 AM	57.34	40.14	47	61
6/27/2016 1:00:00 AM	33.98	47.33	52	43
6/27/2016 2:00:00 AM	53.61	46.15	49	59
6/27/2016 3:00:00 AM	44.3	48.5	50	52
6/27/2016 4:00:00 AM	46.57	39.59	50	55
6/27/2016 5:00:00 AM	47.9	57.16	45	55
6/27/2016 6:00:00 AM	46.04	49.16	58	56
6/27/2016 7:00:00 AM	51.87	40.42	45	60
6/27/2016 8:00:00 AM	56.24	48.07	59	51
6/27/2016 9:00:00 AM	52.42	51.17	37	56
6/27/2016 10:00:00 AM	48.74	41.01	60	54
6/27/2016 11:00:00 AM	44.89	40.67	54	49
6/27/2016 12:00:00 PM	49.85	55.73	56	44
6/27/2016 1:00:00 PM	49.82	39.84	46	51

Queries in Transaction Groups

While Transaction Groups are great at storing Tag data to a Database automatically, the built-in Expression Items can execute a SQL Query within the Transaction Group.



Expression/SQL

Expression Type *: SQL Query

```
SELECT
    COUNT(target_number)
FROM
    workorders
WHERE
    id = {[~]curentWoID}
```

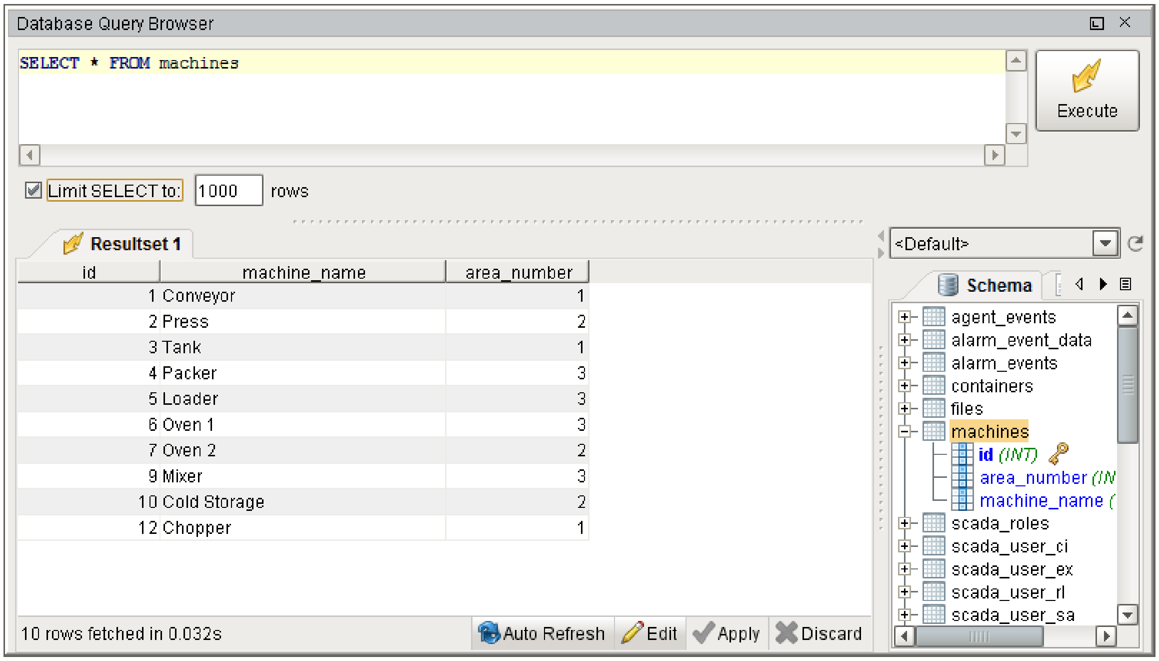
Datasource: <Default> Query Type: Auto Detect

Execution Scheduling: ● Timer ○ Schedule
1 second(s)
Update mode: OPC to DB
Data source: <Default>
Table name: group_table
☑ Automatically create
☐ Use custom index co
☑ Store timestamp to:
☐ Store quality code to
☐ Delete records older
1 day(s)
Table action

Execution Disabled
Last execution: N/A   Total executions: N/A
Last trigger: N/A   Failed executions: N/A

Database Query Browser

The Database Query Browser offers an easy to use environment to run queries in for testing. Here, queries can be tested to figure out what values get returned, or data can be updated through raw queries, or the Database Query Browsers easy to use GUI editor.



```
SELECT * FROM machines
```

☑ Limit SELECT to: 1000 rows

id	machine_name	area_number
1	Conveyor	1
2	Press	2
3	Tank	1
4	Packer	3
5	Loader	3
6	Oven 1	3
7	Oven 2	2
9	Mixer	3
10	Cold Storage	2
12	Chopper	1

10 rows fetched in 0.032s

<Default>

- agent_events
- alarm_event_data
- alarm_events
- containers
- files
- machines
  - id (INT)
  - area_number (IN
  - machine_name (
- scada_roles
- scada_user_ci
- scada_user_ex
- scada_user_rl
- scada_user_sa

Auto Generated Queries

Many systems within Ignition utilize a database connection, but the queries that are executed are constructed automatically by the system and do not require you to build the queries manually. These systems such as the Tag Historian, the Alarm Journal, or the Database User Source are very easy to setup and use since each system will automatically generate the necessary tables in the database, insert the relevant data, and even has prebuilt tools to extract the data. However, it is important to note that while these systems can automatically generate queries for you using the various components, these systems are simply storing data in a database which you can manually query out by building your own SQL queries.

Related Topics ...

- Database Connections
- Connecting to Databases
- Scripting

In This Section ...

Writing SQL Queries

On this page

...

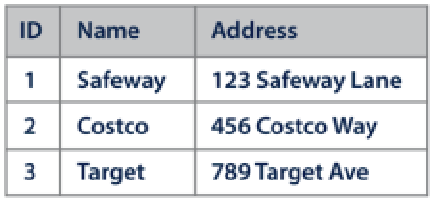
SQL Tables

For information about databases and how to get connected, see the Database Connections section.

The foundation of every database system is a table. Every database consists of one or more tables, which store the database's data/information. Each table is identified by a name (for example **`Customers`** or **`Orders`**`),` and consists of column definitions and rows of data.

The database table columns have their own unique names and have pre-defined data types. Table columns can have various attributes defining the column functionality (such as the primary key, index, default value, and so on).

While table columns describe the data types, the table rows contain the actual data for the columns.



ID	Name	Address
1	Safeway	123 Safeway Lane
2	Costco	456 Costco Way
3	Target	789 Target Ave

Current common databases are almost all Relational Databases. This means that their tables can relate to each other by including an ID in one table that matches the key of another. These are called foreign keys.

Primary Key

A primary key is a way to uniquely identify each row in a table. While it is possible to create a database table without a primary key, it is highly recommended to configure one for each table. A primary key is comprised of either a single column, or set of columns. When multiple columns are specified as a primary key, this is known as a **composite primary key**. No two distinct rows in a table can have the same value (or combination of values) in those columns.

While Primary Keys can be configured in several ways, they *typically* meet the following criteria:

- **Integer Datatype**: The datatype of the key column is typically an integer, and not a varchar. The primary key is only an identifier to a specific row in a table, so an integer datatype can easily be used. Some databases support a UID or a UUID (Universally Unique IDentifier) that looks like a character string, but is something specially made for primary keys.
- **Automatically Incrementing**: The value of the primary key increments as rows are added. The key is usually configured to automatically increment in the database so that external applications (such as Ignition) don't have to figure out the next available value when inserting a new row.
- **Statically Defined**: Any row that is inserted must fill in these value(s) without creating duplicates. Configuring the primary key as **automatically incrementing** means that the database will automatically handle this criteria.
- **Non-NULL**: NULL (empty) values should not be present in the primary key. This column (or columns) will usually not allow NULL values.

Index

Indexes speed up the querying process by providing swift access to rows in the data tables, similarly to the way a book's index helps you find information quickly within that book. Indexes are extremely important when querying large sets of data. You should create an index for the set of columns you use commonly in a `WHERE` clause. For example, you should add an index on the timestamp column of a historical table when querying the table by a start and end date. Ignition does this automatically when it creates tables for Tag History or Transaction Groups.

Foreign Key

A Foreign Key is a referential constraint between two tables. The foreign key identifies a column or a set of columns in one (referencing) table that refers to a column or set of columns in another (referenced) table. The columns in the referencing table must be the primary key in the referenced table. For example, you might have a list of suppliers with an integer ID column. Then your invoices would use the supplier's ID instead of the name. These linked tables can save a lot of space because you don't have to include everything about the supplier in each invoice.

Example

Supplier (**SupplierNumber**, Name, Address, Type)
Invoices (InvoiceNumber, **SupplierNumber**, Text, Cost)

Null Value

NULL is a special marker used in SQL to indicate that a data value does not exist in the database. This way it is clear that there is no data, instead of guessing if a value of 0 is correct or just missing data. By definition, NULL is not equal to anything, even other NULL values. Every database has a function to identify if a value is NULL, usually called isNULL() or something similar.

Comments

Comments can be added to any SQL query just like with scripting. Single line comments are done with two dashes and a space: '-- '

SQL - Single Line Comment

```
-- This is a single line comment in a SQL query.
SELECT * FROM my_table
```

You can also do multi line comments by wrapping text within a forward slash and an asterisk: /* text */

SQL - Multi Line Comment

```
/* A multi line comment
can span multiple lines. The
comment will stop when it is closed
with another asterisk and forward slash.*/
```

SQL Queries

SQL queries (or statements) are used to create, maintain, and view relational databases like MySQL, SQLServer, Oracle, etc. They follow a specific format and use just key words to determine the structure of the query. Unlike most coding languages, SQL does not rely on newlines or start/end markers for it's format, each query is a single line of code. You will often see SQL queries split over several lines, but that is just to make them easier to read.

> You might notice a lot of CAPITALIZED words in SQL queries. While these key words are not case sensitive, it is still common practice for people to capitalize them in a query. Things like SELECT, FROM, WHERE (and a few others) are almost always capitalized because they have a special meaning. Try not to have table or column names that use any of these special words. You will see this capitalization format in examples throughout this user manual and other online references.

Select Command

The SELECT statement is used to select data from a database. The result is returned as a data set, called the result set. This is true even if there is only one value returned. The syntax for a SELECT statement is as follows:

SQL - The Select Command

```
SELECT myColumn FROM myTable
```

Select Examples

The "*" character can be used to specify all columns from a database table. While this is the easiest way to retrieve results from a table, this is not the recommended approach.

SQL - Selecting All Columns

```
SELECT * FROM Customers
```

The recommended approach is to instead specify only the columns that are required for a query. There are several reasons for this, but performance would be the main one: less columns in a statement means less work for the database, and the resulting data set in Ignition will use less memory.

SQL - Selecting a Single Column

```
SELECT Name FROM Customers
```

SQL - Selecting Multiple Columns

```
SELECT Name, Address FROM Customers
```

Where Clause

The WHERE clause is used in conjunction with other commands to extract only those records that fulfill a specified criterion. The WHERE clause usually goes near the end of the query followed by a condition that the values must meet to be returned.

Pseudocode - Where Clause

```
SELECT myColumn FROM myTable WHERE condition
```

The WHERE clause can use various operators for its condition, with the basic operators being:

Operator	Description
=	Equal to.
<>	Not equal to.
>	Greater than.
<	Less than.
>=	Greater than or equal to.
<=	Less than or equal to.

Where Examples

Only return customers from CA.

SQL - Simple Where Clause

```
SELECT * FROM Customers WHERE State = 'CA'
```

Only return users over a specified age.

SQL - Select Users over 25

```
SELECT * FROM Users WHERE Age > 25
```

Insert Into Command

The INSERT INTO statement is used to insert a new row in a table. If any columns have default values or are auto-incrementing, they can be omitted from the INSERT query.

Pseudocode - The Insert Command

```
INSERT INTO myTable (column1, column2)
VALUES ('Value1', 'Value2')
```

Insert Examples

SQL - Insert Values into Columns

```
INSERT INTO Customers (Name, Address, City, State, Zip, Country, Phone)
VALUES ('Inductive Automation', '90 Blue Ravine', 'Folsom', 'CA', '95630', United States, '1-800-266-7798')
```

If inserting a value into every column of the table, the columns do not need to be listed on the INSERT INTO statement. The values just need to be listed in the same order as the columns in the table. The table in the query below has four columns: id, first name, last name, and title.

SQL - Inserting to all Columns

```
INSERT INTO Users
VALUES (5628, 'Bob', 'Smith', 'Project Manager')
```

Update Command

The UPDATE statement is used to update existing records in a table. If a WHERE clause is not used, **all rows in the table will be updated**. As a result, the UPDATE statement should be used in conjunction with a WHERE clause in most cases. Many official management tools like SQLServer's Management studio will not allow UPDATE commands without a WHERE clause.

Pseudocode - The Update Command

```
UPDATE myTable SET myColumn = 'myValue'
```

Update Examples

SQL - Updating All Rows in a Single Column

```
UPDATE Customers SET Name = 'Inductive Automation'
```

SQL - Updating a Single Column for a Single Row

```
UPDATE Customers SET Address = '2110 21st Street' WHERE ID = 1
```

Delete Command

The DELETE statement is used to delete records in a table. **NEVER** run a delete command without a WHERE clause. It will delete **ALL** records from that table. Many official management tools like SQLServer's Management studio will not allow DELETE commands without a WHERE clause.

Pseudocode - The Delete Command

```
DELETE FROM myTable WHERE myColumn = value
```

Delete Examples

SQL - Deleting Based on Column Value

```
DELETE FROM Customers WHERE Name = 'Inductive Automation'
```

SQL - Deleting Rows Based on an ID Column

```
DELETE FROM Customers WHERE id < 538
```

Related Topics ...

- Inserting Data into a Database
- Named Queries
- Query Builder
- Database Query Browser

In This Section ...

SQL Select Options

While the SELECT command in its basic form can be very simple to use, the SELECT statement can be used with other statements or in certain ways that allow you to bring in exactly the data you need.

On this page ...

Selecting Static Values

Static values can be inserted into a resultset returned from a SELECT query as another column. Simply use the static value as a column to select, and the query will return a column where the name of the column is the static value, and every row in that column will return that same static value.

Pseudocode - Selecting Two Columns From a Table, and a Third Column with a Value of 10 for every row

```
SELECT column1, column2, 10 FROM table
```

Select Distinct Command

The SELECT DISTINCT statement works much like a SELECT statement works, in that it selects data from a database. However, SELECT DISTINCT will only return distinct or different values, not duplicates.

Pseudocode - The Select Distinct Command

```
SELECT DISTINCT column FROM table
```

Examples

This can be useful for getting a better idea of the range of values in a particular column.

SQL - Select Distinct Countries

```
SELECT DISTINCT country FROM Customers
```

Order By Clause

The ORDER BY keyword is used to sort the result-set by a specified column set of column. The ORDER BY keyword sorts the records in ascending (ASC) order by default. If you want to sort the records in a descending order, you can use the DESC keyword.

Pseudocode - Order By Clause

```
SELECT column1, column2 FROM table ORDER BY column2 DESC
```

Examples

SQL - Ordering by One Column

```
SELECT * FROM Customers ORDER BY Name ASC
```

You can use multiple columns to sort, this will sort by state first, and for each state the rows will be sorted by name.

SQL - Ordering by Multiple Columns

```
SELECT * FROM Customers ORDER BY State ASC, Name DESC
```

Limiting Rows Returned

SELECT commands can have the number of rows that the query returns limited using a special keyword. The keyword differs between database providers but the effect is the same, limiting the number of rows returned to a value that you specify.

Database	Keyword	Example

MS SQL Server/ MS Access	SELECT TOP value/percent	Note that the SELECT TOP command is unique in that you can also specify a percentage value instead of an exact number of records.
		`SELECT TOP 200 column1, column2 FROM table`
		`SELECT TOP 10 PERCENT column1, column2 FROM table`

MySQL	Limit value	`SELECT column1, column2 FROM table LIMIT 200`
Oracle DB	ROWNUM <= value	Note that the ROWNUM command is unique in that you can use it to identify the row number for any reason, not just limiting the number of rows returned. `SELECT column1, column2 FROM table WHERE ROWNUM <= 200`

Aliases

In a SQL query, aliases are used to give columns or even tables a temporary name for that query. Simply place the keyword AS after a column or table, followed by the alias name. If the alias is two words, it needs to be encapsulated in single quotes.

Pseudocode - Aliasing Columns and the Table

```
SELECT column1 AS a, column2 AS 'b c' FROM table AS t
```

Example

This can be really useful when the table has complex column names.

SQL - Aliasing Complex Column Names

```
SELECT id AS 'Badge Number', name AS 'Employee Name', dob AS Birthday FROM employees
```

This can also be useful when using multiple tables in a query, such as with a JOIN.

SQL - Joining Columns from Two Tables with Aliases

```
SELECT * FROM Contacts AS co JOIN Customers AS cu ON cu.ID = co.CustomerID
```

Union Command

The UNION operator is used to combine the results of two different SELECT statements. This differs from a JOIN in that there does not have to be a relationship between columns. However, both SELECT statements need to select the same number of columns with similar data types in a similar order. So if my first statement selects an int column and then a string column, the second statement needs to do the same. The name of the columns in the resultset will take the name of the columns from the first SELECT in the UNION.

Pseudocode - Union Two Tables

```
SELECT stringCol, intCol FROM table1
UNION
SELECT stringCol, intCol FROM table2
```

Examples

By default, the UNION operator will only select distinct values between the two tables.

SQL - Union of Users and Customers

```
SELECT username FROM users
UNION
SELECT name FROM customers
```

To select all values from both tables, we can use UNION ALL instead.

SQL - Union All

```
SELECT jobTitle FROM jobs
UNION ALL
SELECT position FROM employees
```

Static values can be used in a UNION to help differentiate the rows from each table.

SQL - Differentiating Between Users and Customers

```
SELECT 'User' AS Type, username FROM users
UNION
SELECT 'Customer', name FROM customers
```

Related Topics ...

- Writing SQL Queries
- Creating Tags

SQL Where Clauses

Overview

The SQL WHERE clause is utilized to restrict the number of rows impacted by a query. A WHERE clause is commonly utilized in two scenarios:

- In conjunction with a SELECT statement to filter the number of rows returned.
- As part of an UPDATE or DELETE statements to restrict which rows are manipulated by the query.

In either scenario, the syntax of a WHERE clause is used the same, and can have multiple predicates:

Pseudocode - Where Clause Syntax

```
SELECT * FROM table WHERE column = value
```

On this page ...

Where-Clauses and Data Manipulation

In most cases, statements that modify the content of a database table via UPDATE or DELETE **should** include a WHERE clause: otherwise the manipulation will be applied to every row. To demonstrate, you typically want to avoid queries that look like the following

Pseudocode - Never do this!

```
UPDATE table SET column = 'This was a horrible mistake'
```

When manually modifying a database table, a good habit to develop involves first writing a SELECT statement. If you can successfully write a WHERE clause that only returns the results you need to modify, then you can simply change the rest of your query to manipulate the table. Thus, we could write a query like the following:

Pseudocode - Select-Statement with a Where Clause

```
SELECT * FROM table WHERE id in (100,101,150,174)
```

If we receive only the results we need to modify in the query above, we can make a simple modification to our query to now delete just the rows we want.

Pseudocode - Delete-Statement with a Where Clause

```
DELETE FROM table WHERE id in (100,101,150,174)
```

Another common use of the WHERE clause is is to search through records and return the ones during a particular time frame. Timestamp columns can use the < and > operators to compare to each other, and certain string formats can be compared to timestamps like this query:

Pseudocode - Where Clause with a Timestamp

```
SELECT * FROM table WHERE t_stamp > '1984-01-25 16:35:55'
```

And Operator

The AND operator allows you to specify two or more conditions in a WHERE clause, where each condition must be true for the row to be returned.

Pseudocode - Using And to Specify Multiple Conditions

```
SELECT column1, column2, column3 FROM table WHERE column1 > value AND column2 < value AND column3 = value
```

Example

This helps to narrow down the result set even further by adding in additional conditions that must be met. This will only return rows for customers from Germany who are also over 20 years old.

SQL - Customers from Germany Over 20

```
SELECT * FROM customers WHERE country = 'Germany' AND age > 20
```

Or Operator

The OR operator allows you to specify two or more conditions in a WHERE clause, but only one of the conditions need to be true for the row to be returned.

Pseudocode - Using Or to Specify Multiple Conditions

```
SELECT column1, column2, column3 FROM table WHERE column1 > value OR column2 < value OR column3 = value
```

Example

The OR operator can help pull in data from two different subsets in the table. This will only return rows for customers from Germany or customers who are over 20 years old.

SQL - Customers from Germany or Customers Over 20

```
SELECT * FROM customers WHERE country = 'Germany' OR age > 20
```

Not Condition

The NOT condition allows you to specify a condition that must not be met for the row to be returned.

Pseudocode - Using Not to Specify a Condition that Shouldn't be met

```
SELECT column1, column2, column3 FROM table WHERE NOT column1 = value
```

Example

This can be useful for finding all data other than a certain subset. This will return all customers who are not from Germany.

SQL - Customers that are not from Germany

```
SELECT * FROM customers WHERE NOT country = 'Germany'
```

Between Operator

The BETWEEN condition allows you to specify a range of values separated by an AND that the value must be in for a condition to be true. The value can be numbers, text or dates and is inclusive of the first and last values in the range.

Pseudocode - Using Between to Specify a Range of Values

```
SELECT column1, column2, column3 FROM table WHERE column1 BETWEEN value1 AND value2
```

Examples

SQL - Customers that are Between the Ages of 20 and 40

```
SELECT * FROM customers WHERE age BETWEEN 20 AND 40
```

Note that the BETWEEN operator would work similarly to using a greater than or equal condition and a less than or equal condition.

SQL - Customers that are Between the Ages of 20 and 40 with no Between Operator

```
SELECT * FROM customers WHERE age >= 20 AND age <= 40
```

Timestamps can also use the BETWEEN operator to check for a given start time and end time.

SQL - Customers that are Between the Ages of 20 and 40 with no Between Operator

```
SELECT * FROM customers WHERE start_time BETWEEN '1984-01-25 00:00:00' AND '1984-01-25 16:35:55'
```

Like Condition

The LIKE condition allows you to specify a condition that must meet a certain pattern. Typically used to compare to string values, the pattern can be built using a combination of characters and the two wildcard values.

- % - Used to specify any number of any characters including zero characters.
- _ - Used to specify exactly one character.

Pattern Examples

Pattern	Meaning	Possible Matches
'%a%'	Values that have an 'a' in them.	'a', 'Inductive Automation', 'almost', 'create'
'_a_'	Values that have an 'a' with exactly one character before and after the 'a'.	'bat', 'cat', 'can'
'_a%'	Values that have an 'a' as the second character.	'da', 'saw', 'catcher'
'a%t'	Values that start with 'a' and end with 't'.	'about', 'at'
'%a%_%_%_'	Values that contain an 'a' with at least 3 other characters after it.	'trains', 'airplane', 'canteen'
'%a%a%'	Values that contain at least two 'a' characters in them.	'Inductive Automation', 'separate', 'apart'

Once the pattern has been constructed, it can be used with the LIKE operator to find values that match the specified pattern.

Pseudocode - Using Like to Specify a Pattern of Values

```
SELECT column1, column2, column3 FROM table WHERE column1 LIKE '%a%'
```

Example

The LIKE operator can be used to find all values that match a criteria, such as all countries with 'land' in that name

SQL - Customers that are not from Countries with 'land' in the Name

```
SELECT * FROM customers WHERE country LIKE '%land%'
```

In Condition

The IN operator allows you to specify a subset of values, with the condition that the return match at least one of them. Using an IN operator is similar to using multiple OR operators for the same column.

Pseudocode - Using IN to Specify Multiple Values

```
SELECT column1, column2, column3 FROM table WHERE column1 IN (value1, value2, value3)
```

Examples

The IN can be used as a shorthand way of writing out multiple conditions for the same column separated by OR operators. This would select all values where the country is either Germany, France, or USA.

SQL - Customers from Germany or France or USA

```
SELECT * FROM customers WHERE country IN ('Germany', 'France', 'USA')
```

This would be similar to doing something like this.

SQL - Customers from Germany or France or USA

```
SELECT * FROM customers WHERE country = 'Germany' OR country = 'France' OR country = 'USA'
```

The real power of the IN operator is that instead of specifying static values, an entirely new query can be run to compare values against.

SQL - Customers from Countries that users are also in

```
SELECT * FROM customers WHERE country IN (SELECT country FROM users)
```

Combining Multiple Operators

Multiple AND and OR operators can be combined to specify multiple different conditions that need to be met in order for a particular row to be returned. Additionally, each condition can be simple using the mathematical operators or complex using the conditions listed above. When using AND and OR operators in a WHERE clause, the AND will take precedence, evaluating first before the OR. In the pseudocode below, the row will be returned if either both the first and second conditions are met, or the third condition is met.

Pseudocode - Using Multiple Where Clause Operators

```
SELECT column1, column2, column3 FROM table WHERE column1 > value AND column2 < value OR column3 = value
```

However, the order at which the operators get evaluated can change by placing parentheses around the conditions which should be evaluated first. In the pseudocode below, the row will be returned if both the first condition is met, and either the second or third condition is met.

Pseudocode - Using Multiple Where Clause Operators with Parentheses

```
SELECT column1, column2, column3 FROM table WHERE column1 > value AND (column2 < value OR column3 =
value)
```

Examples

We can use complex conditions with different operators to find all customers who are over the age of 50 in a country that has 'land' in the name, or any customers in Germany or France.

SQL - Multiple Complex Conditions

```
SELECT * FROM customers WHERE country LIKE '%land%' AND age > 50 OR country IN ('Germany', 'France')
```

Using parentheses in the same query can drastically change what valid return conditions are. Here, the customer must both be from a country with 'land' in the name, as well as either over 50 or from Germany or France.

SQL - Multiple Complex Conditions

```
SELECT * FROM customers WHERE country LIKE '%land%' AND (age > 50 OR country IN ('Germany', 'France'))
```

Related Topics ...

- Writing SQL Queries
- system.db.runNamedQuery

SQL Table Joins

On this page

...

Overview

The SQL JOIN allows you to run a single SELECT statement that references multiple tables. This can be used for more advanced filtering, as well as combining data from multiple tables in a single result set. The process of joining two tables involves stating both tables in the query, and then specifying that a column from one table relates to another in some way.

Joins may look imposing at first, but they are simply SELECT statements that utilize columns from multiple tables.

The JOIN keyword works in conjunction with a SELECT statement. However, there are some key concepts that must be addressed when attempting to use the JOIN keyword.

Specifying Each Column

When listing column in a statement that uses the JOIN keyword, you must denote which table each column is being retrieved from. You can do this by using an Alias, or with the fully qualified column name. This prevents ambiguous columns in the context of the query, and makes it easier for you to use other keywords in the statement: i.e., adding WHERE clauses that apply to multiple tables.

Pseudocode - Fully Qualified Column Name

```
table_name.column_name
```

Declare the Relation

After the JOIN keyword, you must state which columns from each table relate to each other. This is accomplished by stating the name of the table, using the ON keyword, and then stating that a column on the first table is equal to a column on the second table. The columns specified are typically primary keys for their respective tables.

In the example below, we're stating that the values in **some_column** on **tableA** should be associated with matching values in **some_other_column** on **tableB**.

Pseudocode - Using the Join Keyword

```
FROM
        TableA
JOIN tableB ON tableA.some_column = tableB.some_other_column
```

Note that the order you present the columns **after** the ON clause does not matter: A = B is equivalent to saying B = A, so we could switch the columns listed with no distinguishable impact on the resulting query.

Combining the Concepts

Altogether, a basic JOIN looks like the following:

Pseudocode - Join Syntax

```
SELECT
        tableA.column,
        tableB.column
FROM
        tableA
JOIN tableB ON tableA.identity_column = tableB.identity_column
```

Joins with Three or More Tables

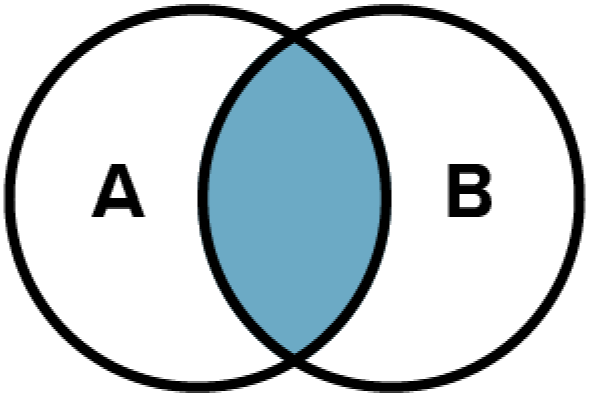
Joins can even be done between three or more tables. The syntax is similar, with each new table relation declared below the first.

Pseudocode - Join Syntax

```
SELECT
        tableA.column,
        tableB.column,
        tableC.column
FROM
        tableA
JOIN tableB ON tableA.identity_column = tableB.identity_column
JOIN tableC ON tableA.identity_column = tableC.identity_column
```

Join

The standard JOIN, also referred to as INNER JOIN, will only return rows where the joined columns contain matching values. If one of the joined columns contains a value that is not present in the in the other, then the row is not represented in the result set. You would use JOIN when you only want results that are represented in both tables.



This section will demonstrate the various uses of the JOIN keyword. For the sake of clarity, the queries will run against tables that look like the following:

Products Table	
id	**product_name**
1	Apples
2	Oranges
3	Grapes
4	Plums

Inventory Table			
id	**product_id**	**quantity**	**product_vendor**
1	1	15	Apple Corp
2	2	25	Orange Ya-Glad
3	3	56	Grape Escape
4	5	45	Banana Solutions

Join in Action

In this demonstration, only rows that pertain to Apples, Oranges, and Grapes are being returned. We're using a JOIN between **products.id** and **inventory.product_id**, so our results will only contain rows that have matching values from both of those columns. Our result set does not contain any information on products with **products.id** values of 4 or **inventory.product_id** values of 5, because those values are not present in **both** of the joined columns

SQL - Joining Products and Inventory

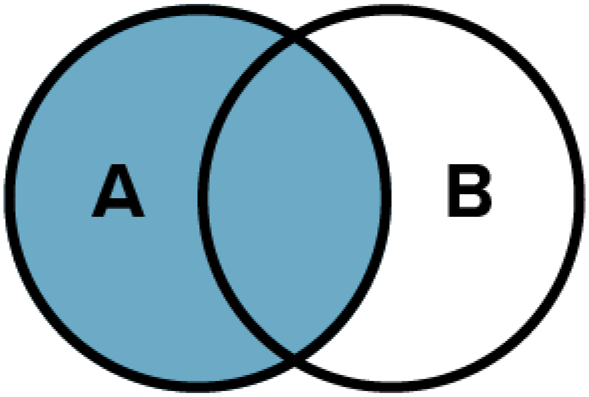
```
SELECT
        products.id
        ,inventory.product_id
        ,products.product_name
        ,inventory.product_vendor
        ,inventory.quantity
FROM
        products
JOIN inventory ON products.id = inventory.product_id
```

Example Results

id	product_id	product_name	product_vendor	quantity
1	1	Apples	Apple Corp	15
2	2	Orange	Orange Ya-Glad	25
3	3	Grapes	Grape Escape	56

Left Join

Return all rows from the left-most table (**table A** in the diagram), even if there are no matches on the right-most table (**table B**). If there isn't a matching record in the right table, then NULL values are returned.



Left Join in Action

Here we see all rows returned from our products table (since it is the left-most table in our query). In row 4, columns that are being populated via the inventory table (**product_id**, **product_vendor**, and **quantity**) contain NULL values, because there isn't a row on the inventory table that matches with a **product_id** value of 4. The query must return something in this case, so it returns NULL for these columns.

SQL - Left Joining Products and Inventory

```
SELECT
        products.id
        ,inventory.product_id
        ,products.product_name
        ,inventory.product_vendor
        ,inventory.quantity
FROM
        products
LEFT JOIN inventory ON products.id = inventory.product_id
```

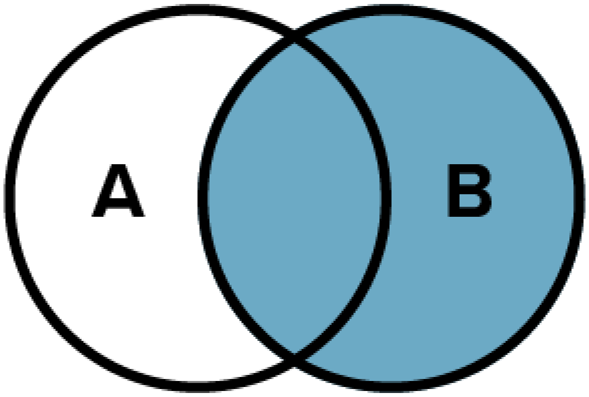
Example Results

id	product_id	product_name	product_vendor	quantity

1	1	Apples	Apple Corp	15
2	2	Orange	Orange Ya-Glad	25
3	3	Grapes	Grape Escape	56
4	NULL	Plums	NULL	NULL

Right Join

Return all rows from the right-most table (**table B**), even if there are no matches on the left-most table (**table A**). If there isn't a matching record on the left table, then NULL values are returned.



Right Join in Action

When using a RIGHT JOIN, all rows will be returned from the inventory table. The products table does not have a row that contains an **id** value of 5, so the **id** and **product_name** columns will show NULL values in our result set.

SQL - Right Joining Products and Inventory

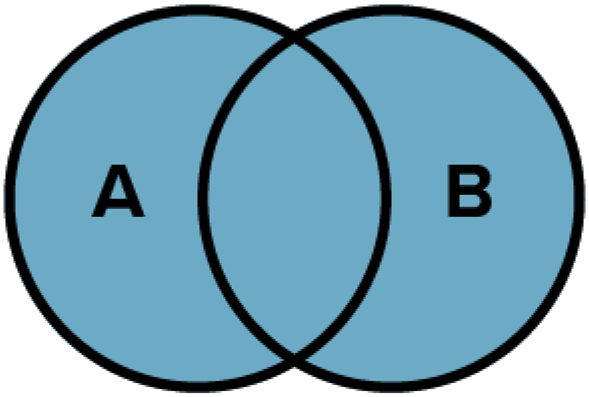
```
SELECT
        products.id
        ,inventory.product_id
        ,products.product_name
        ,inventory.product_vendor
        ,inventory.quantity
FROM
        products
RIGHT JOIN inventory ON products.id = inventory.product_id
```

Example Results

id	product_id	product_name	product_vendor	quantity
1	1	Apples	Apple Corp	15
2	2	Orange	Orange Ya-Glad	25
3	3	Grapes	Grape Escape	56
NULL	5	NULL	Banana Solutions	45

Full Join

The FULL JOIN returns all rows from both tables, regardless if there are matching values in the joined columns. You would use a FULL JOIN in cases where you want to show all applicable records from both tables, and synchronize the data across both tables via the joining columns where possible.



Full Join in Action

Note that we're using the same query as the standard JOIN, but we've prepended "FULL" to the last line of the query. Note the NULL values returned in cases where our product ID is not fully represented on both tables.

SQL - Full Joining Products and Inventory

```
SELECT
        products.id
        ,inventory.product_id
        ,products.product_name
        ,inventory.product_vendor
        ,inventory.quantity
FROM
        products
FULL JOIN inventory ON products.id = inventory.product_id
```

Example Results

id	product_id	product_name	product_vendor	quantity
1	1	Apples	Apple Corp	15
2	2	Orange	Orange Ya-Glad	25
3	3	Grapes	Grape Escape	56
4	NULL	Plums	NULL	NULL
NULL	5	NULL	Banana Solutions	45

Full Joins in MySQL

MySQL does not have an equivalent FULL JOIN. However, you can emulate one by utilizing a LEFT JOIN, RIGHT JOIN, and the UNION keyword. For the sake of simplicity, we will return all columns in the following example, but you would still want to specify individual columns in both SELECT query.

SQL - Full Join in MySQL using Left Join, Right Join, and Union Keyword

```
SELECT * FROM products
LEFT JOIN inventory ON products.id = inventory.product_id

UNION ALL

SELECT * FROM products
```

```
RIGHT JOIN inventory ON products.id = inventory.product_id
WHERE products.id IS NULL
```

Related Topics ...

- Writing SQL Queries
- SQL Query Data Source

SQL Common Functions

Functions are available in most SQL databases, and can provide some helpful utility to any queries you may be calling in Ignition.

This page contains some commonly used SQL functions that some databases contain. The exact functions available and usage depends on the database, so always check your database's documentation for a more complete list of available functions.

On this page ...

Using Column Values

It is important to understand that when calling these functions, you generally use a column name instead of a static number. For the sake of simplicity, the tables below demonstrate how to use the functions with static values, but they usually are switched for column names, ie:

Pseudocode - Passing a Column to a Function

```
SELECT SUM(downtime_duration) FROM downtime_events
```

Example Table

Some of the functions on this table are better demonstrated when used in conjunction with a table (i.e., using the AVG() function with a single value isn't too interesting). Thus, the following table contains sample data that the functions on this page will utilize if necessary.

Products Table

id	product_quantity	product_name	date_added	date_updated	origin_state
1	100	apples	Mon Jan 29 00:00:00 PST 2018	Mon Jan 29 12:00:00 PST 2018	California
2	24	oranges	Mon Feb 13 00:00:00 PST 2017	Mon Feb 13 09:00:00 PST 2017	Florida
3	56	grapes	Mon Mar 07 00:00:00 PST 2016	Mon Mar 07 05:00:00 PST 2016	California

Numeric Functions

Function	Description	Example	Output
ABS(value)	Returns the absolute value of the passed number or column.	`SELECT ABS(-3.5)`	`3.5`
AVG(value)	Takes the values of a single numeric column, and returns an average. A WHERE clause may be used in the same statement to filter out some of the rows on the table.	`SELECT AVG(product_quantity) FROM products`	`60`

CEILING(value)	Returns the next greatest integer value based on the argument provided. Thus, CEILING(10.1) would return 11.	`SELECT CEILING(10.1)`	11
COUNT(value)	Returns a row count. Typically takes either a single column, *, or 1. Regardless of which row is passed, the function will return the number of rows on the table that meet the criteria of any WHERE clauses.	`SELECT COUNT(*) FROM products`	3
FLOOR(value)	Returns the next smallest integer value based on the argument provided. Thus, FLOOR(10.9) would return 10.	`SELECT FLOOR(10.9)`	10
MAX(value)	Returns the largest value from the specified column.	`SELECT MAX(product_quantity) FROM products`	100
MIN(value)	Returns the smallest value from the specified column.	`SELECT MIN(product_quantity) FROM products`	24
ROUND(value, decimal_places)	Returns a number rounded to a certain number of decimal places. Takes two parameters. The first is the number to round to, and the second is the number of decimal places to round to.	`SELECT ROUND(1.234, 1)`	1.2
SUM(value)	Takes the value of a single numeric column, and returns the sum. A WHERE clause may be used in the same statement to filter out some of the rows on the table.	`SELECT SUM(product_quantity) FROM products`	180

String Functions

String Functions		**Example**	**Output**

CONCAT (value1, valu2,.... valueN)	Concatenates multiple strings or values. Some databases may require you to convert each value to a string before concatenating.	`SELECT CONCAT (product_name, ':', product_quantity) FROM products`	apples:100 oranges:24 grapes:56
LOWER (value)	Converts a string to lowercase.	`SELECT LOWER ('MAKE Me smALL')`	make me small
LTRIM(value)	Removes leading space from a string.	`SELECTCT LTRIM ('        Take a little off the left')`	Take a little off the left
REPLACE (orignal_string, target_string, replacement_string)	Searchings a string for a substring (target_string), and replaces the substring with the replacement_string.	`SELECT REPLACE('Who is awesome', 'Who is', 'You are')`	You are awesome
RTRIM(value)	Removes leading space from a string	`SELECT RTRIM ('Take a little off the right        ')`	Take a little off the right
SUBSTRING (orignal_string, character_index, [length])	Extracts a substring from another string based on character index. Takes two parameters: the original string, and the character index to start at. An optional third parameter can specify the number of characters to extract. Character index is one-based, so the first character in the string rests at index 1.	`SELECT SUBSTRING ('This is my string!' , 9, 9)`	my string
TRIM(value)	Removes both leading and trailing space from a string.	`SELECT TRIM ('   Trim Both Sides      ')`	Trim Both Sides
UPPER(value)	Converts a string to uppercase.	`SELECT UPPER ('super size me')`	SUPER SIZE ME

Date Functions

There are many date and time functions for each database (MySQL, MSSQL, Oracle, etc), but they all vary wildly. These examples work in most databases:

Function	Description	Example	Output

CURRENT_TIMESTAMP()	Returns the current date and time, as reported by the database.	`SELECT`
		`CURRENT_TIMESTAMP()`
			`Returns the current time`

TIMEDIFF (date1, date2)	Returns a difference between two dates. Assumes thate date1 is the most recent datetime.	`SELECT`
		`TIMEDIFF(date_updated, date_added)`
		`FROM`
		`products`
		`WHERE`
		`id =1`
			`Thu Jan 01 12:00:00 PST 1970`

Logic Functions

Function	Description	Example	Output

COALESCE (value1, value2,... valueN)	Returns the first non-null expression.	`SELECT`
		`COALESCE(NULL, 'Pick me!')`
			Pick me!

ISNULL (expression)	Returns true if an expression is NULL.	`SELECT`
		`ISNULL(NULL)`
		`SELECT`
		`ISNULL(14)`
			True
			False

NULLIF (expression1, expression2)	Compares two expressions. If they are equal to each other, then the function returns a NULL. If the two expressions are not equal, the first expression passed to NULLIF() is returned.	`SELECT`
		`NULLIF(100, 100)`
		`SELECT`
		`NULLIF(100, 3)`
			NULL
			100

Group By Clause

The GROUP BY statement is used in conjunction with the aggregate functions to group the result-set by one or more columns. This way you can find the MIN, MAX, Average, COUNT, etc., for each group of rows. Grouping can also be done for multiple columns, with precedence going in the order that they are listed.

Pseudocode - Passing a Column to a Function

```
SELECT SUM(column1) FROM table GROUP BY column2
```

Example

Here, we are grabbing the sum of product quantity for each origin state.

SQL - Simple Group By

```
SELECT SUM(product_quantity), origin_state, FROM products_table GROUP BY origin_state
```

Related Topics ...

- Writing SQL Queries
- SQL Query Data Source
- system.db.runNamedQuery

SQL Stored Procedures

For the uninitiated, Stored Procedures are a series of predefined SQL statements that are configured and stored in the database, and can be easily called from another application, such as Ignition, with a single statement. Conceptually, they are very similar to a scripting function: parameters may be passed to them, they can utilize looping and conditional logic, and can return a value.

On this page ...

Stored Procedure Syntax

Stored procedures are created and maintained in your database. As a result, the creation of a stored procedure is outside the scope of the Ignition User Manual: the commands used to create a stored procedure vary per database.

In regard to calling a Stored Procedure, the syntax can also differ.

SQL Server

The EXEC command is utilized to execute a stored procedure.

Pseudocode - Executing a Stored Procedure in SQL Server

```
EXEC dbo.myStoredProcedure
```

Parameters may be passed to the Stored Procedure. SQL Server's documentation has more details on utilizing parameters with Stored Procedures.

Pseudocode - Executing a Stored Procedure in SQL Server with Parameters

```
EXEC dbo.myStoredProcedure @myParam = 100, @AnotherParameters = 
'Hello'
```

MySQL

MySQL uses the CALL command to execute a Stored Procedure. Note the parentheses characters at the end, as they must be present even when the Stored Procedure is not utilizing parameters.

Pseudocode - Executing a Stored Procedure in MySQL

```
CALL myStoredProcedure()
```

If parameters are defined, they can be passed via the parenthesis, similar to how functions work in Python.

Pseudocode - Executing a Stored Procedure in MySQL with Multiple Parameters

```
CALL myStoredProcedure(100, 'Hello')
```

For information on the creation of a stored procedure, as well as proper SQL syntax to call a Stored Procedure, reference your database's documentation. Alternatively, if you have a database administrator, they can typically help with the creation and execution of a Stored Procedure.

Calling Stored Procedures in Ignition

There are several locations in Ignition where Stored Procedures may be utilized from.

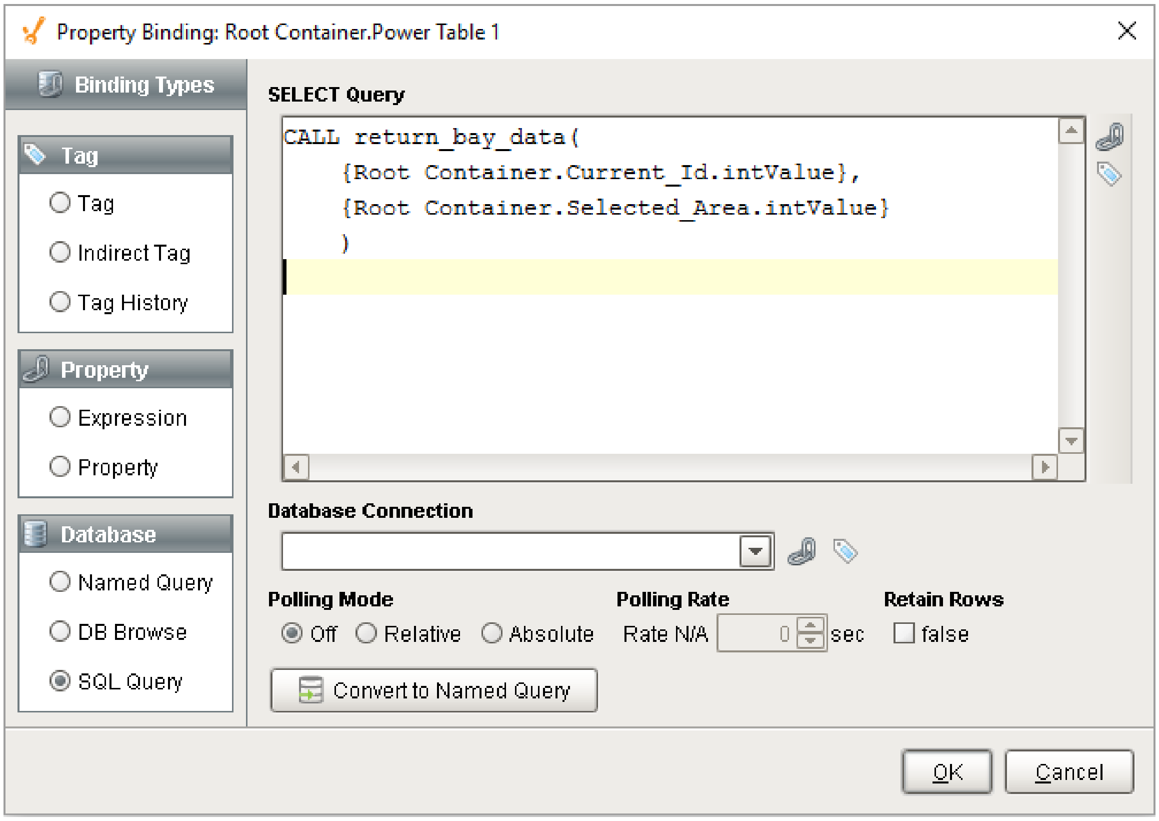
SQL Query Bindings

Instead of typing a query directly into a SQL Query binding, a Stored Procedure may be executed instead. Assuming a MySQL database contains a Stored Procedure named 'return_all_bays', we can call the procedure on a binding with the following:

SQL - Calling a Stored Procedure in MySQL

```
CALL return_all_bays()
```

As with all bindings, Tag and Property references may be utilized by clicking the appropriate icons in the binding window.



Property Binding: Root Container.Power Table 1

SELECT Query

```
CALL return_bay_data(
    {Root Container.Current_Id.intValue},
    {Root Container.Selected_Area.intValue}
    )
```

Database Connection:

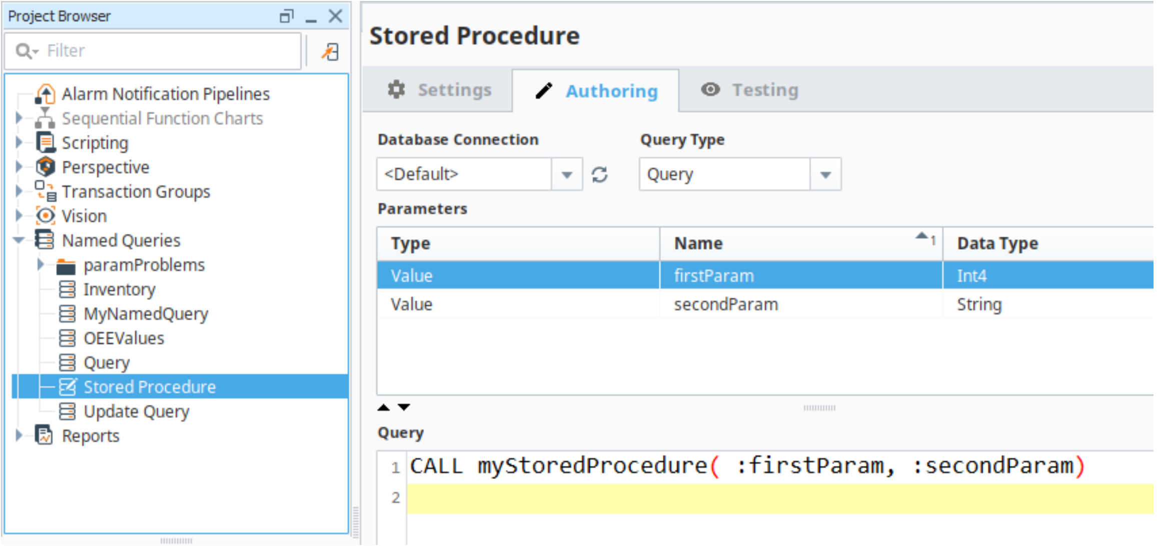
Polling Mode: ● Off ○ Relative ○ Absolute

Polling Rate: Rate N/A 0 sec

Retain Rows: □ false

Named Queries

Named Queries support Stored Procedure calls. As mentioned under Stored Procedure Syntax, the syntax is based entirely on how your database expects a Stored Procedure to be called. Be mindful of the Query Type setting, as it has to match what the stored procedure is doing: if it is returning a result set, leave it set to Query, if it is modifying a record in the database, then set the type to Insert Query.



Database Connection: <Default>

Query Type: Query

Parameters

Type	Name	Data Type
Value	firstParam	Int4
Value	secondParam	String

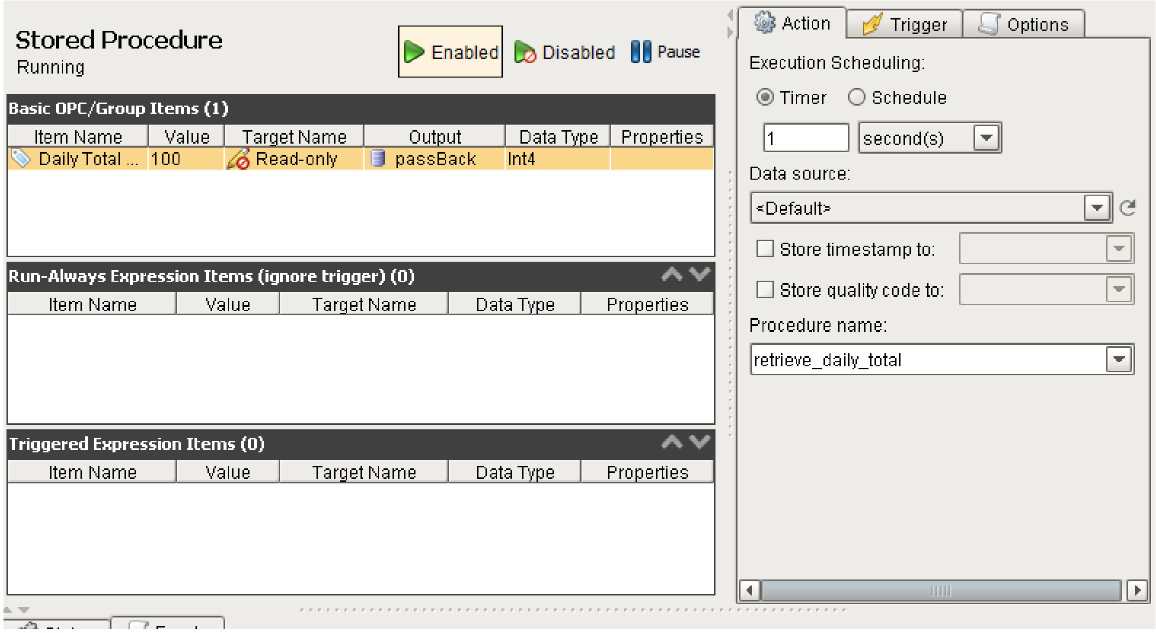
Query

```
CALL myStoredProcedure( :firstParam, :secondParam)
```

Stored Procedure Groups

One of the easiest ways to utilize Tags with a Stored Procedure is to use the Stored Procedure Group. Parameters can be easily assigned to each item in the group, and utilize all of the features of a Transaction Group, such as scheduled execution and triggers.

Each item in the group is linked to a specific parameter in the Stored Procedure. Any IN or INOUT parameters can write directly to the Tags, while new values can be fed into OUT and INOUT parameters allowing you to easily move data from Tags into the database with the Stored Procedure.



Basic OPC/Group Items (1)

Item Name	Value	Target Name	Output	Data Type	Properties
Daily Total ...	100	Read-only	passBack	Int4	

Run-Always Expression Items (ignore trigger) (0)

Item Name	Value	Target Name	Data Type	Properties

Triggered Expression Items (0)

Item Name	Value	Target Name	Data Type	Properties

Execution Scheduling:

● Timer ○ Schedule

1 second(s)

Data source: <Default>

☐ Store timestamp to:

☐ Store quality code to:

Procedure name: retrieve_daily_total

Stored Procedures in Python Scripts

There are a few ways to call a Stored Procedure from a script in Ignition.

Using createSProcCall and execSProcCall

The recommended approach to calling a Stored Procedure from a Python script in Ignition typically involves two main steps:

1. Calling system.db.createSProcCall to create a call context, or object that effectively represents the impending stored procedure call. This object can be used to specify parameter values that will be passed to the Stored Procedure.
2. Using system.db.execSProcCall to execute the Stored Procedure.

Once the Stored Procedure has been executed, the call context generated in step #1 can be used to reference any values that were returned.

Other System Functions

Technically, most other system functions in the "db" library, such as system.db.runPrepQuery, can be used to call a Stored Procedure. We generally recommend against this, as system.db.createSProcCall and system.db.execSProcCall are better suited to work with Stored Procedures and have some additional functionality not found in the other db functions.

Related Topics ...

- SQL Query Bindings in Vision
- Understanding Transaction Groups

Query Builder

On this page

...

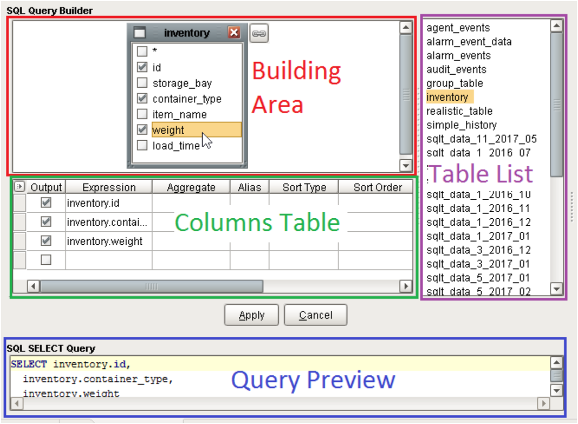
Crafting Queries with the Query Builder

Many places in Ignition that allow for SQL queries have a link to the SQL Query Builder tool. The Query Builder is a powerful Drag-and-Drop query building GUI that allows you to make complex queries from your connected databases. While a basic understanding of SQL helps make the most of the Query Builder tool, most people will have no problem creating effective queries after a brief tutorial. Additionally, the Query Builder does go over many advanced features of SQL that may be unfamiliar. We suggest looking up how these work in your favorite SQL resource guide as this covers how to use them, not what they do.

Builder Anatomy

Once opened, the Query Builder has the following items:

- **Building Area**: Visualizes the table relationships of all tables included in the query. Tables are typically dragged in from the **Table List**.
- **Columns Table**: Shows which columns from all tables are referenced in the query.
- **Table List**: Shows all database tables in the selected Database Connection.
- **Query Preview**: Shows a preview of the query that will be created once the **Apply** button has been pressed.



Output	Expression	Aggregate	Alias	Sort Type	Sort Order
☑	inventory.id				
☑	inventory.contai...				
☑	inventory.weight				
□					

```
SELECT inventory.id,
  inventory.container_type,
  inventory.weight
```

Opening the Query Builder

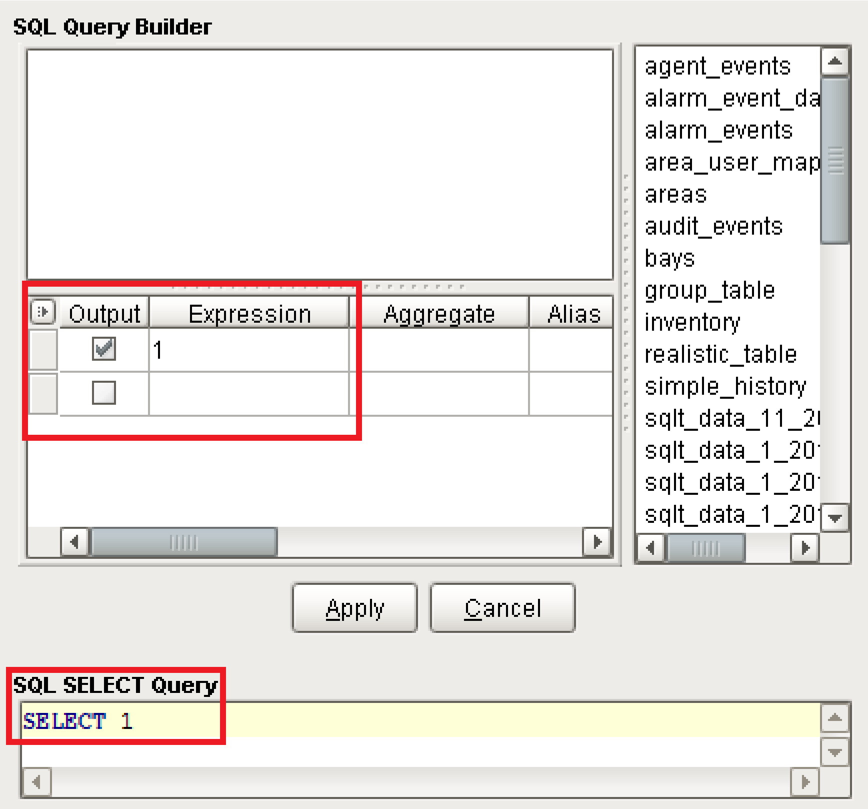
The Builder can be accessed from several different resources in Ignition, notably when using Named Queries and setting up Report Data. To use the Builder, click the **Show Builder** button.

Clicking this button will open the Builder workspace with an empty query. If a query was already written before the button was pressed, then the Builder will attempt to load the query into the Builder's interface. For example, if the following query was typed before pressing the button:

SQL - Select Statement - Loading into the Builder's Workspace

```
SELECT 1
```

The following would be loaded into the Builder's workspace. Note that the **Building Area** is empty, because we're not querying from a table.



Output	Expression	Aggregate	Alias
☑	1		
□			

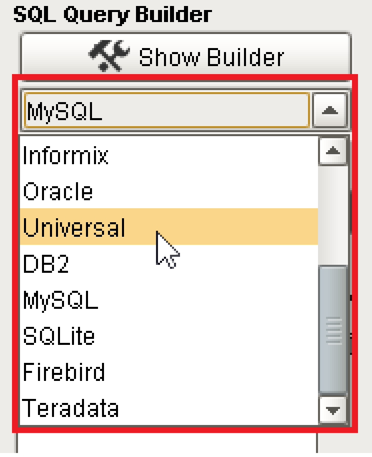
SQL SELECT Query

```
SELECT 1
```

The Builder attempts to parse your query based on the **Syntax Parser**.

Using the Syntax Parser

When starting with a pre-existing query, the Syntax Parser tells the Builder which implementation of SQL your query's syntax is using. In most cases, this can be left with the default value of **Universal**. When set to a specific implementation of SQL, this allows the Builder to understand/accept implementation-specific keywords and syntax.



Syntax Parser in Action

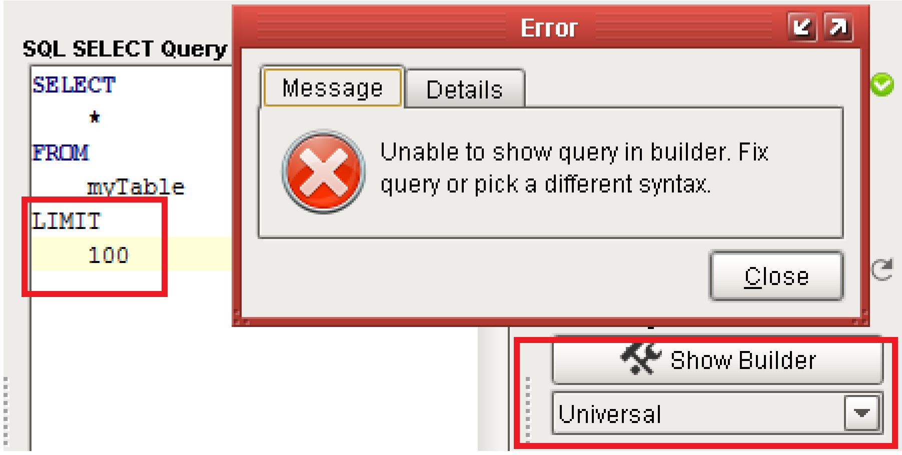
In MySQL, query results may be limited with the **LIMIT** keyword:

SQL - Limit Statement

```
SELECT
        *
FROM
```

```
        myTable
LIMIT
        100
```

Attempting to open the Builder while using **LIMIT**, and the Syntax Parser is configured to a syntax that does recognize the **LIMIT** keyword, will result in an error:



```
SELECT
    *
FROM
    myTable
LIMIT
    100
```

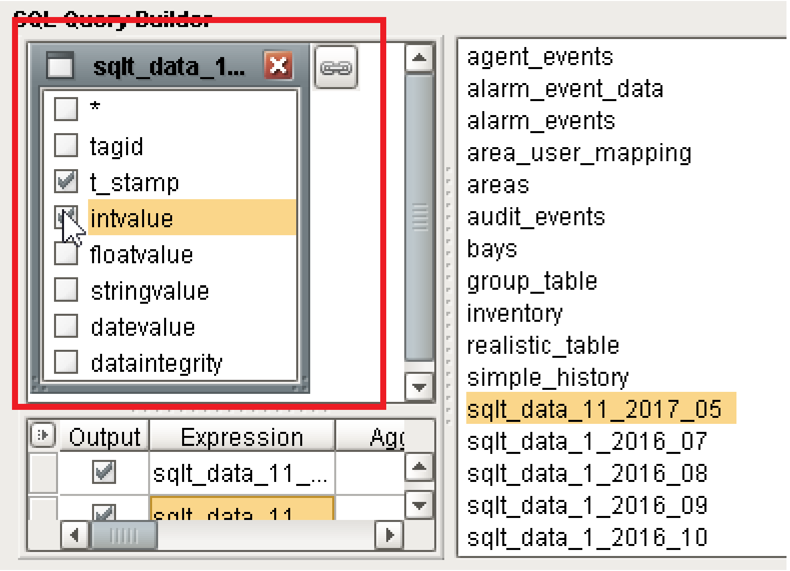
Unable to show query in builder. Fix query or pick a different syntax.

In these scenarios, you can switch to a syntax that supports the query, or remove the offending lines.

Using the Builder

Once the Builder is open, (and if you have tables in your database connection), you can start building a query by dragging and dropping a table from the Table List into the Building Area. From here, you can select which columns on the Table Object you want to bring into the query, or drag and drop them into the Columns Table below.

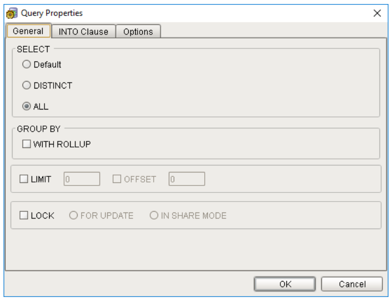
The ***** option at the top of the table is special. If the ***** character is selected, then all columns from the table will be included in the results just like a "SELECT * " query.



Query Properties

Right clicking on the empty space in the Building Area and selecting properties will bring up the Query Properties window that allows you to customize how the query works. How it looks and what it contains can vary, depending on what syntax parser you have selected.

MySQL Query Properties



Query Properties

General | INTO Clause | Options

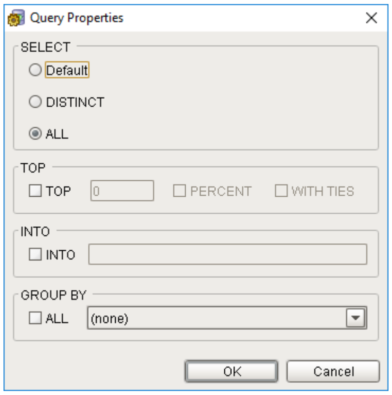
SELECT
○ Default
○ DISTINCT
● ALL

GROUP BY
□ WITH ROLLUP

□ LIMIT 0 □ OFFSET 0

□ LOCK ○ FOR UPDATE ○ IN SHARE MODE

MSSQL Query Properties



Query Properties

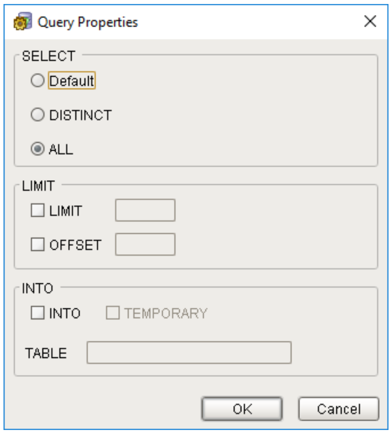
SELECT
○ Default
○ DISTINCT
● ALL

TOP
□ TOP 0 □ PERCENT □ WITH TIES

INTO
□ INTO

GROUP BY
□ ALL (none)

PostgreSQL Query Properties



Query Properties

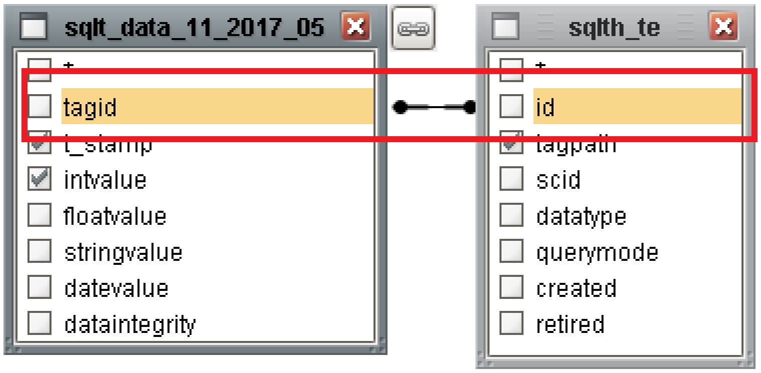
SELECT
○ Default
○ DISTINCT
● ALL

LIMIT
□ LIMIT
□ OFFSET

INTO
□ INTO □ TEMPORARY
TABLE

Joining Tables in the Query Builder

Multiple tables may be added to the Building Area. Once there, the Builder can JOIN the two tables by dragging from one column in a table on to another column on a different table.



sqlt_data_11_2017_05
□ *
□ tagid
☑ t_stamp
☑ intvalue
□ floatvalue
□ stringvalue
□ datevalue
□ dataintegrity

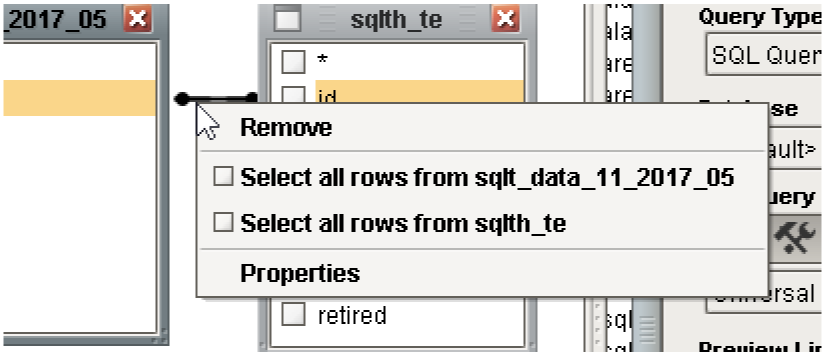
sqlth_te
□ *
□ id
☑ tagpath
□ scid
□ datatype
□ querymode
□ created
□ retired

Note that "tagid" and "id" are now linked together. This means the resulting query will JOIN the two tables based on the linked columns. Much like a SQL JOIN, the joining columns do not need to be selected to be utilized in the JOIN.

Joins Right-Click Menu

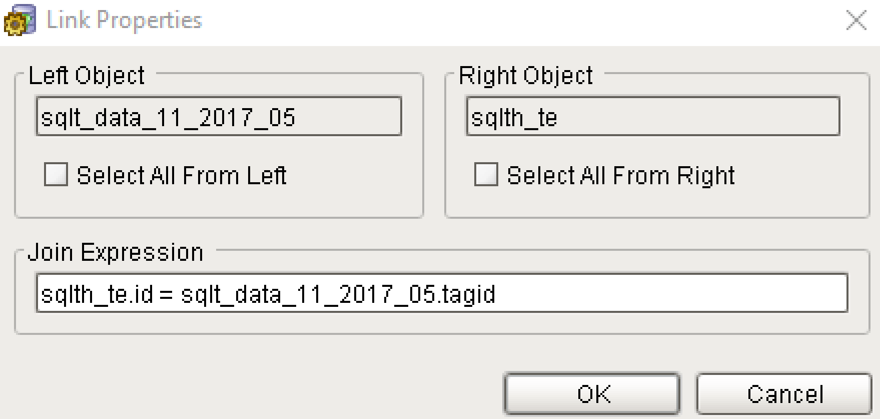
Right-clicking on the link will create a menu that allows you to remove the **JOIN**, as well as access properties of the JOIN.

Additionally, you can select all rows from one of the joining tables, which effectively creates a **RIGHT OUTER JOIN** or **LEFT OUTER JOIN**, depending on which table you select.


2017_05
sqlth_te
*
id
retired
Remove
☐ Select all rows from sqlt_data_11_2017_05
☐ Select all rows from sqlth_te
Properties
Query Type
SQL Quer

Join Properties

Clicking on **Properties** on the Right-Click Menu allows you to view the relation between each table, as well as apply **LEFT** or **RIGHT JOINs**, as mentioned above.


Link Properties

Left Object: sqlt_data_11_2017_05

☐ Select All From Left

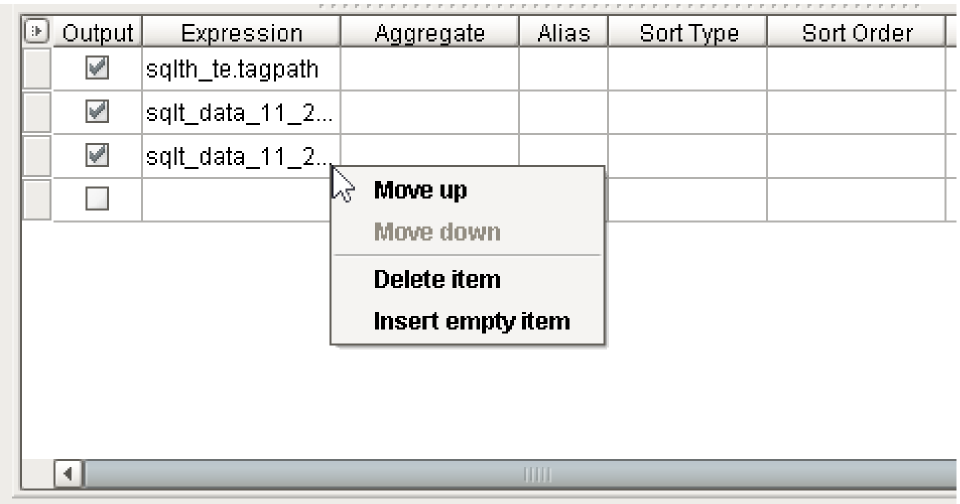
Right Object: sqlth_te

☐ Select All From Right

Join Expression: sqlth_te.id = sqlt_data_11_2017_05.tagid

Columns Table

The Columns Table allows you to modify individual columns in the query. Typically, this allows you to add aggregates or aliases to each column from the Builder. Each row represents an expression, or combination of functions, columns, variables, and constants that will ultimately be a single column in the resulting query.



Output	Expression	Aggregate	Alias	Sort Type	Sort Order
☑	sqlth_te.tagpath				
☑	sqlt_data_11_2...				
☑	sqlt_data_11_2...				
☐					

Right-Clicking on an Item

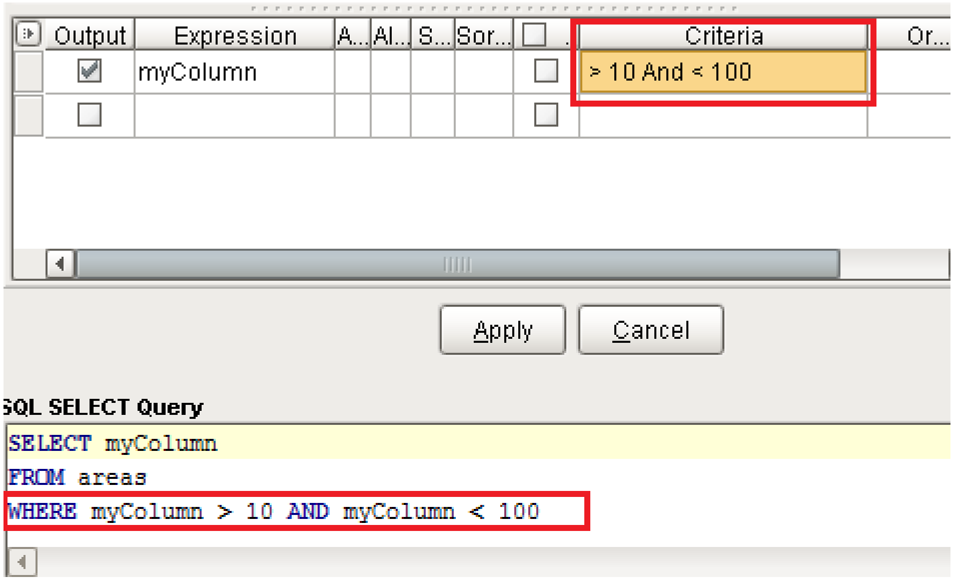
Right-clicking on a row of the Columns Table brings up a menu with the following options:

Option	Description
Move up	Moves the selected row up in the table. This means the column will appear in the query results prior to the other listed columns.
Move down	Similar to move up, but moves the row down, or towards the end of the listed columns.
Delete item	Removes the row from the Columns Table, effectively removing it from the resulting query.
Insert empty item	Adds a new row to the Columns Table with blank values.

Field Description

Each column in the Columns Table allows you to modify the resulting query in some way. The following is a description of each column:

Name	Desciption
Output	Specifies if the row should appear in the resulting query. Disabling a row on the table removes the column from the results.
Expression	The column from one of the tables in the Building Area that this row represents. Left-clicking on this field will create a dropdown list of possible rows.
Aggregate	Allows you to aggregate the expression. Blank by default (no aggreggate), configuring this column allows you to do things like sum column, or return the average. Possible values are: • Blank (no aggregation will be performed) • Avg • Count • Max • Min • Sum You can also add the Distinct keyword to any non-blank option. For example, you could enter "Sum Distinct" or "Count Distinct".
Alias	Creates an Alias, or alternative name for the column. Effectively adds the SQL "AS" keyword to the column, which allows you to rename the resulting column.
Sort Type	Sorts the results based on this column. Enabling a Sort Type is similar to added an ORDER BY to your query. Possible values are: • Blank (no sorting on this column will be performed) • Ascending • Descending
Sort Order	When a **Sort Type** has been specified on multiple rows, this determines which row the query will sort on first.
Grouping	Allows you to group the results. Adds a GROUP BY statement to the resulting query. Enabling grouping on one of the columns makes the **Criteria for** column appear.
Criteria for	A dropdown list that sets how you want the Criteria column to apply to the grouping. Does not apply if both Grouping and the WHERE clause Criteria are not being used. • For Values will place the WHERE clause specified in the Criteria column first, filtering out rows that don't apply, and then applying the grouping. • For Groups will instead change the WHERE clause into a HAVING clause that is used to filter after the grouping has taken place.
Criteria	Allows you to add a WHERE clause. Supports the use of both the **OR** and **AND** keywords for multiple conditions. Example: assuming a column named "myColumn", we could limit the results of our query to rows where myColumn has a value greater than 10 AND less than 100:

	 Output ☑ Expression: myColumn Criteria: > 10 And < 100 SQL SELECT Query SELECT myColumn FROM areas WHERE myColumn > 10 AND myColumn < 100 Output Expression A... Al... S... Sor... Criteria Or... Apply Cancel
Or...	Allows you to add an additional condition to a WHERE clause. Will separate each grouped condition with "()" characters to maintain logic.

⚠ The Query Builder is a third party tool that we brought into Ignition and while we go over how to use it here, you can also check out the Active Query Builder's documentation for more information on how this feature works.

Related Topics ...

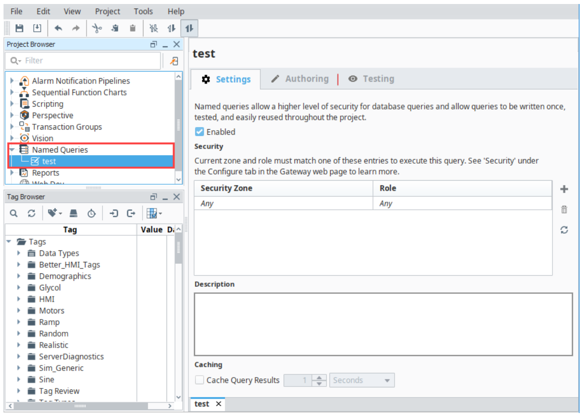
- Named Queries
- Named Query Data Source

Named Queries

What Are Named Queries?

Named Queries are preconfigured queries that may be invoked elsewhere in a project. Named Queries are conceptually very similar to Project Scripts; defined in a single location, and then referenced in multiple places throughout the project. When executing a Named Query, parameters may be passed to return a dynamic result set. This way, a query may be written once, and then called from multiple locations throughout the project.

Named Queries have their own workspace in the **Project Browser** section of the **Designer**.



test

Settings | Authoring | Testing

Named queries allow a higher level of security for database queries and allow queries to be written once, tested, and easily reused throughout the project.

☑ Enabled

Security

Current zone and role must match one of these entries to execute this query. See 'Security' under the Configure tab in the Gateway web page to learn more.

Security Zone	Role
Any	*Any*

Description

Caching

□ Cache Query Results 1 Seconds

test ✕

On this page

...

- What Are Named Queries?
- Secure Query Execution
- Where are Named Queries Used?
  - Binding
  - Reporting
  - Scripting

INDUCTIVE UNIVERSITY

Named Query Overview

Watch the Video

Secure Query Execution

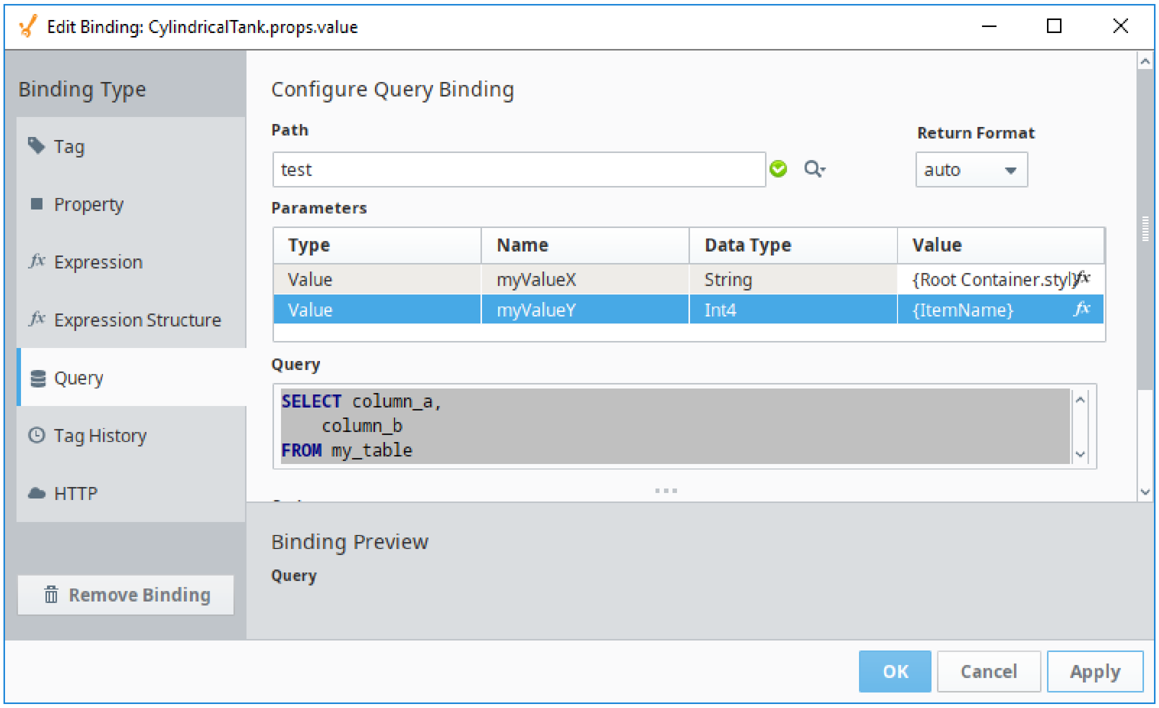
While clients may request data from a Named Query, the actual execution of the query always takes place on the Gateway. Clients simply specify which query should run, and pass parameters that the Gateway will use. Additionally, the Gateway has an opportunity to restrict access to the query based on Security Zone and/or User Role. This provides a single interface to restrict access to the queries, and better protect your data. Additionally, queries cannot be modified by a Client other than by passing variables into it. This creates a very secure method to control what queries are being run against your database.

Where are Named Queries Used?

Named Queries may be used in multiple locations in Ignition. They are used in all the same places that a normal SQL query would be used. Here are a few examples of locations in which named queries can be used. A more detailed example of a named query is provided here.

Binding

A Query Binding type has been added to leverage named queries for any component property bindings. Named Query Parameters may be bound to Tags or other properties on the same window, allowing your users to modify the resulting dataset by manipulating other components similar to the original SQL Query binding type. You can also select and update a query to simulate a bi-directional binding to the database.



Path: test

Return Format: auto

Parameters

Type	Name	Data Type	Value
Value	myValueX	String	{Root Container.styl
Value	myValueY	Int4	{ItemName}

Query

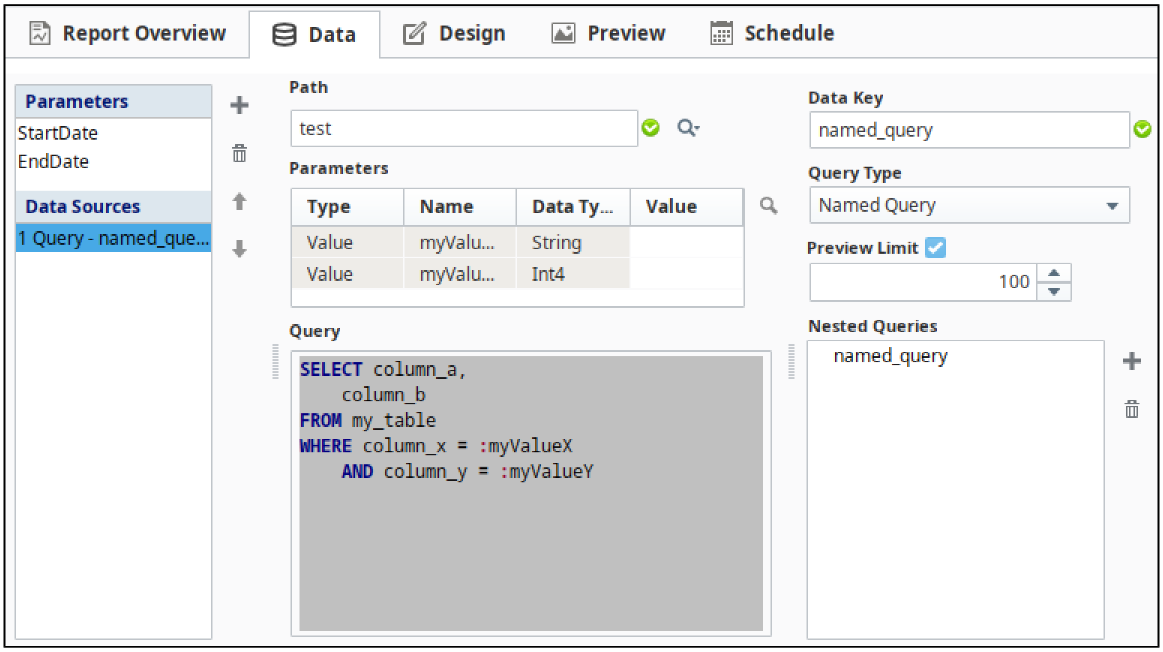
```
SELECT column_a,
    column_b
FROM my_table
```

Binding Preview

Query

Reporting

A Named Query Data Source has been added to **Reports**. Report Parameters may be used by the Named Query to generate dynamic reports.



Parameters
StartDate
EndDate

Data Sources
1 Query - named_que...

Path: test

Parameters

Type	Name	Data Ty...	Value
Value	myValu...	String	
Value	myValu...	Int4	

Query

```
SELECT column_a,
    column_b
FROM my_table
WHERE column_x = :myValueX
    AND column_y = :myValueY
```

Data Key: named_query

Query Type: Named Query

Preview Limit ☑ 100

Nested Queries
named_query

Scripting

The system.db.runNamedQuery() function may be used to called a Named Query from any Python Script. This provides a secure method to execute a query from any location in Ignition.

Related Topics ...

- system.db.runNamedQuery
- Named Query Data Source
- Named Query Bindings

In This Section ...

Named Query Workspace

Named Query Workspace Overview

Named Queries have a dedicated workspace inside of the Project Section of the Designer. This workspace allows for the creation and testing of Named Queries. Once created, the Named Query may be called from another resource, such as a datasource in a report or a component using a Named Query Bindings.

Named Queries are created by **right-clicking** the Named Query item in the Project Browser. Like other resources in the Project Browser, Named Queries can be organized in folders, which creates a unique path to the query and helps keep your queries organized.

⚠ Named Queries are referenced by path, so renaming the Named Query or any parent folders will require you to update the path on any other resources that are using it.

Also like other resources, multiple Named Queries may be opened in the same Designer session. Tabs at the bottom of the Designer allow for easy swapping between Named Queries.

On this page

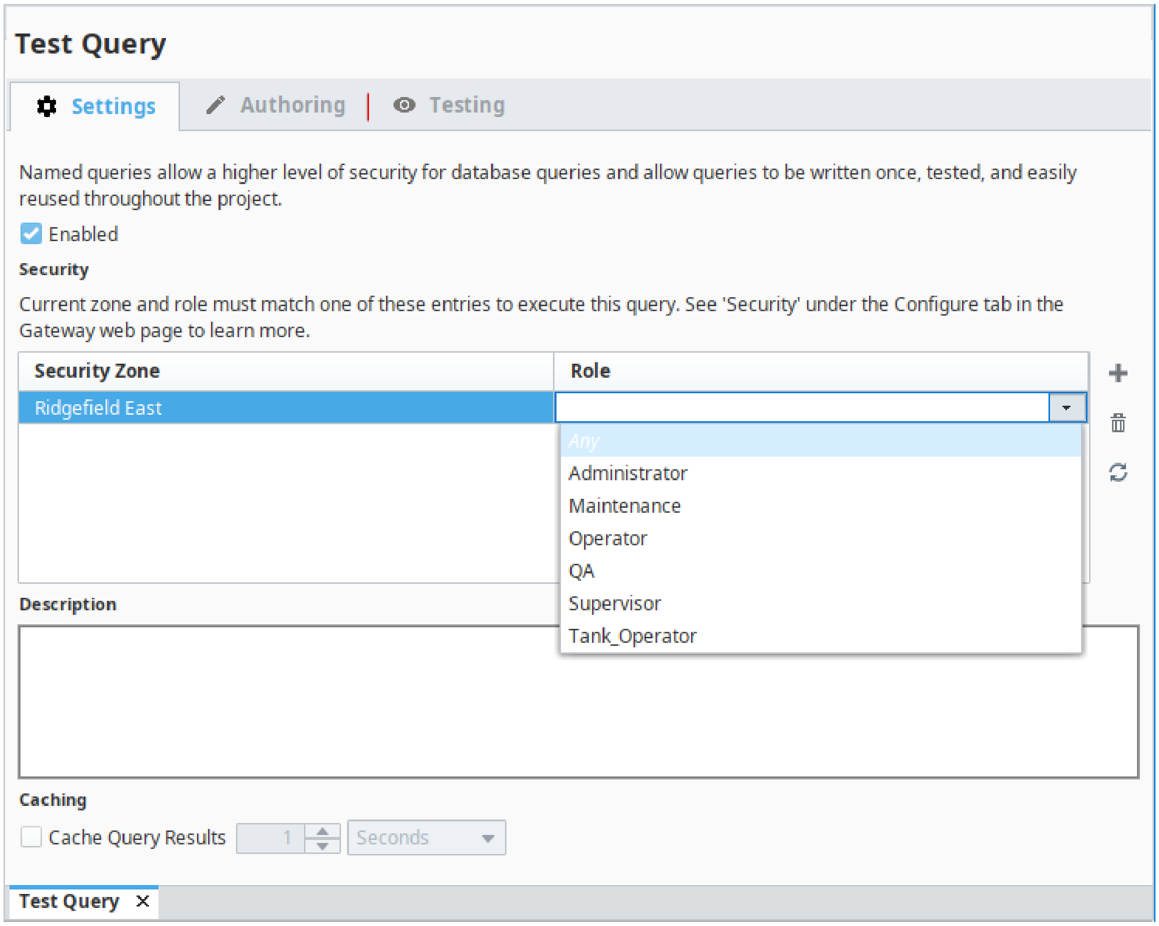
...

Workspace Sections

The Named Query workspace contains three tabs: Settings, Authoring, and Testing. A description of each section follows.

Settings

The Settings ⚙ tab contains configuration properties and security for the selected Named Query.



Named queries allow a higher level of security for database queries and allow queries to be written once, tested, and easily reused throughout the project.

☑ Enabled

Security

Current zone and role must match one of these entries to execute this query. See 'Security' under the Configure tab in the Gateway web page to learn more.

Security Zone	Role
Ridgefield East	

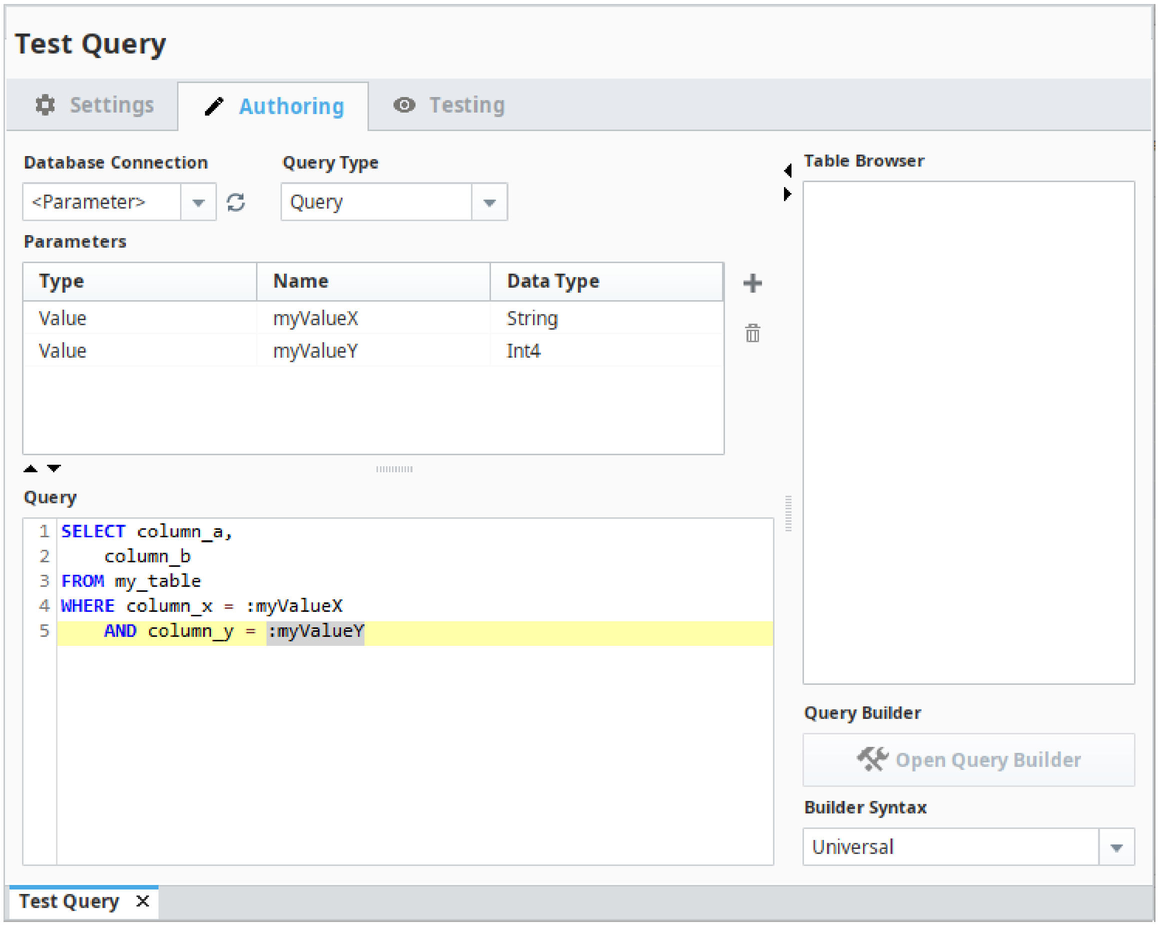
□ Cache Query Results 1 Seconds

The following properties are available:

Item Name	Description
Enabled	Determines if the Named Query is enabled or disabled. A disabled Named Query may not be executed.
Security	Specifies a combination of Security Zones and Roles that may call the Named Query. Only roles in the projects user source will be available in the Role dropdown. Multiple rows may be configured to account for granular access restrictions (i.e., requests from **Administrator** roles originating from the **Office** security zone area could be allowed, while requests from same users in the **plant floor** zone could be denied). If the request does not match any of the specified zone and role combinations, then the query will not run. Additionally, if a **Scalar Query** type has a Fallback value configured, that Fallback value will not be returned either: the query will not execute due to security settings, so there is never a chance for other errors to occur. Either the Security Zone or the Role (but not both) may be left blank. This means it is available to all objects of that type (i.e., with a blank Role and the "office" Security Zone, all roles in the "office" zone are valid).
Description	Allows you to give the Named Query a description.
Caching	Allows the Gateway to cache the results of the query. See the Named Query Caching page for more details.

Authoring

The Authoring ✏ tab is where the query and parameters are created. There is also a Table Browser and Query Builder that can be used to help you to create your query.



Test Query

Database Connection: <Parameter>

Query Type: Query

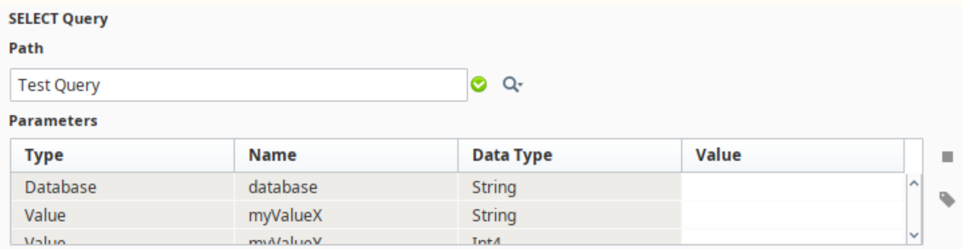
Parameters

Type	Name	Data Type
Value	myValueX	String
Value	myValueY	Int4

Query

```
SELECT column_a,
    column_b
FROM my_table
WHERE column_x = :myValueX
    AND column_y = :myValueY
```

Builder Syntax: Universal

Item Name	Description
Database Connection	The database connection the Named Query should run against. In addition to a list of the database connections configured in the Gateway, this dropdown contains two unique values: **<Default>** and **<Parameter>**. - **<Default>**: The query will execute against the project's default database connection. - **<Parameter>**: The query expects the database connection name to be passed in as a parameter when called. This allows you to use a dynamic database connection. ⚠ A parameter for the database connection does not need to be manually created in the Parameters section. Instead, the Named Query will have a special "Database" type parameter available when called. Below is an image of a Named Query binding that is utilizing the <Parameter> connection type.  SELECT Query Path: Test Query Parameters Type | Name | Data Type | Value Database | database | String | Value | myValueX | String | Value | myValueY | Int4 |
Query Type	The type of query to execute. The following options are available: - **Query**: Allows SELECT Queries and returns a full dataset. This type should be selected when running a SELECT statement that returns multiple rows or columns.

	- **ScalarQuery**: Allows SELECT queries and returns a single value. This type should be used when running SELECT statements that only return a single value. The very first cell returned will be the only output. This type is special in that a **Fallback** value may be defined. The Fallback value will be returned if the Named Query would return an error. Note, the **Fallback** parameter will not be returned if the request does not meet the security requirements. - **UpdateQuery**: Allows all **UPDATE** types of queries (querys that mutate or otherwise modify rows on a table) and returns the number of rows affected by the query. This type should be selected when modifying the database in some way, such as when running an INSERT, UPDATE, or DELETE query.
Parameters	A table of the parameter names and types that will be used in the query. These parameters have three types, Value, QueryString, and Database. Most commonly, the **Value** type is used and can be accessed by using the **:paramName** notation. More details on this field may be found on the Named Query Parameters page.

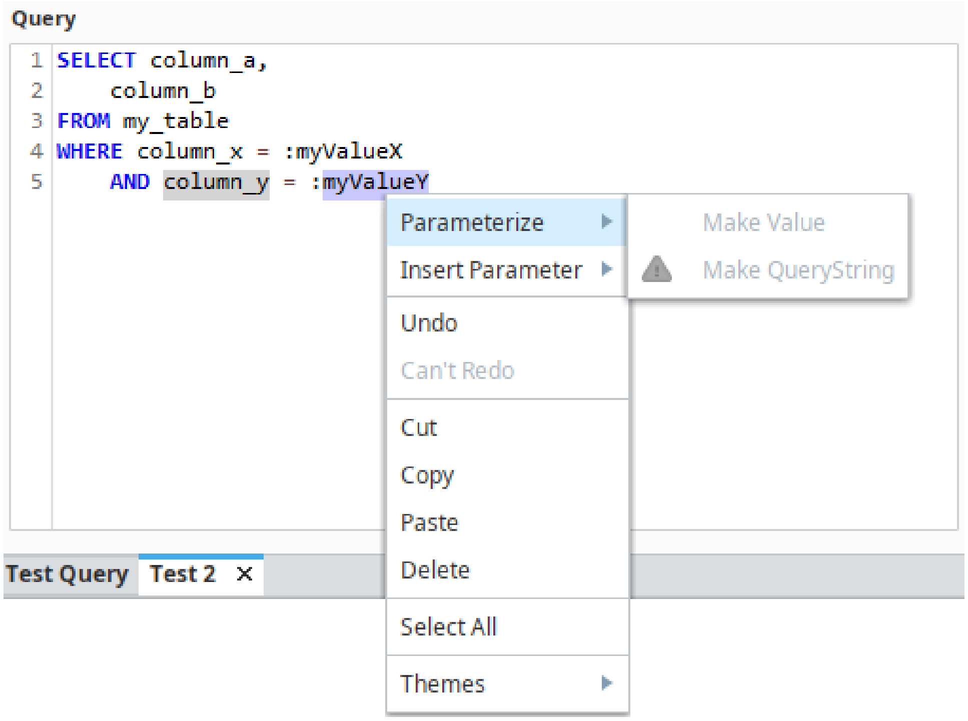
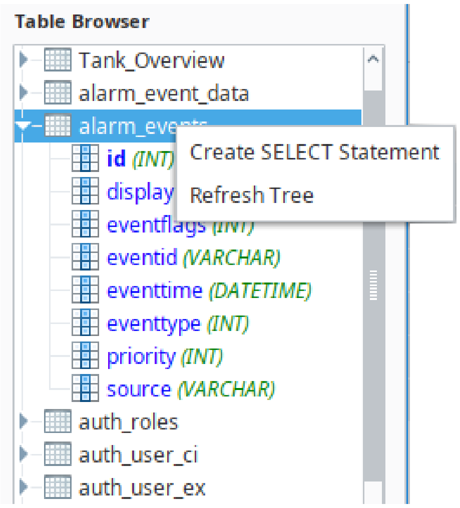
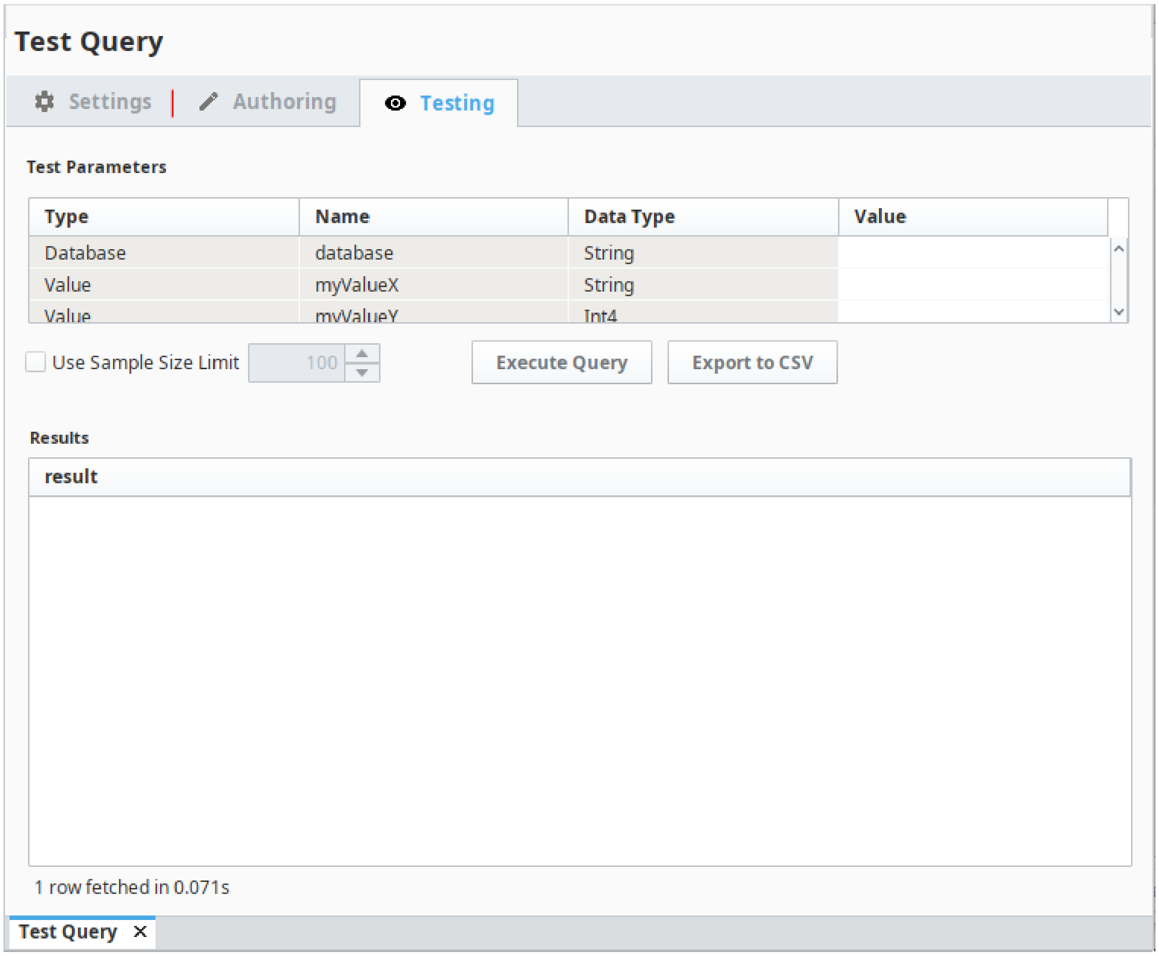
Query	The query that will execute when the Named Query is called. You can type directly into this field or use the Table Browser on the right to get started. **Right-Clicking** inside this field will cause a popup menu to appear:
	
	 Query
	Parameterize
	Insert Parameter
	Undo
	Can't Redo
	Cut
	Copy
	Paste
	Delete
	Select All
	Themes
	Make Value
	Make QueryString
	Test Query
	Test 2 ✕
	`SELECT column_a,`
	`    column_b`
	`FROM my_table`
	`WHERE column_x = :myValueX`
	`    AND column_y = :myValueY`
	Most of the items on this menu are self-explanatory, but a few require special mention:
	Parameterize: Contains two sub-menu items, which are detailed below. Note, that the sub-items will be **disabled** unless you right-click on some text that does not reference a parameter, table name, or column name in the query.
	○ **Make Value**: Turns the selected text into a value-type parameter. The new value-type parameter will appear in the **Parameter** table above the query.
	○ **Make QueryString**: Turns the selected text into a QueryString. The new QueryString will appear in the **Parameter** table above the query. Note that QueryStrings are susceptible to SQL injection attacks. Because of this, the **Make Value** option is recommend over this option.
	Insert Parameter: Quickly creates a reference to the selected parameter. This menu is an alternative to dragging-and-dropping from the Parameters table or typing the name of the parameter.

Table Browser	Provides a list of the tables in the selected **Database Connection**. Tables may be dragged into the Query field to quickly insert the name of the table. Additionally, **right-clicking** on a table in the list will cause a popup menu to appear:

	 Table Browser Tank_Overview alarm_event_data alarm_events id (INT) display eventflags (INT) eventid (VARCHAR) eventtime (DATETIME) eventtype (INT) priority (INT) source (VARCHAR) auth_roles auth_user_ci auth_user_ex Create SELECT Statement Refresh Tree • **Create SELECT Statement** will populate the Query field with a SELECT statement targeting the selected table or selected row of the table. • **Refresh Tree** will refresh the Table Browser.
Query Builder	Opens the Query Builder, which provides an easy way to create SQL queries using a drag-and-drop interface. This button will be disabled if the **Database Connection** property is set to **<Parameter>**. This is the same Query Builder used in other places like the reporting data page.
Builder Syntax	Specifies the syntax the **Query Builder** should use. Contains syntax for many popular databases, and has a Universal selection that should work in most scenarios.

Testing

In the Testing 👁 tab, you can test your query without leaving the workspace. Fill in your values and click the **Execute Query** button to see your results.



Test Query

Settings | Authoring | Testing

Test Parameters

Type	Name	Data Type	Value
Database	database	String	
Value	myValueX	String	
Value	myValueY	Int4	

□ Use Sample Size Limit 100

Execute Query | Export to CSV

Results

result

1 row fetched in 0.071s

Test Query ✕

Item Name	Description
Test Parameters	Allows you to manually supply test values to the parameters to the queries. The table is populated by the **Parameters** field on the Authoring section.
Use Sample Size Limit	When checked, allows you to set the maximum number of rows the query will return while testing the Named Query. In addition this setting also limits the results of that named query called elsewhere in the Designer. This property is only enabled when the **Query Type** on the Authoring section is set to **Query**.
Execute Query	Runs the Named Query using the parameter values listed above. The **Results** area will display any results returned by the query.
Export to CSV	Exports the results of the query to a CSV file. The button becomes available after results are returned. Note that this will only return the values shown, check the Sample Size Limit when using this button.
Results	The results returned by the Named Query when testing. Populated by the **Execute Query** button.

Related Topics ...

- Named Query Parameters
- Named Query Conversions
- Named Query Caching

Named Query Parameters

Parameters allow you to make Named Queries dynamic. They act as placeholders you can pass values into when requesting the query to execute. Other resources in Ignition can then pass arguments into the parameters. The exact implementation depends on what resource is requesting the Named Query, such as a Named Query Bindings, a Named Query Data Source, or the system.db.runNamedQuery function. Check out the Named Query Example page for a complete example on passing parameters into a named query.

On this page ...

Types of Named Query Parameters

There are three types of Parameters in Named queries. Each varies in usage.

Value

The Value type should be used whenever a Named Query needs a dynamic WHERE clause. These act like values passed to a prepared statement, meaning they can never be used to parameterize column or table names. However, they are resilient to SQL injection attacks.

SQL Query - Using a Parameter

```
SELECT * FROM mytable
WHERE name = :myParam
```

QueryString

QueryStrings are more flexible than the Value type in that they can be used to parameterize column and table names. However, their values are never sanitized, which causes them to be more susceptible to SQL injection attacks. When using QueryStrings, it is best to avoid situations where the user can manually type in the value that will be passed to the Name Query. Additionally, if you are using a QueryString for a string in the where clause, you would need to provide quotation marks.

SQL Query - A Using QueryString

```
SELECT {myColumnName} FROM mytable
WHERE name = '{myName}'
```

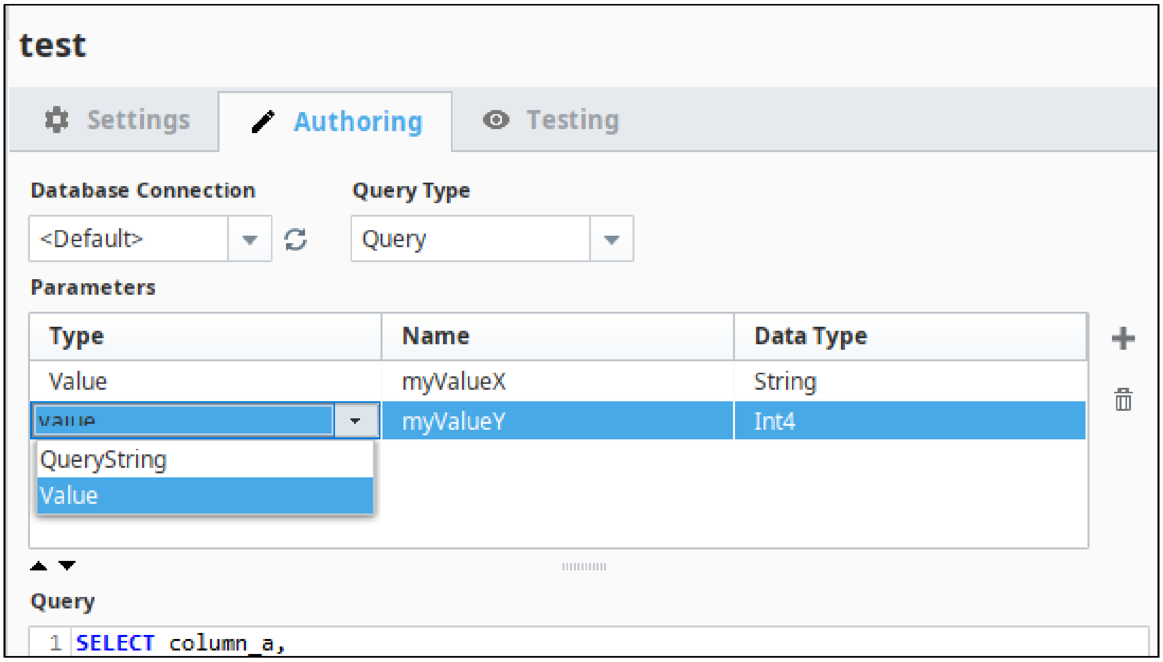
Database

Database type parameters cannot be created manually. Instead, it is automatically created when the **Database Connection** dropdown on the **Authoring** section is set to **<Parameter>**. Additionally, this parameter is not used in the body of the query. This type allows you to parameterize the database connection when the Named Query is called. This way the Named Query can run against multiple database connections specified by the resource that made the request.

Parameters while Authoring a Named Query

Creating Parameters

New parameters can be created in the Authoring section of a Named Query by clicking the Add ✚ icon next to the Parameters table.



Database Connection

<Default>

Query Type

Query

Parameters

Type	Name	Data Type
Value	myValueX	String
Value	myValueY	Int4

Query

```
SELECT column_a,
```

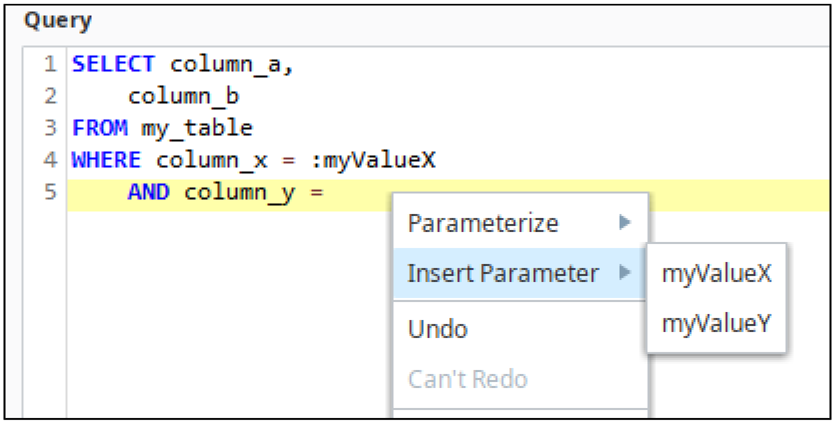
In Named Queries, Parameters are referenced by their name, so renaming the Parameter will require you to update it on any other resources that are using it.

Each parameter has three properties represented by different columns in the table. These may be edited by double-clicking on the cell you wish to modify:

- **Type**: Changes the type between Value and QueryString.
- **Name**: Determines the name of the parameter, and how it will appear in the query. Names are **not** case-sensitive and must be unique. Additionally, they may only use letters, numbers, dashes and underscores.
- **Data Type**: Specifies the datatype of the parameter. The Type of the parameter determines which data types are available. Note, that QueryStrings may only be configured as strings, where as Value-type parameters have more types available.

Using Parameters in the Query

Once created, parameters can be inserted into the Query field by **drag-and-drop** from the Parameter table onto the Query field, or by using the **right-click menu** in the Query field.



Query

```
SELECT column_a,
    column_b
FROM my_table
WHERE column_x = :myValueX
    AND column_y =
```

Additionally, the parameters may be typed in manually, but the correct syntax must be used.

Related Topics ...

- Named Query Bindings
- system.db.runNamedQuery
- Named Query Conversions
- Using Named Queries - Example

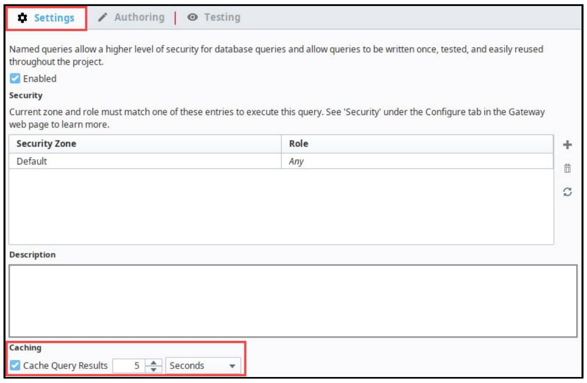
Named Query Caching

Overview

Named queries can opt-in to caching the results on the Gateway. This means if another request to the same Named Query comes in, the Gateway can return the cached result instead of having the database executing the query again. This will use more memory on the Gateway (to maintain the results), but could result in less queries running against the database.

Named Query caching is disabled by default, but can be enabled on the **Settings** section of each Named Query. If caching is enabled, the spinner and dropdown fields set the **lifespan** of the cache. Once the lifespan expires, the cache is invalidated. After the cache is invalidated, the next request for the Named Query will force the Gateway to re-execute the query and build a new cache.

Caching is especially useful for tables that are not updated often like recipe or inventory tables. Tables that update often like historical storage tables are bad candidates.



Named queries allow a higher level of security for database queries and allow queries to be written once, tested, and easily reused throughout the project.

☑ Enabled

Security

Current zone and role must match one of these entries to execute this query. See 'Security' under the Configure tab in the Gateway web page to learn more.

Security Zone	Role
Default	*Any*

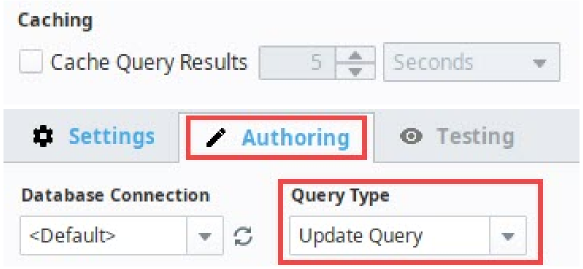
Description

Caching

☑ Cache Query Results 5 Seconds

When Caching is Disabled

Update queries are not allowed to cache their results. With UPDATE or DELETE statements, the work will already have been performed after the first execution, and INSERT statements typically utilize different parameters each execution. Thus if the Caching setting is disabled, then this typically means that the **Query Type** in the **Authoring** tab is set to an **Update Query**.



Caching

□ Cache Query Results 5 Seconds

Database Connection <Default>

Query Type Update Query

Scenario

Consider the following:

- A Named Query is created. This runs a query to fetch data from an arbitrary table.
- A component in a project is configured to call the Named Query on a window.
- **Client_A** navigates to the window, which triggers a request for the Named Query be executed.
- Several seconds later, **Client_B** opens the same window, and needs the same results.

In this scenario, if caching was enabled on the Named Query, then **Client_B** would not cause another query execution (assuming both **A** and **B** passed the same values to the parameters). This would result in less network traffic between the Gateway and the Database, and less work for the Database. An example with two clients isn't exciting, but the same scenario with ~50 clients would mean a huge potential performance boost, especially if those requests were polling at a 5 second rate, and the cache period was configured to 5 seconds.

Considerations

The following are considerations that should determine whether or not caching the results of a Named Query is helpful.

Gateway Memory

Each time the Named Query is called, if the arguments passed differ from those used to create the current cache, a new cache is created. This means Named Queries that are frequently called with varying parameter values will create multiple caches. If the results are large datasets, this can result in a large amount of the Gateway Memory being tied up maintaining these caches. In this scenario, you will want to monitor the memory usage of the Gateway.

This is especially important to consider when dealing with queries that accept a timestamp parameter that uses an expression like now() that will return time to the current second. If 5 clients are opened with each one 1 second apart, they will all have different timestamps and create separate cache entries.

How often the Database Values are Updated

Once a cache is created, the Named Query will not look for any changes made to the database table until the query next executes. This means changes, such as adding a new row to the table, will not appear in the Named Query's results until the cache is invalidated. Data that changes often may not be a good candidate for caching. In this case, the lifespan of the cache should be set to a short amount of time, depending on how often the data may get updated.

How often the Named Query is Executed

If a Named Query is called frequently and there are not many updates to the table data, then there can be a huge performance benefit to caching the results. Fewer calls to the database result in less network traffic and better overall performance.

Cached Query Updates and Designer Values

Cached query results have some notable interactions in the Designer. Specifically:

- **Named Queries results executed in the Designer never cache**: This is because the Designer uses the live version of the query in the Designer instead of the saved Gateway version. This also prevents bad results being stored in the cache, and then appearing in the Clients.
- **Making changes to a Named Query and saving will invalidate all caches for that Named Query**: When a change is made to a Named Query and saved, the new query is pushed to the Gateway, which means all current caches are immediately outdated.

Scripting Functions to Clear Cache from a Named Query

You can also clear the Named Query cache using the following scripting functions.

- system.db.clearNamedQueryCache - Clears the cache of a single Named Query.
- system.db.clearAllNamedQueryCaches - Clears the caches of all Named Queries.

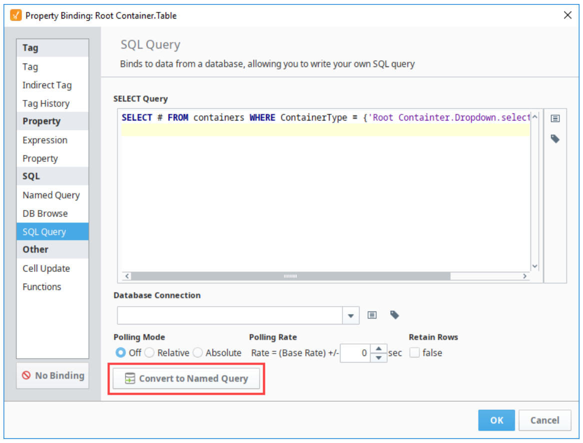
Related Topics ...

- Named Query Conversions
- Using Named Queries - Example

Named Query Conversions

Converting SQL Queries to Named Queries

If you have a SQL Query binding on a component, it is easy to convert it to a Named Query. Simply open up the binding and click the **Convert to Named Query** button.



SELECT Query

```
SELECT # FROM containers WHERE ContainerType = {'Root Containter.Dropdown.select
```

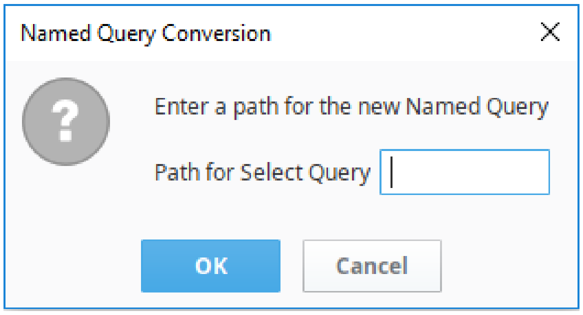
Database Connection

Polling Mode ● Off ○ Relative ○ Absolute

Polling Rate Rate = (Base Rate) +/- 0 sec

Retain Rows □ false

A popup appears where you can enter a path to the new Named Query.



Enter a path for the new Named Query

Path for Select Query

Converting with an Update Query

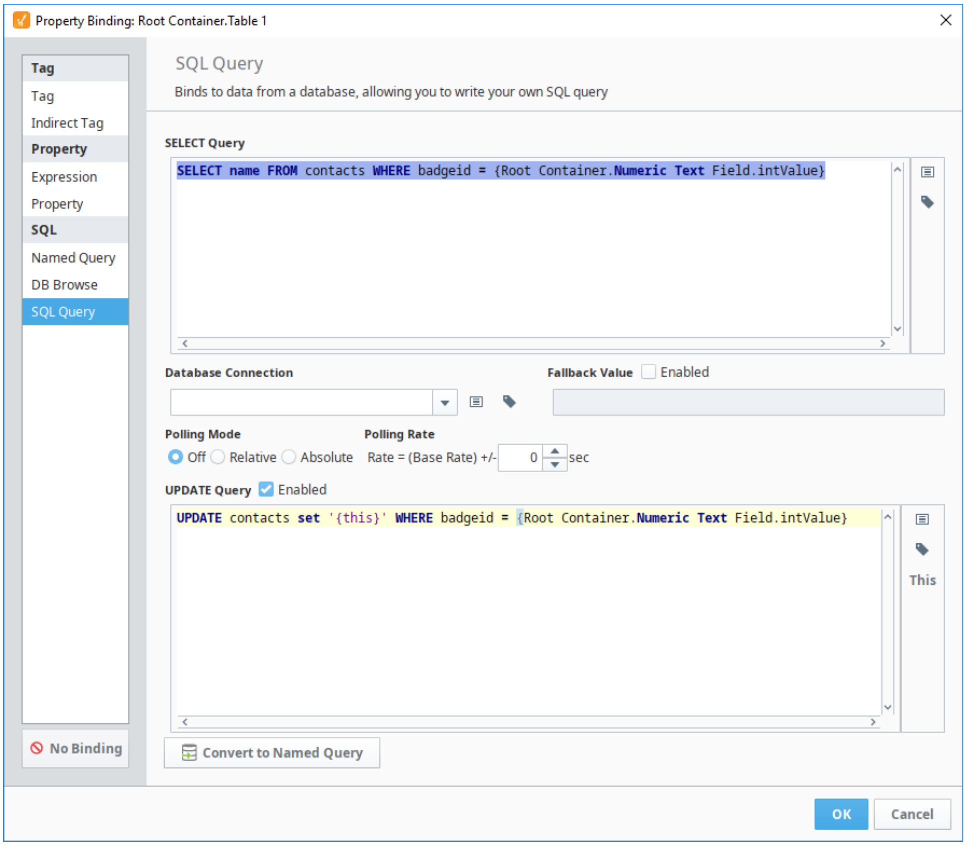
If your SQL Query binding is on an component property whose value can be updated and written back to the database (for example, input components and button components), you have the option to create two Named Queries: one for the Select Query in the binding and a second for the Update Query.

1. Enter the Select Query.
2. Click the Enabled checkbox for the Update Query.
3. Enter the Update Query.
4. Click OK.

On this page ...

Named Query Conversion Example

Watch the Video



SQL Query

Binds to data from a database, allowing you to write your own SQL query

SELECT Query

```
SELECT name FROM contacts WHERE badgeid = {Root Container.Numeric Text Field.intValue}
```

Database Connection

Fallback Value □ Enabled

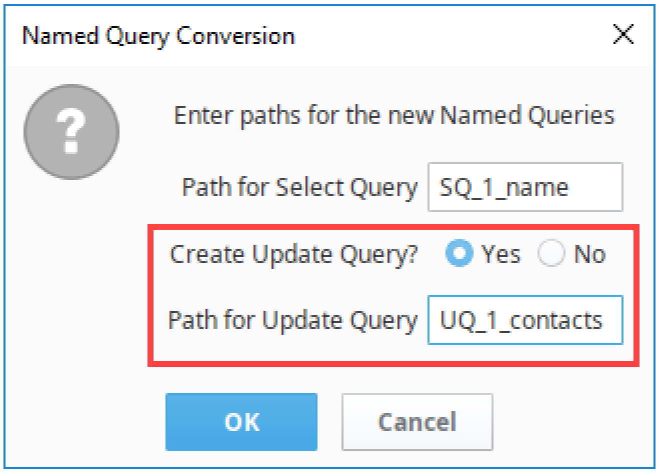
Polling Mode ● Off ○ Relative ○ Absolute

Polling Rate Rate = (Base Rate) +/- 0 sec

UPDATE Query ☑ Enabled

```
UPDATE contacts set '{this}' WHERE badgeid = {Root Container.Numeric Text Field.intValue}
```

5. In the popup, enter the path for the Select Query.
6. Click the Yes button next to Create Update Query,
7. Enter the path for the Update Query,
8. Click Save.



Enter paths for the new Named Queries

Path for Select Query: SQ_1_name

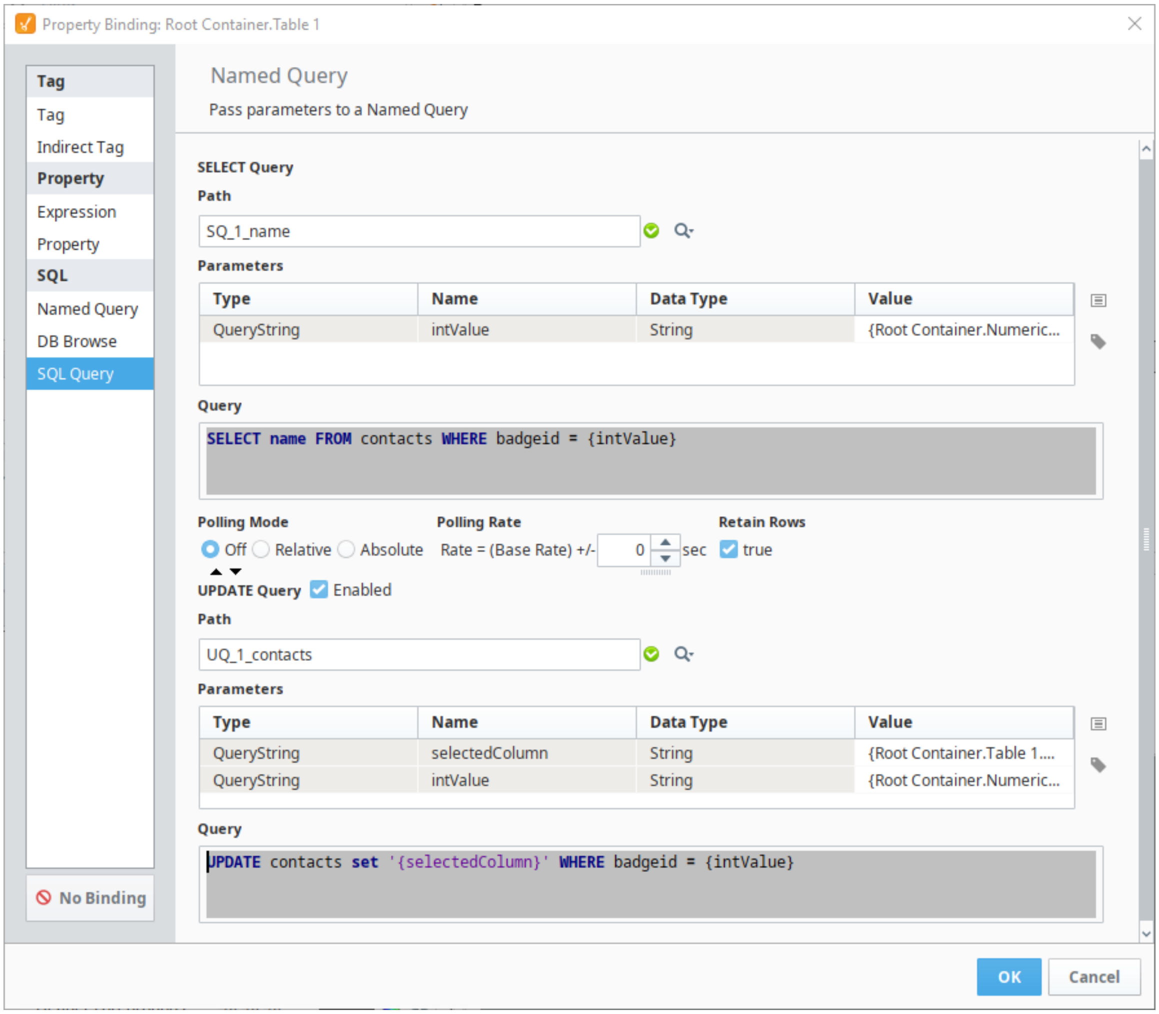
Create Update Query? ● Yes ○ No

Path for Update Query: UQ_1_contacts

Warning for 7.9.5 and Prior

When converting a SQL Query Binding that contains an UPDATE Query in 7.9.5 and prior, an Update query will not be generated. Thus, it was possible to lose the Update Query upon conversion.
If using these versions of Ignition, it is highly recommended to manually create the Update Named Query before pressing the Conversion button. As of 7.9.6, this is no longer an issue, and the **Convert to Named Query** button may freely be used.

After confirming the conversion, the Window will show the query/queries. Click the **OK** button to save this change.



Named Query

Pass parameters to a Named Query

SELECT Query

Path: SQ_1_name

Parameters

Type	Name	Data Type	Value
QueryString	intValue	String	{Root Container.Numeric...

Query

```
SELECT name FROM contacts WHERE badgeid = {intValue}
```

Polling Mode: ● Off ○ Relative ○ Absolute

Polling Rate: Rate = (Base Rate) +/- 0 sec

Retain Rows: ☑ true

UPDATE Query ☑ Enabled

Path: UQ_1_contacts

Parameters

Type	Name	Data Type	Value
QueryString	selectedColumn	String	{Root Container.Table 1....
QueryString	intValue	String	{Root Container.Numeric...

Query

```
UPDATE contacts set '{selectedColumn}' WHERE badgeid = {intValue}
```

When converting in this manner, all parameters will be created at **QueryStrings**. It is highly recommended that you modify your new Named Queries so that these values become **Value Parameters** instead, as mentioned later on this page.

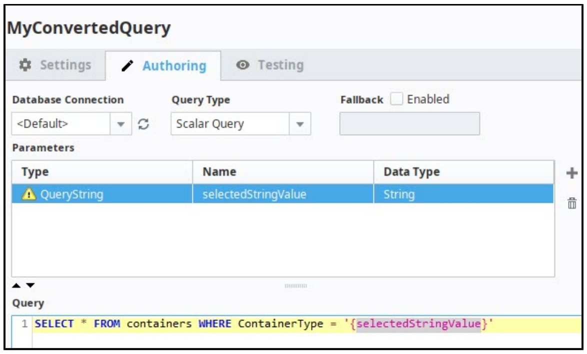
Modifying Converted Query to Use Parameters

When converting a query to a Named Query, it is **strongly** recommended that you go back into the query and convert the parameter from a **QueryString** type to a **Parameter** type. To do this, you first need to change all QueryString type parameters to the Parameter type. This is as simple as selecting Parameter from the dropdown under Type.

Now that your parameter(s) types have been modified, the second thing you need to do is modify syntax in the query to use the parameter(s). The Parameter and QueryString types are referenced differently in the query, so you will need to ensure that you modify how the parameter is referenced. In addition because the Parameter type works like a prepared statement, it does not need any quotation marks around any string type parameters like a QueryString would, so all quotation marks around parameters should be removed from the query. See the images below for an example.

From this:

To this:



MyConvertedQuery

Database Connection: <Default>

Query Type: Scalar Query

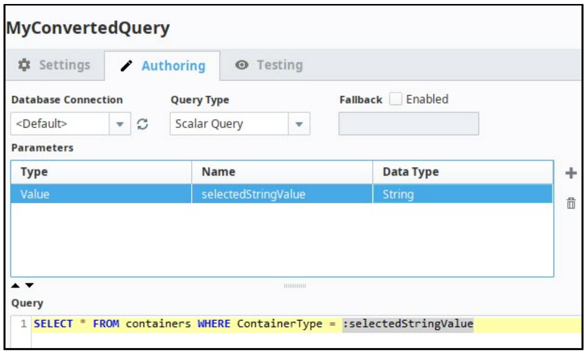
Fallback □ Enabled

Parameters

Type	Name	Data Type
QueryString	selectedStringValue	String

Query

```
SELECT * FROM containers WHERE ContainerType = '{selectedStringValue}'
```



MyConvertedQuery

Database Connection: <Default>

Query Type: Scalar Query

Fallback □ Enabled

Parameters

Type	Name	Data Type
Value	selectedStringValue	String

Query

```
SELECT * FROM containers WHERE ContainerType = :selectedStringValue
```

Once the syntax in the query has been modified, test it out in the **Testing** section at the top to make sure everything works correctly. If so, then you have successfully converted to a Named Query.

Related Topics ...

- Named Queries
- Named Query Bindings
- SQL Query Bindings in Vision

Using Named Queries - Example

A Named Query Example

Here we will go over the steps necessary to put together a basic named query. This example will create a Named Query that uses a single parameter to run a select query, and then add a Table to a window and create a binding that uses our new Named Query.

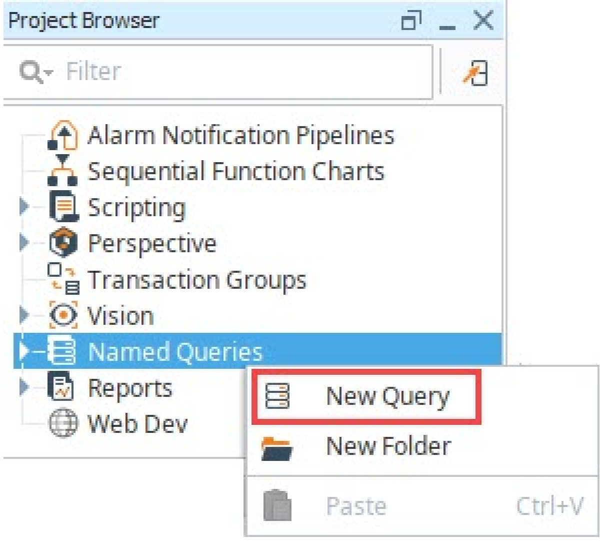
We are going to be querying a table in the database that holds information about products stored in storage bays. The database table is named "containers" and has the following structure.

id	StorageBay	ContainerType	ItemName	Weight	Time
1	1	Jug	Vanilla	25.2	2017-06-25 15:58:47
6	1	Mason Jar	Chocolate	12.3	2017-06-26 16:05:27
18	2	Tray	Swiss Cheese	88.8	2017-06-25 01:21:31
22	3	Tray	Cheddar Cheese	54.7	2017-06-25 03:52:16
23	3	Basket	Strawberry	36.8	2017-06-25 15:56:03
31	3	Jug	Whole Milk	80.1	2017-06-27 09:51:31
32	3	Jug	Fat Free Milk	76.9	2017-06-27 09:52:52

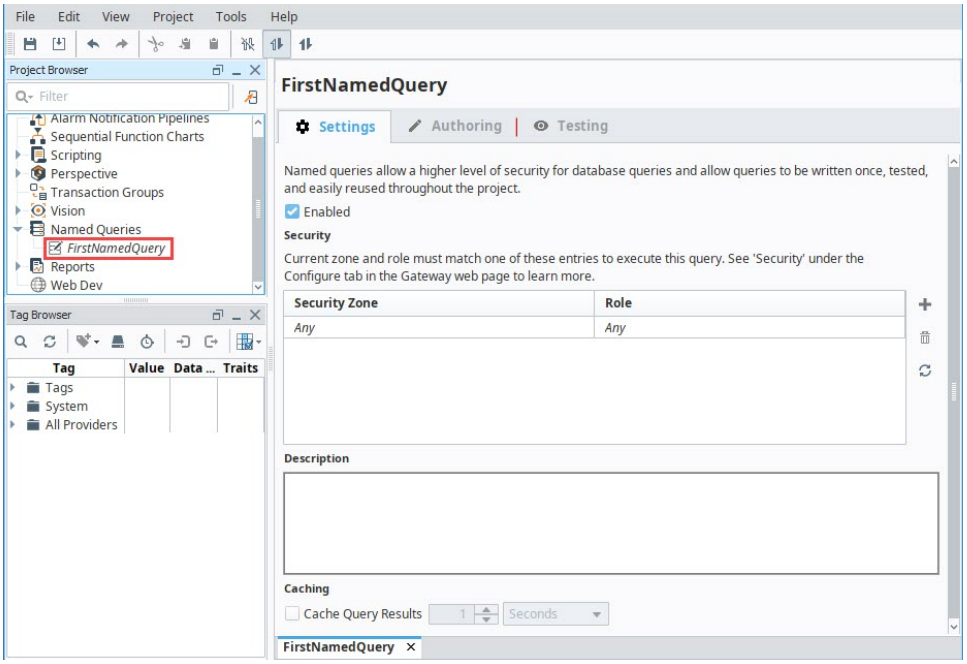
If you want to follow along with the example, feel free to make a database table that looks similar to this one and add as many rows of data as you would like, otherwise, you can use your own and substitute in the proper column names from your database.

Creating a Named Query and Adding Security

1. Start by opening up the Designer and loading a project.
2. Locate the Named Query section of the project browser, right click on it, and select the **New Query** option.



3. You should now have a fresh Named Query that you can rename whatever you want. We used **FirstNamedQuery** in the example.



Named queries allow a higher level of security for database queries and allow queries to be written once, tested, and easily reused throughout the project.

☑ Enabled

Security

Current zone and role must match one of these entries to execute this query. See 'Security' under the Configure tab in the Gateway web page to learn more.

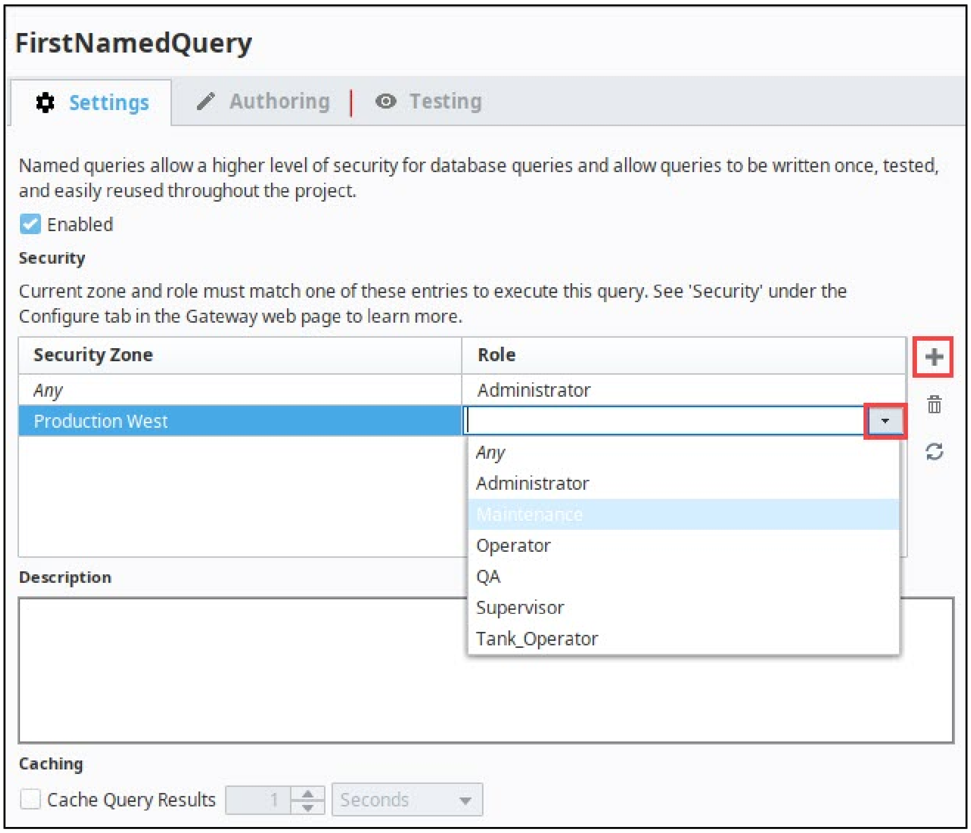
Security Zone	Role
Any	*Any*

Description

Caching

☐ Cache Query Results 1 Seconds

4. To setup security on the Named Query, set required Security Zone and Role combinations in the Security table of the Settings tab. You can leave this blank if you don't have roles or zones set up yet.
5. If multiple security combinations are required, use the Add + icon to add additional rows.



Named queries allow a higher level of security for database queries and allow queries to be written once, tested, and easily reused throughout the project.

☑ Enabled

Security

Current zone and role must match one of these entries to execute this query. See 'Security' under the Configure tab in the Gateway web page to learn more.

Security Zone	Role
Any	Administrator
Production West	

Description

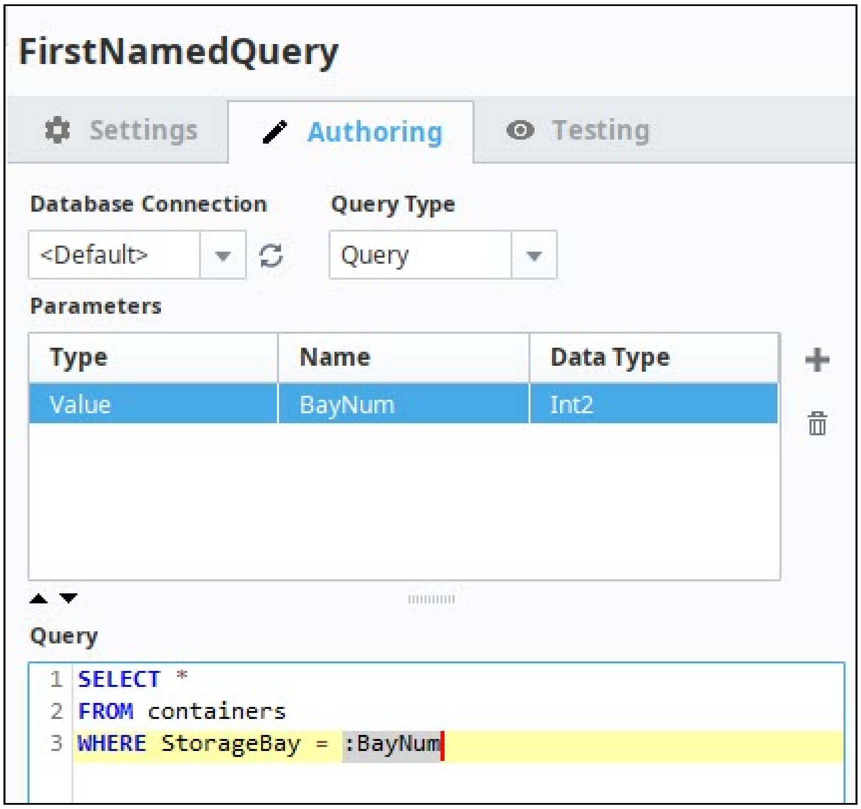
Caching

☐ Cache Query Results 1 Seconds

Building the Query

1. Click on the **Authoring** tab. Here is where we do most of the work.
2. Under **Database Connection**, we need to select a database connection that this named query will use. We selected **<Default>**.

3. For the **Query Type**, we can decide what type of query this will be. For this example, we are running a select query that will return a dataset, so we chose **Query**.
4. In the Parameters section, we can decide on a list of parameters that will be used in this query. This query is fairly simple and will only use a single parameter.
   Click the Add ✚ icon to add a new parameter and set the following values:
   Type: Parameter
   Name: BayNum
   Data Type: Integer
5. The Query section below is where we construct our query using the Table Browser.
   a. Right click on the containers table in the Table Browser on the right, and click on **Create SELECT Statement**. This will populate our query field with a basic select all statement.
   b. Type into the **Query** field and add the following WHERE clause: "**WHERE StorageBay =** "
   c. Now drag the **BayNum** parameter from the Parameters table to the end of the query you just typed. Notice "**:BayNum**" will be added at the end of the query.



FirstNamedQuery

Database Connection

<Default>

Query Type

Query

Parameters

Type	Name	Data Type
Value	BayNum	Int2

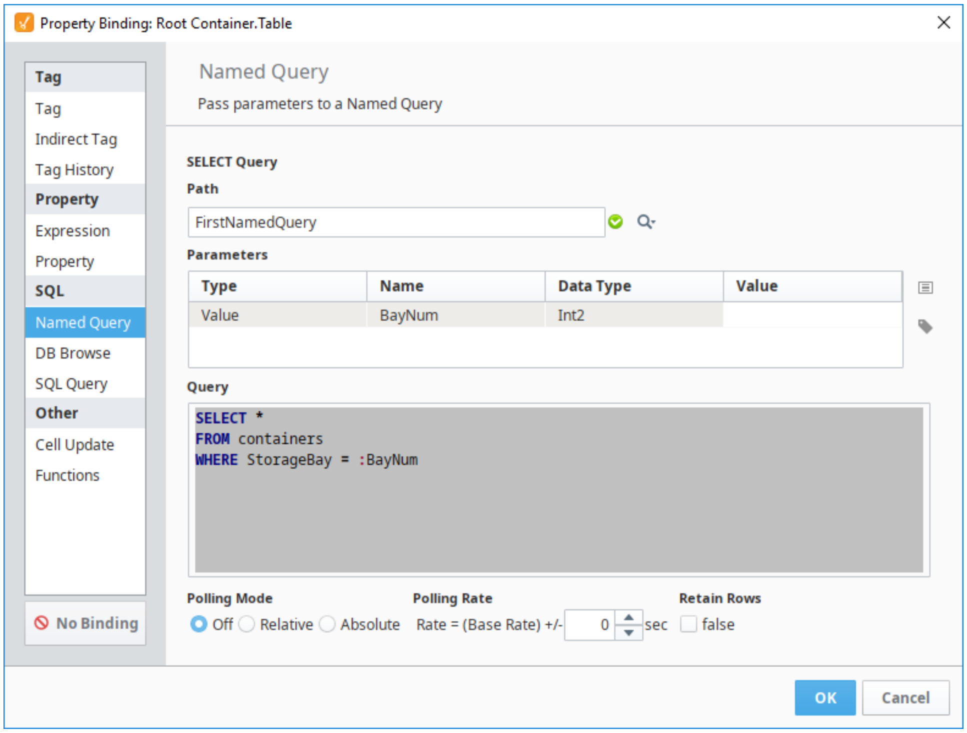
Query

```
SELECT *
FROM containers
WHERE StorageBay = :BayNum
```

Using the Query

1. In the Project Browser, create or open a Main Window.
2. Drag a **Dropdown List** component onto the window.
3. The Dropdown List is where we will be able to select a Bay Number to use as our Named Query's BayNum Parameter. Use the Data set Viewer to set the **Data** property of the Dropdown to look like this:

Value	Label
1	Bay 1
2	Bay 2
3	Bay 3



Named Query

Pass parameters to a Named Query

SELECT Query

Path

FirstNamedQuery

Parameters

Type	Name	Data Type	Value
Value	BayNum	Int2	

Query

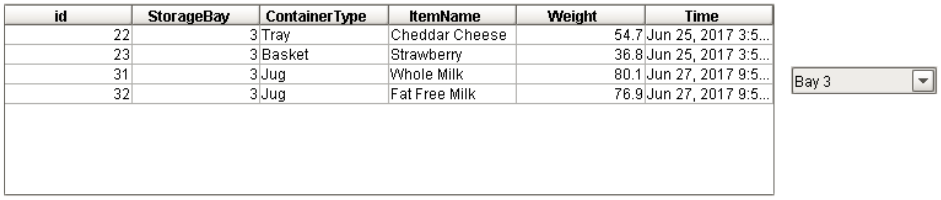
```
SELECT *
FROM containers
WHERE StorageBay = :BayNum
```

Polling Mode: ● Off ○ Relative ○ Absolute

Polling Rate: Rate = (Base Rate) +/- 0 sec

Retain Rows: □ false

4. Now drag a **Table** component onto the window. We can setup a Named Query binding on the Data property.
5. Click on the **Binding** 🔗 icon for the **data** property and select the **Named Query** binding type.
6. Set the Path property by clicking on the **Select Resource Path** 🔍 icon and selecting your new Named Query from the list. For this example, it is **FirstNamedQuery**. Alternately, you can type the name in.
7. Highlight your **BayNum** Parameter (in the Parameters table) and click on the Insert Property ☰ icon. We want to select the **Selected Value** property of our Dropdown List.
8. Finally, we want to ensure the Polling Mode is set to **Off**. This means the query will not run continuously, but will only run when it changes such as when a new bay is selected from the dropdown.
9. Click the **OK** button to save your binding, and put your Designer into **Preview Mode** ▶ to test it. We can then make a selection (i.e., Bay 3) with our Dropdown, and see the table populate with data.



id	StorageBay	ContainerType	ItemName	Weight	Time
22	3	Tray	Cheddar Cheese	54.7	Jun 25, 2017 3:5...
23	3	Basket	Strawberry	36.8	Jun 25, 2017 3:5...
31	3	Jug	Whole Milk	80.1	Jun 27, 2017 9:5...
32	3	Jug	Fat Free Milk	76.9	Jun 27, 2017 9:5...

Bay 3

Related Topics ...

- Named Queries

Queries in Scripting

On this page

...

Overview

In addition to using a binding, queries can be called from a Python script. This typically involves using one of Ignition's built-in scripting functions. This page presents several approaches to interacting with a database from a Python Script. See the Scripting section for more information about using Python.

Named Queries

Named Queries may be called from a Python script using the system.db.runNamedQuery function. Named Queries can execute as a Prepared Statement when not utilizing Query String parameters, so they can offer a secure approach to calling a query from a script.

Additionally, Named Queries are incredibly easy to call from a script, since you only need to know their name, and don't have to bother with SQL syntax in the script. This is another way to protect your database from other users in the Designer, as you can Protect Named Queries.

Python - Calling a Named Query

```
namedQuery = "Add New Order"
parameters = {"accountId":123, "productName":"Bananas"}

system.db.runNamedQuery(namedQuery, parameters)
```

For more information, refer to the Named Queries section.

Prepared Statements vs Standard Statements

A prepared statement is a feature in a database that executes a parameterized query. One of the main reasons to utilize a prepared statement is because they are resilient to SQL Injection Attacks. Because of this, we **highly recommend** you utilize Prepared Statements over Standard Statements, but Named Queries are the most secure. Especially in cases where the users can type values that will be included in the query.

Prepared Statements typically involve passing two pieces of information to the database:

- A query string that contains placeholders to represent where arguments will be passed in later. These are represented in the query as question mark characters ("?")
- A series of arguments to replace the placeholder characters in the static query.

Standard Statements vs Prepared Statements

Typical SQL insert queries look like the following:

SQL - Standard Statement

```
INSERT INTO orders (account_id, product_name) VALUES (123, 'Bananas')
```

A Prepared Statement instead looks like the following. Note, that the placeholders do not require quotation marks even when a string will be passed in.

SQL - Prepared Statement

```
INSERT INTO orders (account_id, product_name) VALUES (?, ?)
```

How do I Call a Prepared Statement?

Prepared Statements can be called from a script using specific functions. Typically, they contain "Prep" in the name such as system.db.runPrepQuery, or system.db.runPrepUpdate. When in doubt, take a look at the sub pages in the system.db section.

There are typically two required parameters with these functions: a string literal that represents the query, and a list of parameters.

Python - Calling an Insert Query as a Prepared Statement

```
query = "INSERT INTO orders (account_id, product_name) VALUES (?, ?)"
args = [123, "Bananas"]
system.db.runPrepUpdate(query, args)
```

Notable Prepared Statement Functions

- system.db.runPrepQuery: Should be used when SELECT statements are used, as it returns a dataset containing the results. This function is effectively "read-only", as it does not manipulate data in the database.
- system.db.runPrepUpdate: Should be used when manipulating the database in some way. When executing a statement that utilize INSERT, UPDATE, or DELETE, the runPrepUpdate function should be called. Note that it does return the number of rows affected, so the return value can be used to keep track of how much of an impact the query had.
- system.db.runScalarPrepQuery: Like runPrepQuery above, but only returns the first column of the first row: i.e. a single value is returned instead of a full dataset. This is useful when using an aggregate function of some sort to return a count or total, as it saves your script the work of extracting the value from the full dataset that runPrepQuery normally returns.

Standard Statements

Queries can be called as a Standard Statement (a statement that that isn't a Prepared Statement) by using the system.db.runQuery and system.db.runUpdateQuery functions. However, these are susceptible to **SQL Injection attacks**, and should be avoided where possible: especially when users have access to a keyboard and can directly type values that will be used in the query.

Calling a Standard Statement involves building the entire query as a single string, and passing the string on to our Standard Statement functions.

Python - Calling an Insert Query as a Standard Statement

```
query = "INSERT INTO orders (account_id, product_name) VALUES (%i, '%s')" % (123, "Bananas")
system.db.run(query)
```

Notable Standard Statement Functions

- system.db.runQuery: Executes a SELECT statement, returning a result set as a dataset.
- system.db.runUpdateQuery: Executes a statement that manipulates the database in someway. Should be used with INSERT, UPDATE, and DELETE statements.
- system.db.runScalarPrepQuery: Similar to runQuery, except only a single value is returned: the first column of the first row. Generally used in conjunction with SELECT statements that contain an aggregate function.

Stored Procedures

If your database administrator has already configured Stored Procedures for you to use, then they can easily be called from a Python Script. Using Stored Procedures in a script typically involves two main steps:

1. A SProcCall object is created with the system.db.createSProcCall function. The SProcCall object contains several functions that can be used to register parameters, and access the results set returned by the Stored Procedure after it has been executed.
2. The system.db.execSProcCall function must be used to execute the Stored Procedure.

Python - Creating and Executing a Stored Procedure

```
# Create a SProcCall object, which will be used to configure parameters on the Stored Procedure, and
then executed.
myCall = system.db.createSProcCall("insert_new_order")

# Register parameters on the SProcCall object.
myCall.registerInParam(1, system.db.INTEGER, 123)
myCall.registerInParam(2, system.db.VARCHAR, "Bananas")

# Execute the Stored Procedure.
system.db.execSProcCall(myCall)
```

Take a look at the SQL Stored Procedures page for more details.

Transactions

A SQL Transaction can also be executed from a script. For the unfamiliar, a Transaction is a batch of statements that will be executed together, and either succeed or fail as a group. Note, that the statements executed in the Transaction are not visible by other connections in the database until you commit them.

Transactions typically involve several steps:

1. Call system.db.beginTransaction. This returns a **transaction identifier** that can be used with other statements. Using this identifier is how you specify that a statement should be included in the transaction.
2. Start calling other statements with other functions, such as system.db.runPrepUpdate. The function's "tx" parameter will be passed the **transaction identifier**.
3. Commit or Rollback the transaction. Use system.db.commitTransaction to commit, and system.db.rollbackTransaction to rollback. These options are essentially the same as applying or canceling the results of the queries. Commiting will make the updated results available to other database connections.
4. Close the Transaction once you're done, which can be accomplished with the system.db.closeTransaction function. This invalidates the **transaction identifier**.

Python - The SQL Transaction Workflow

```
# 1) Begin the transaction. This returns a transaction identifier that we can use with other statements.
transactionId = system.db.beginTransaction(timeout = 5000)

# 2) Now we can execute statements. Because we want these to run as part of the transaction, we need to
include our identifier.
query = "INSERT INTO orders (account_id, product_name) VALUES (?, ?)"
args = [123, "Bananas"]
system.db.runPrepUpdate(query, args, tx = transactionId)

# 3) We can continue to add statements, but in this case we'll commit them. We could instead rollback if
there was an issue with our previous statement.
system.db.commitTransactions(transactionId)

# 4) We're done, so close the Transaction.
system.db.closeTransaction(transactionId)
```

Related Topics ...

- Writing Basic SQL Queries
- system.db.runPrepUpdate
- Named Queries
- SQL Stored Procedures

Common SQL Tasks

This section contains examples of items we identified as "common" tasks. When first starting out with SQL in Ignition, many users are looking for examples in order to grasp concepts, or examples to possibly use in their projects. Additionally, this section aims to demystify some of the more complex or abstract tasks that our users may encounter when working with SQL in Ignition.

The examples in this section document several types of SQL tasks that may also touch many other areas of Ignition. While these examples are typically focused on a single concept or end result, they can easily be expanded or modified. In essence, they serve as a great starting point for users new to Ignition, as well as experienced users that need to get acquainted with a new or unfamiliar feature.

Below is a list of common SQL tasks related to using SQL in Ignition.

On this page ...

Filter Data in a Table

The Filter Data in a Table section explains how using a simple Dropdown box and making some modifications to the SQL query can allow users to dynamically filter data coming into the table to only show a specific subset of data, or all of the data. This section also includes variations of the SQL query, thus returning different filtered results to the table.

Add Data to a Database

A common method of adding data to a database from within a client is using a Button component that executes an SQL query. This section provides an SQL script showing how to collect relevant properties and insert them into a database, as well as how to setup your components on a window.

Edit Data in a Database

Editing data in a database is very similar to how we add data to a database. Users can edit data within a table on the screen in realtime, and the changes are then pushed back to the database table. This section goes over the different ways to edit data in the database depending if you're using a Table component or Power Table component.

Refreshing SQL Data on a Component

Sometimes, it may be unnecessary to have a table constantly updating and requerying the database for data. By using the system.db.refresh function, we can turn Polling "Off" on our query, and have a button that allows us to manually update the table with new data when the Button is pressed. Alternatively, we can add the system.db.refresh function at the end of a script to refresh the newly entered data automatically without pressing a button.

Editing Multi-Selected Rows from Table

Tables have the ability to select and edit multiple rows at a time. Oftentimes, this can be used to manipulate multiple rows simultaneously, such as deleting all of the selected rows at once.

Storing PDF Files in a Database

The database is a powerful tool and can allow you to store files such as PDFs in a database table. The Storing PDF Files in a Database section explores how we can take a PDF file and store it in the database table, as well as how to pull it back out and display it in the Vision - PDF Viewer component.

Simple Database Editor

It may be beneficial to have a frontend to a database table built right into an Ignition project that allows you to control the data in the table without having to go into the database's built-in frontend, such as MySQL workbench. In the Simple Database Editor section, you will learn how to build a simple database table editor on a window that can add, edit, and delete data from the database.

Related Topics ...

- Writing SQL Queries
- Scripting Examples
- Common Reporting Tasks

In This Section ...

Filter Rows in a Table

Creating the Components

On this page

...

Filtering table data by using a Dropdown box is possible if the **Table** component's **Data** property is bound to a SQL query. We can create a dynamic WHERE clause that will allow us to select all the data or a specific subset of it. To implement this solution, you must have a **Table** component and a **Dropdown List** component on a window. With a Dropdown component, we can create an option for each way of filtering the data, each with a unique value associated with it. An additional "all" entry can also be added, with its own unique value.

We can then use the value of the Dropdown to drive the query. When you select a different option from the Dropdown component the binding gets re-evaluated on the table's data binding resulting in the query executing the WHERE clause with the new parameters. You may want to turn Polling Mode off on the tables Data property binding in order to limit the periodic querying of the database.

Here we can use a value of 1 to show only entries which match that filter column value.

Pseudocode - Filtering for Area 1

```
SELECT * FROM table WHERE filterColumn = 1
```

While the following query will return everything, but will still evaluate the query's WHERE clause.

Pseudocode - Return All Rows

```
SELECT * FROM table WHERE 0=0
```

Filter Data in Table

Watch the Video

In this case 0=0 will always evaluate as true, therefore, the query will return every row.

We can combine the logic of each WHERE with an OR into a single query, and substitute in our Dropdown value. The SQL Query binding would then look something like this:

SQL - Where Clause Combined with OR Condition

```
SELECT * FROM table WHERE filterColumn = {dropdown value} OR 0 = {dropdown value}
```

This way, the WHERE clause will only be true when the filter column matches with our selected option, or it will return all rows, if we setup our "all" option to have a value of 0.

Example - Filtering Data on an Area Number

We can put together an easy example on how this might with data that stores what machines are in which areas. Our data table should look something like this:

id	machine_name	area_number
1	Conveyor	1
2	Press	2
3	Tank	1
4	Packer	3
5	Loader	3
6	Oven 1	3
7	Oven 2	2
8	Wrapper	1
9	Mixer	3

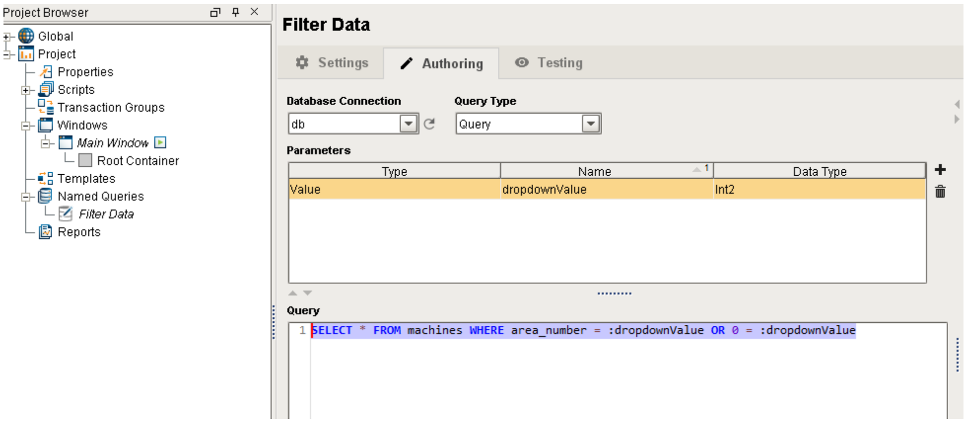
10	Cold Storage	2
11	Dryer	2

We can then put together the query and components necessary to get this working:

1. Create a new Named Query. Set up security to fit your needs, and name it appropriately. For more information on creating Named Queries, see Using Named Queries - Example.
    a. Create a single Value type Parameter that is an Int2 data type. I called mine dropdownValue
    b. Add in the combined query that we went over above, but use the machine table name and column names.

SQL - Selecting Values from a Table and Filtering on a Dropdown

```
SELECT * FROM machines WHERE area_number = :dropdownValue OR 0 = :dropdownValue
```



Database Connection: db

Query Type: Query

Parameters

Type	Name	Data Type
Value	dropdownValue	Int2

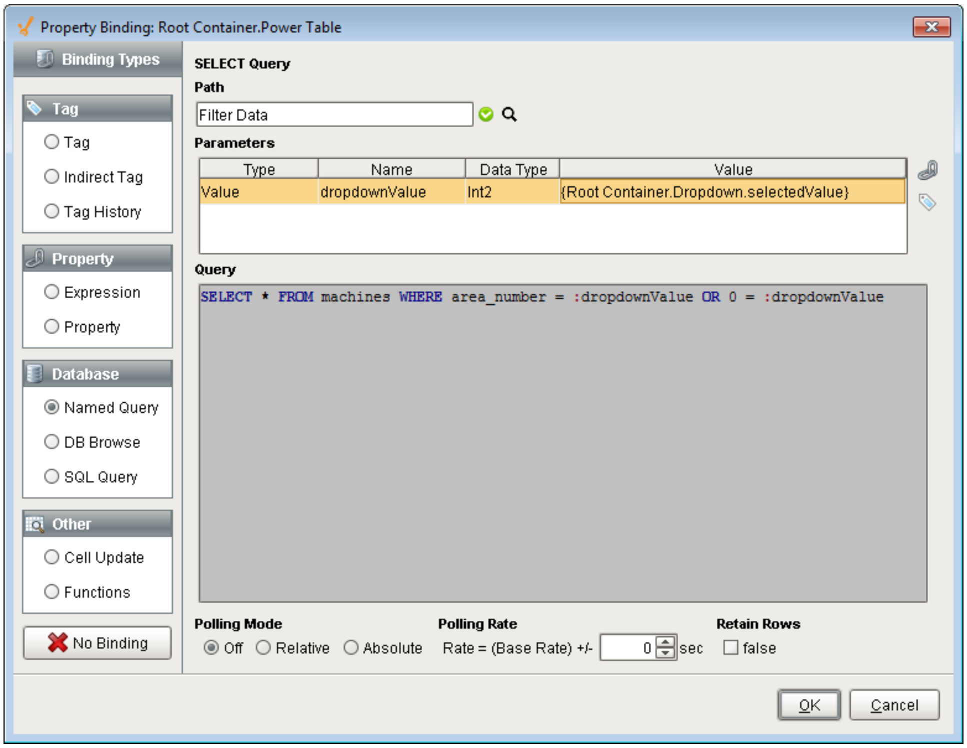
Query

```
SELECT * FROM machines WHERE area_number = :dropdownValue OR 0 = :dropdownValue
```

2. Create a new Main Window and add a Power Table component and a Dropdown component to the window.
    a. On the Dropdown's **Data** property, create a dataset that looks like this:

Value	Label
1	Area 1
2	Area 2
3	Area 3
0	All Areas

3. On the Power Table component, create a Named Query Binding on the **Data** property.
    a. Select the Named Query that was created in step 1.
    b. For the parameter, bind its value to the **Selected Value** property of the Dropdown component, and the click the **OK** button.



Tag: ○ Tag ○ Indirect Tag ○ Tag History

Property: ○ Expression ○ Property

Database: ● Named Query ○ DB Browse ○ SQL Query

Other: ○ Cell Update ○ Functions

SELECT Query

Path

Filter Data

Parameters

Type	Name	Data Type	Value
Value	dropdownValue	Int2	{Root Container.Dropdown.selectedValue}

Query

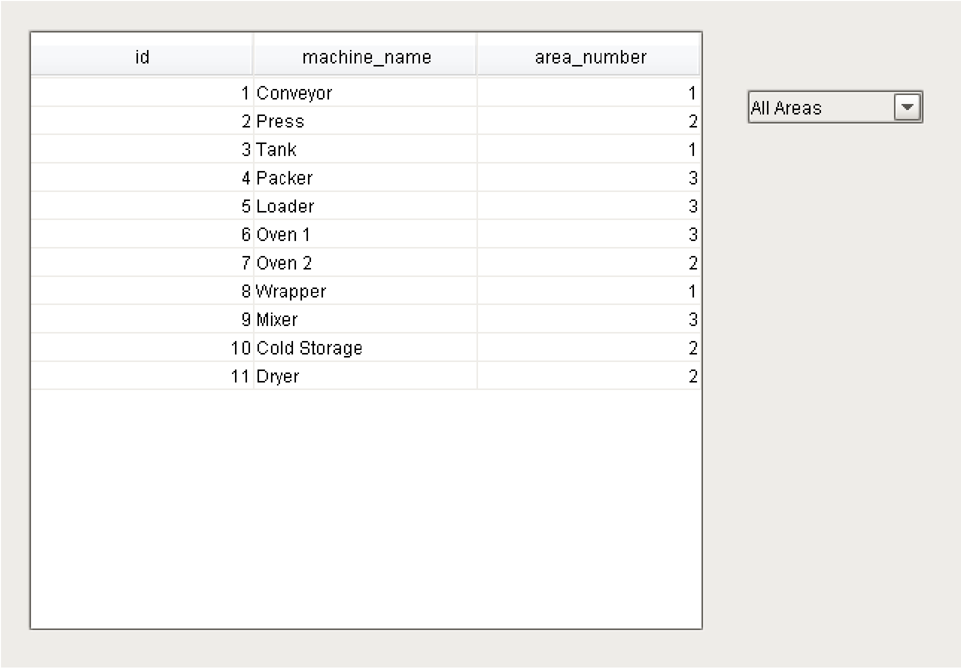
```
SELECT * FROM machines WHERE area_number = :dropdownValue OR 0 = :dropdownValue
```

Polling Mode ● Off ○ Relative ○ Absolute

Polling Rate Rate = (Base Rate) +/- 0 sec

Retain Rows □ false

4. Put the Designer into **Preview** mode and try out the Dropdown Filter. You will see that selecting a different value in the dropdown filters the data coming back so that only certain rows are shown.



id	machine_name	area_number
1	Conveyor	1
2	Press	2
3	Tank	1
4	Packer	3
5	Loader	3
6	Oven 1	3
7	Oven 2	2
8	Wrapper	1
9	Mixer	3
10	Cold Storage	2
11	Dryer	2

All Areas

Related Topics ...

- Writing SQL Queries

Inserting Data into a Database

Inserting Data on a Button Press

A common way to insert information into a database is to execute a SQL query after the user presses a Button. The button can run a script to collect the information it needs and then execute a SQL `INSERT` statement to push that data into the correct Database table. A script executed on the Button's **actionPerformed** event handler would collect the relevant properties and insert them into a database. The script would look like this:

Pseudocode - Collects Data and Insert into the Database

```
value1 = {component property reference}
value2 = {tag value}
value3 = {static value}

query = "INSERT INTO table (col1, col2, col3) VALUES (?,?,?)"
args = [value1, value2, value3]
system.db.runPrepUpdate(query, args)
```

Notice that the script isn't limited from taking values from any one place, grabbing tag values and property values and inserting them into a table. It is important to note that using the runPrepUpdate function will require that certain Client Permissions have been enabled.

Add Data to Database

Watch the Video

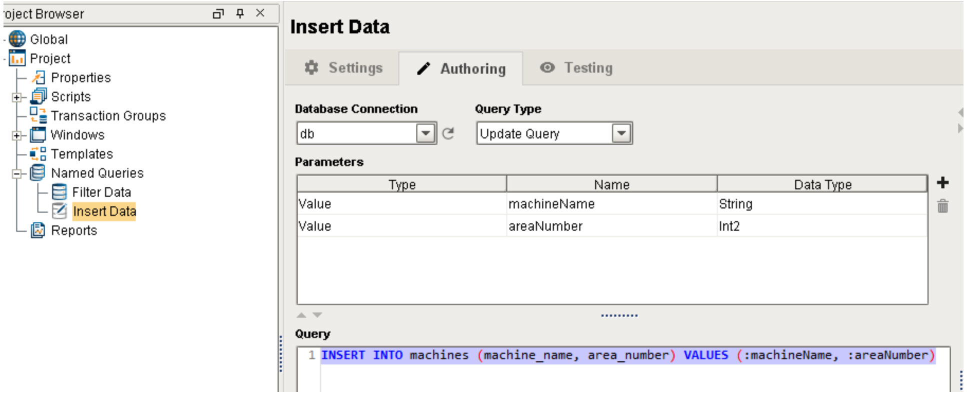
Example - Inserting Values

Say we have a table called machines in a database with three columns: an id column, a machine_name column, and an area_number column. We can build a query and a script that will insert the data into the database:

1. Create a new Named Query. Set up security to fit your needs and name it appropriately. For more information on creating Named Queries, see Using Named Queries - Example.
    a. Set the Query Type to Update Query.
    b. Create two Value type Parameters.
        i. The first will be a string data type and will be for the name of the machine so it can be called machineName.
        ii. The second will be an Int2 data type and will be for the area number so it can be called areaNumber.
    c. Create the insert query.

SQL - Inserting Values from Components on the Window

```
INSERT INTO machines (machine_name, area_number) VALUES (:machineName, :areaNumber)
```



Database Connection: db

Query Type: Update Query

Parameters

Type	Name	Data Type
Value	machineName	String
Value	areaNumber	Int2

Query

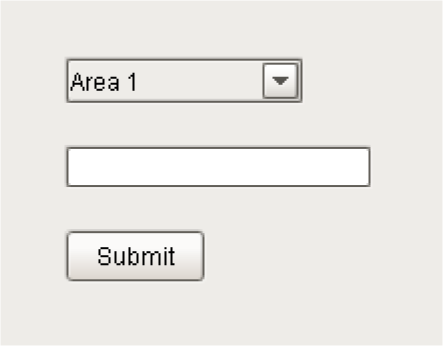
```
INSERT INTO machines (machine_name, area_number) VALUES (:machineName, :areaNumber)
```

2. On a Main Window, add a Button component, a Text Field component, and a Dropdown Component.

   a. On the Dropdown's Data property, create a dataset that looks like this:

Value	Label
1	Area 1
2	Area 2
3	Area 3

   b. On the Button's Text property, change the value to something like "Submit"



3. Right Click on the Button and select Scripting. Navigate to the Script Editor tab on the actionPerformed Event Handler.

   a. Copy in this script, which will pull in the value from the Dropdown and the Text Field and insert them into the table using the Named Query we built in step 1.

Python - Insert Values into a Database Table

```
# Grab the area number and machine name from the components we added to the window.
areaNum = event.source.parent.getComponent('Dropdown').selectedValue
machineName = event.source.parent.getComponent('Text Field').text

# A call to our Named Query, inserting the two parameters using dictionary syntax.
system.db.runNamedQuery("Insert Data", {"machineName":machineName, "areaNumber":areaNum})
```

4. Test it out by selecting an area, entering in a machine name, and clicking the submit button. You can check out the new rows using the Database Query Browser to select the values from that table.

Related Topics ...

- Simple Database Editor

Updating the Database through the Power Table

Updating Table Data

Table components can do more than show data from a database. A properly configured Table can make the data of the Table accessible to the Client and allow the user to edit the data in realtime. This page shows two examples using the Table and Power Table components editing data in a database in realtime.

Suppose your database had a Table like this:

id	UserName	FirstName	LastName	Notes
1	JS	John	Smith	Likes bikes
2	LJ	Luke	Johnson	Lives in town
3	PB	Peter	Burke	Enjoys cooking

INDUCTIVE UNIVERSITY

Edit Data in Database

Watch the Video

Using a Power Table Component

Let's create a script on a Power Table component and the **onCellEdited** extension function that will write to the cell in the database.

This function will not work using default Client Permission settings. The Legacy Database Access will need to be enabled for this to work.

1. Drag a Power Table component to your window, and bind the **Data** property of the Power Table using a SQL Query Binding Type and the query below.

   SQL - Selects All Records

   ```
   SELECT * FROM users
   ```

2. Right-click on the **Power Table** and select **Customizers > Table Customizer**.

3. Select the **Editable** checkbox for each of the columns that we want to edit (i.e., same as the above example). Note, the **id** column should not be editable. Click the **OK** button.

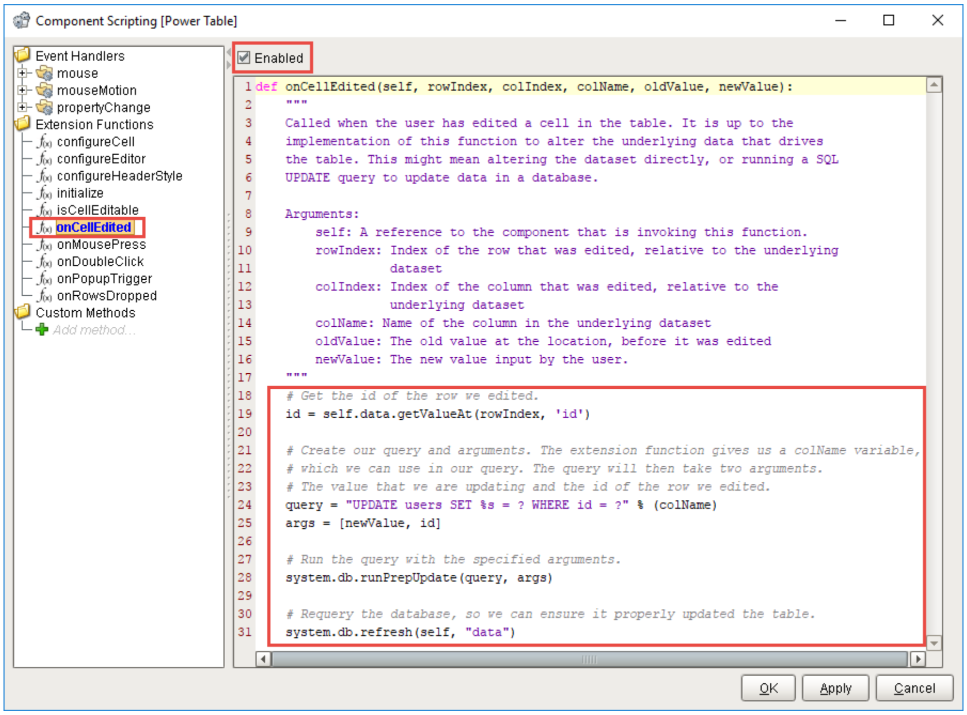
4. Right-click the **Power Table** and select **Scripting**.

5. Under the Extension Functions folder, select the **onCellEdited** function, and check the **Enable** box. We can put a script in here that will grab the id column value of the row that we modify, and then use that and the new value that we entered to update the Database table. The extension function makes this easy, because it provides variables for the column name of the row we are editing, the row number that was edited, as well as the new value. Paste the following script in the scripting area.

   Python - Writes to a Cell in the Database

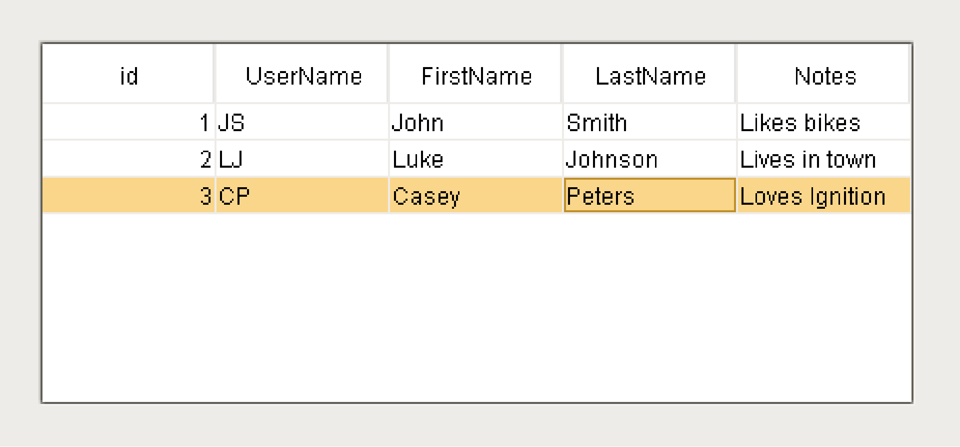
   ```
   # Get the id of the row we edited.
   id = self.data.getValueAt(rowIndex, 'id')

   # Create our query and arguments. The extension function gives us a colName variable,
   # which we can use in our query. The query will then take two arguments.
   ```

```
# The value that we are updating and the id of the row we edited.
query = "UPDATE users SET %s = ? WHERE id = ?" % (colName)
args = [newValue, id]

# Run the query with the specified arguments.
system.db.runPrepUpdate(query, args)

# Requery the database, so we can ensure it properly updated the table.
system.db.refresh(self, "data")
```



```
def onCellEdited(self, rowIndex, colIndex, colName, oldValue, newValue):
    """
    Called when the user has edited a cell in the table. It is up to the
    implementation of this function to alter the underlying data that drives
    the table. This might mean altering the dataset directly, or running a SQL
    UPDATE query to update data in a database.

    Arguments:
        self: A reference to the component that is invoking this function.
        rowIndex: Index of the row that was edited, relative to the underlying
                  dataset
        colIndex: Index of the column that was edited, relative to the
                  underlying dataset
        colName: Name of the column in the underlying dataset
        oldValue: The old value at the location, before it was edited
        newValue: The new value input by the user.
    """
    # Get the id of the row we edited.
    id = self.data.getValueAt(rowIndex, 'id')

    # Create our query and arguments. The extension function gives us a colName variable,
    # which we can use in our query. The query will then take two arguments.
    # The value that we are updating and the id of the row we edited.
    query = "UPDATE users SET %s = ? WHERE id = ?" % (colName)
    args = [newValue, id]

    # Run the query with the specified arguments.
    system.db.runPrepUpdate(query, args)

    # Requery the database, so we can ensure it properly updated the table.
    system.db.refresh(self, "data")
```

6. Now, let's test it out. Put the Designer in **Preview Mode**, select the cell you want to edit, hit enter to commit the change, or tab to the next cell to make additional edits.



id	UserName	FirstName	LastName	Notes
1	JS	John	Smith	Likes bikes
2	LJ	Luke	Johnson	Lives in town
3	CP	Casey	Peters	Loves Ignition

Using a Table Component

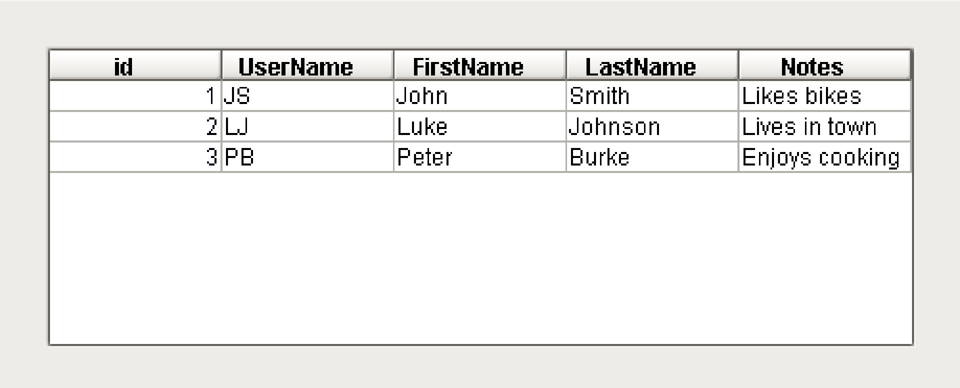
Let's create another script like the example above, but this time we'll use a Table component's **cellEdited** event that will write to the cell in the database when we update it on the component. The script will be slightly different.

This function will not work using default Client Permission settings. The Legacy Database Access will need to be enabled for this to work.

1. Drag a Table component to your window. With the Table component selected, bind the **Data** property of the Table using a SQL Query Binding Type and the query below.

SQL - Select All Records

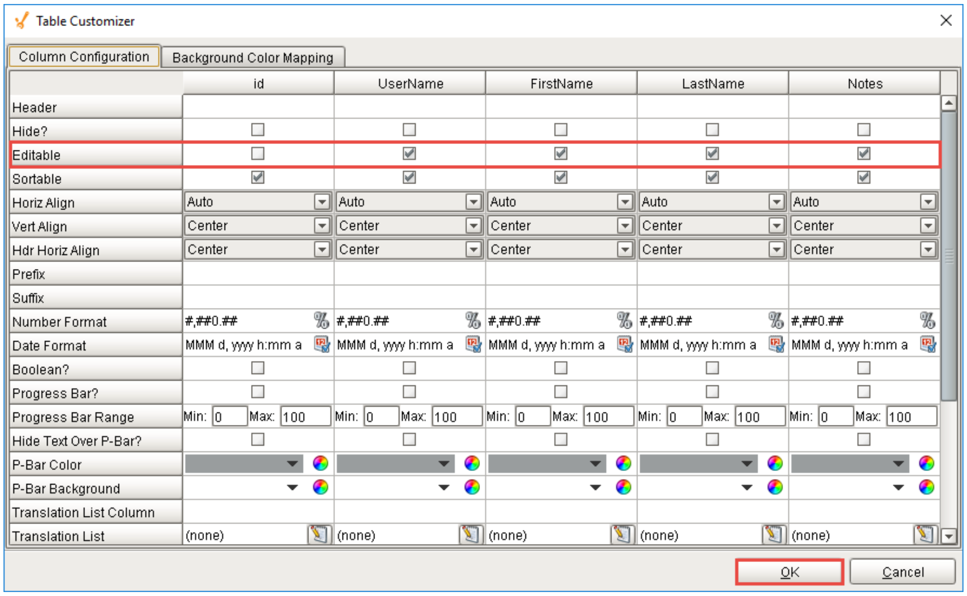
```
SELECT * FROM users
```



id	UserName	FirstName	LastName	Notes
1	JS	John	Smith	Likes bikes
2	LJ	Luke	Johnson	Lives in town
3	PB	Peter	Burke	Enjoys cooking

2. Right-click on the **Table** and select **Customizers > Table Customizer**.

3. Select the **Editable** checkbox for each of the columns that we will want to edit. Note, the **id** column should not be editable. Click the **OK** button.



	id	UserName	FirstName	LastName	Notes
Header					
Hide?	☐	☐	☐	☐	☐
Editable	☐	☑	☑	☑	☑
Sortable	☑	☑	☑	☑	☑
Horiz Align	Auto	Auto	Auto	Auto	Auto
Vert Align	Center	Center	Center	Center	Center
Hdr Horiz Align	Center	Center	Center	Center	Center
Prefix					
Suffix					
Number Format	#,##0.##	#,##0.##	#,##0.##	#,##0.##	#,##0.##
Date Format	MMM d, yyyy h:mm a	MMM d, yyyy h:mm a	MMM d, yyyy h:mm a	MMM d, yyyy h:mm a	MMM d, yyyy h:mm a
Boolean?	☐	☐	☐	☐	☐
Progress Bar?	☐	☐	☐	☐	☐
Progress Bar Range	Min: 0 Max: 100	Min: 0 Max: 100	Min: 0 Max: 100	Min: 0 Max: 100	Min: 0 Max: 100
Hide Text Over P-Bar?	☐	☐	☐	☐	☐
P-Bar Color					
P-Bar Background					
Translation List Column					
Translation List	(none)	(none)	(none)	(none)	(none)

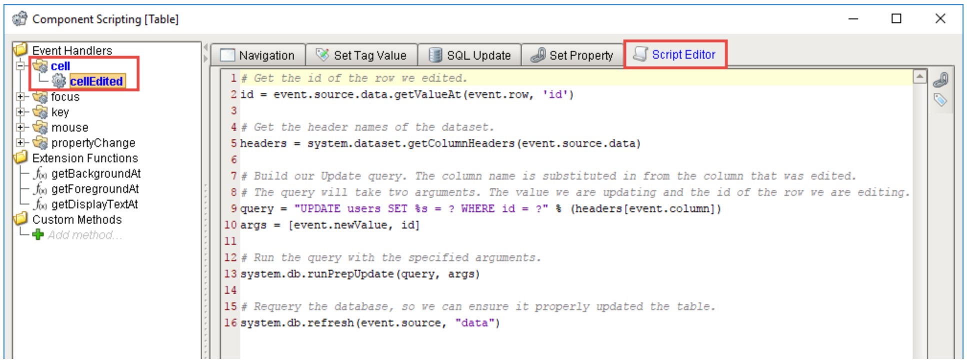
4. Right-click the **Table** and select **Scripting**.

5. Create a script in the **cell > cellEdited** event handler using the script provided below. In this script, we have variables that contain the row number that was edited, the column number that was edited, and the new value. Because we do not have the name of the column that was edited, we must first grab the list of columns using the **getColumnHeaders()** system function. We can then use the column number that was edited to find the name of the column that changed and use it in the query.

Python - Writes to a Cell in the Database When Component is Updated

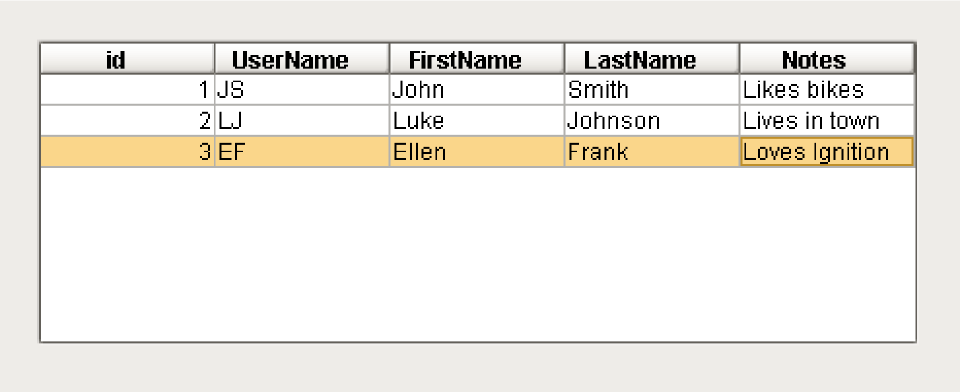
```
# Get the id of the row we edited.
id = event.source.data.getValueAt(event.row, 'id')

# Get the header names of the dataset.
headers = system.dataset.getColumnHeaders(event.source.data)

# Build our Update query. The column name is substituted in from the column that was edited.
# The query will take two arguments. The value we are updating and the id of the row we are
```

```
editing.
query = "UPDATE users SET %s = ? WHERE id = ?" % (headers[event.column])
args = [event.newValue, id]

# Run the query with the specified arguments.
system.db.runPrepUpdate(query, args)

# Requery the database, so we can ensure it properly updated the table.
system.db.refresh(event.source, "data")
```



```
# Get the id of the row we edited.
id = event.source.data.getValueAt(event.row, 'id')

# Get the header names of the dataset.
headers = system.dataset.getColumnHeaders(event.source.data)

# Build our Update query. The column name is substituted in from the column that was edited.
# The query will take two arguments. The value we are updating and the id of the row we are editing.
query = "UPDATE users SET %s = ? WHERE id = ?" % (headers[event.column])
args = [event.newValue, id]

# Run the query with the specified arguments.
system.db.runPrepUpdate(query, args)

# Requery the database, so we can ensure it properly updated the table.
system.db.refresh(event.source, "data")
```

6. Now, you're ready to test it out! Put the Designer in **Preview Mode**, select the cell you want to edit, hit enter to commit the change, or tab to the next cell to make additional edits.



id	UserName	FirstName	LastName	Notes
1	JS	John	Smith	Likes bikes
2	LJ	Luke	Johnson	Lives in town
3	EF	Ellen	Frank	Loves Ignition

Related Topics ...

- Simple Database Editor

Refreshing a SQL Query

Refreshing a Query in Vision

The SQL query that populates a property on a component will refresh its data periodically if the Polling Mode is set to either Relative or Absolute. However, there are times when you want the data to query the database once when the window is opened, and then retain explicit control over each subsequent refresh. The Table data can be requeried using system.db.refresh. There are two main ways of doing this: either placing the script on a button, or at the end of some code.

On this page ...

Refresh After Edits

If edits are being made to the Table data, or additional rows are being added, it can be a good idea to call the system.db.refresh() function after manipulating data. This way, the data in the Table will automatically refresh with the newly entered data, saving the user the hassle of clicking a refresh button.

Pseudocode - Refreshing a Table after an Edit

```
doWork()

system.db.runPrepUpdate("INSERT INTO table")

system.db.refresh(component, "propertyName")
```

INDUCTIVE UNIVERSITY

Refreshing SQL Data on Component

Watch the Video

Refresh on a Button

We can use a Button component with a script on it to refresh the component with the query binding on it. Typically the button is placed close by the query bound component, and will say something like "Refresh Data". The script is actually fairly simple, and can be placed on the **actionPerformed** event of the Button.

1. Drag a **Button** component onto a window that has a Power Table or Table component querying a Database table.
   a. Change the Button's Text property to say "Refresh."

2. On the Power Table component, open the **data** property binding and set the polling mode to **off**.

3. Right click on the Button and select **Scripting**.
   a. Select the **actionPerformed** Event Handler and navigate to the Script Editor tab.
   b. Add in this script, which will force the Table's data property to refresh.

Python - Refreshing the Database when the Button is Pressed

```
# Will force the Power Table's Data property to run the query again.
system.db.refresh(event.source.parent.getComponent('Power Table'), "data")
```

4. Try it out by adding new data to the Database table and then clicking the Refresh Button.

Refreshing a Query in Perspective

The system.db.refresh() function does not work in Perspective. However, an additional component method has been added to handle the more general task of refreshing a binding. The method is called refreshBinding(), and can be called on any Perspective component:

```
self.refreshBinding("props.data")
```

The function takes a string as a parameter, the path to the property to be refreshed. More info on the refreshBinding() function can be found at Perspective Component Methods.

Related Topics ...

- system.db.refresh
- SQL Query Binding
- DB Browse Binding
- Named Query Bindings

Editing Multi-Selected Rows from a Table

Editing Multiple Table Rows

A common user experience is to select multiple rows of a table and then edit all of those rows from the database at once, such as deleting all of them. When a user selects more than one row in a table, there is a special function called getSelectedRows() available on both the Table and Power Table components that returns the row indexes of the table as a Python list. This list can be iterated through in order to delete the selected rows from the database.

Example - Deleting Selected Rows

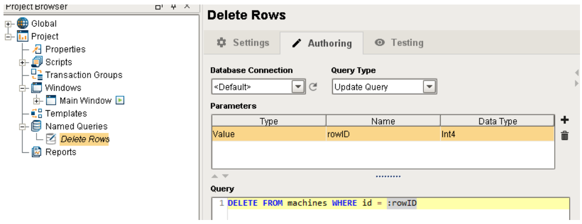
We can use the list of selected rows to delete them from the database. Start with a table in the Database table that looks like this:

id	machine_name	area_number
1	Conveyor	1
2	Press	2
3	Tank	1
4	Packer	3
5	Loader	3
6	Oven 1	3
7	Oven 2	2
8	Wrapper	1
9	Mixer	3
10	Cold Storage	2
11	Dryer	2

1. Create a new Named Query that will be used to delete rows of data. Setup Security to fit your needs, and name it appropriately. For more information on creating Named Queries, see Using Named Queries - Example.
    a. Set the Query Type to **Update Query**.
    b. Create a single Value type Parameter that is an Int4 data type. This will hold the id of the row, so we can name it rowID.
    c. Create the Delete Query.

SQL - Delete Row with Matching ID

```
DELETE FROM machines WHERE id = :rowID
```



Delete Rows

Database Connection

<Default>

Query Type

Update Query

Parameters

Type	Name	Data Type
Value	rowID	Int4

Query

```
DELETE FROM machines WHERE id = :rowID
```

2. Create a second Named Query as in step 1, but this one will be used to select the data into a table.

INDUCTIVE UNIVERSITY

Deleting Multi-Selected Rows from Table

Watch the Video

   a. Set the Query Type to Query, no parameters, and add a basic select query.

SQL - Selecting from a Table

```
SELECT * FROM machine
```

3. On a new Main Window, add a Power Table component and a Button component.

4. On the Power Table's **Data** property, set up a Named Query binding to the Select Query that was made in step 2.
   a. Ensure that the Selection Mode property is set to **Multiple Interval**.

5. On the Button, change the Text property to say "Delete".
   a. Right click on the Button and select Scripting.
   b. Select the actionPerformed Event Handler and navigate to the Script Editor Tab.
   c. Here we need to call the **getSelectedRows()** function on the Power Table to determine what rows are selected, and then loop through those to grab the value of the id column in each row and delete the row based on that id.

Python - Looping Through the Selected Rows and Deleting Them

```
# Get the data from the table and assign it to the
variable called data.
data = event.source.parent.getComponent('Power Table').
data

# Get the rows of the data that the user has currently
highlighted.
rows = event.source.parent.getComponent('Power Table').
getSelectedRows()

# Iterate through each row of the list that is
associated with the rows variable.
for row in rows:

        # Get the value associated with the current row
and the the column called "id".
        id = data.getValueAt(row, "id")

        # Run the query to delete from the database,
where tableName is the name of the database table.
        system.db.runNamedQuery("Delete Machine Rows",
{"rowID":id})

# Refresh the table data.
system.db.refresh(event.source.parent.getComponent
('Power Table'), "data")
```

6. You can test it out by putting the Designer into Preview Mode, selecting a few rows, and then clicking the Delete Button.

Related Topics ...

- Updating the Database through the Power Table
- Refreshing a SQL Query

Storing Files in a Database

Storing and Displaying Files in a Database

On this page ...

Ignition can store different types of files into a database by storing the raw file bytes into a special database column. Ignition can also pull these file bytes out and display certain files within a Client.

Each database has different column types that are used to store files, so it is important to check with your database documentation to see which data type the column would need to be set to for it to accept file bytes. For example, in MySQL, the datatype that accepts PDF bytes is a **LongBlob** datatype, so you will need to set the PDF Data column to the **LongBlob** datatype. MS SQL accepts the **Varbinary** datatype, so you'll need to set the PDF Data column to a **Varbinary** datatype.

Example - PDF File

One of the most common file types that is stored is PDF files. This allows you to store each PDF file within the central database where each client will have access to it, instead of placing the file in a shared drive that all Client computers have access to.

Storing Files in a Database

Watch the Video

Uploading PDF Files to the Database

With a simple script on a Button component, we can store a PDF file into the database so that any user can view it later. This part does not use a Named Query, because the Named Query Parameters do not have a data type that allows us to pass in the raw file bytes. We can instead use system.db.runPrepUpdate to call a query from the script.

This example requires that you have a table with a byte array column in it. For example: MySQL user the BLOB data type and MSSQL uses the varbinary() data type.

This function will not work using default Client Permission settings. The Legacy Database Access will need to be enabled for this to work.

1. Add a Button component to a new Main Window.

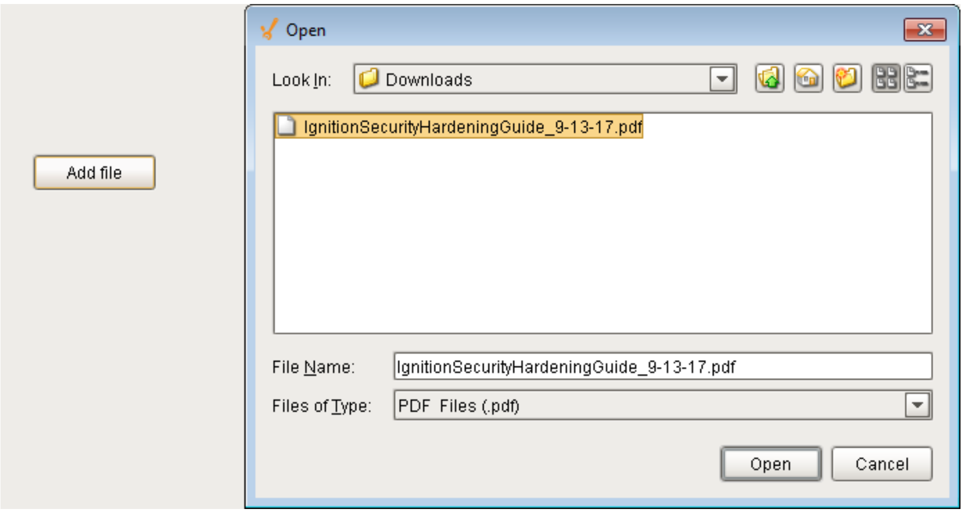
2. Change the Text property to say Add File.

3. Right click on the Button and select Scripting.
   a. Navigate to the Script Editor Tab of the actionPerformed Event Handler. Here we can put a script that will grab the file bytes using the file path and the system.file.readFileAsBytes function, and then insert that into the database, along with a user selected file name.

Python - Uploads PDF Files to a Database Using a Button

```
# Find the path to the PDF file.
path = system.file.openFile("pdf")

# Check to ensure that the user actually selected a filepath.
if path != None:
        # Read the file as bytes.
        data = system.file.readFileAsBytes(path)

        # Ask the user to enter in a filename.
        # Will grab just the filename and extension from the path as a suggestion.
        name = system.gui.inputBox("Enter a name for the file", path.split('\\')[-1])

        # Check to ensure that the user entered a name for the file.
        if name != None:

                # Insert the data and name into the database.
                system.db.runPrepUpdate("INSERT INTO files (fileName, fileBytes) VALUES
(?,?)", [name,data])
```

4. Test out the script by putting the Designer into Preview Mode and clicking the Button.
    a. First, select the file to load.



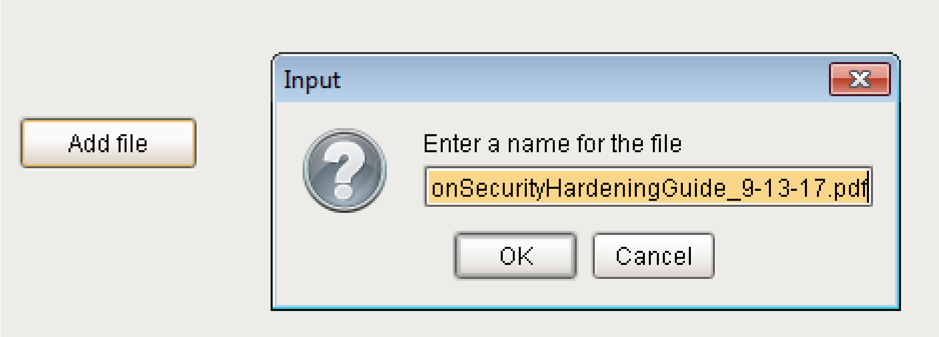
Look In: Downloads

IgnitionSecurityHardeningGuide_9-13-17.pdf

File Name: IgnitionSecurityHardeningGuide_9-13-17.pdf

Files of Type: PDF Files (.pdf)

    b. Then enter in a file name.



Enter a name for the file

onSecurityHardeningGuide_9-13-17.pdf

Displaying PDF Files from the Database

Ignition can render a PDF document inside the Vision - PDF Viewer component, which is a part of the Reporting Module. To view PDF files in the Client, your Ignition server must have the Reporting Module installed. Once the module is installed, you can load the bytes from the database into the PDF Viewer component.

1. Create a new Named Query that will be used to select the file names and ids for a Dropdown selection. Set up security to fit your needs, and name it appropriately. For more information on creating Named Queries, see Using Named Queries - Example.
    a. With no Parameters, add a query to select all the files in our files table.

SQL - Selecting all Files

```
SELECT id, fileName FROM files
```

2. Create a second Named Query, same as in step 1. This will be used to grab the file name and bytes after the user has chosen a file.
    a. Add a single Value type Parameter, "fileID" that will be an Int4 data type.
    b. Add the query to select the file name and bytes based on the selected ID.

SQL - Selecting a File based on an ID

```
SELECT fileName, fileBytes FROM files WHERE id = :fileID
```

3. On the Main Window, add a Dropdown List component and a PDF Viewer component.

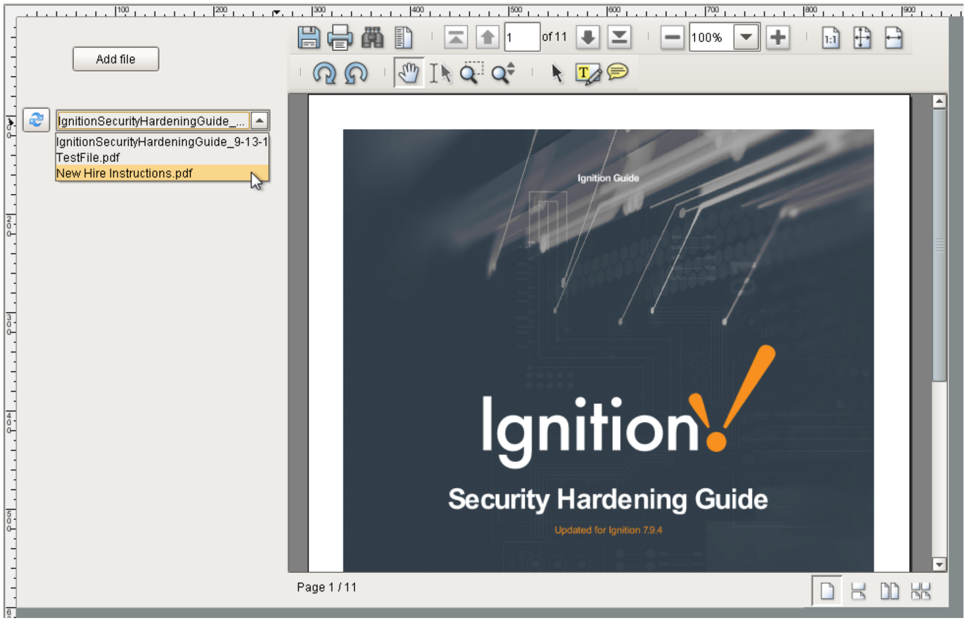
4. On the Dropdown List component, add a Named Query binding to the Data property.
    a. Set the binding to the Named Query that was made in step 1.
    b. Place a small refresh Button next to the Dropdown that will refresh this query. See Refreshing a SQL Query for more information for how to refresh on a button.

5. Right click on the Dropdown List component and select Scripting.
    a. Select propertyChange and navigate to the Script Editor tab.
    b. This script will take any new selected value and use it in the Named Query we made in step 2 to get the file name and bytes. We can then load the bytes into the PDF Viewer.

Python - Displays PDF Files from a Database Using the PDF Viewer

```
# Check to see if the property that changed was the Selected Value property.
if event.propertyName == "selectedRow":

        # Run the query to grab the file name and bytes using the new selected ID value.
        data = system.db.runNamedQuery("Read File", {"fileID":event.newValue})

        # Grab the file bytes and name from the same row.
        bytes = data.getValueAt(0, "fileBytes")
        name = data.getValueAt(0, "fileName")

        # Load the bytes into the PDF Viewer component.
        event.source.parent.getComponent('PDF Viewer').loadPDFBytes(bytes, name)
```

6. Place the Designer into Preview Mode, and try selecting one of the stored files.


Add file
IgnitionSecurityHardeningGuide_...
IgnitionSecurityHardeningGuide_9-13-1
TestFile.pdf
New Hire Instructions.pdf
1
of 11
100%
Ignition Guide
Ignition
Security Hardening Guide
Updated for Ignition 7.9.4
Page 1 / 11

Related Topics ...

- Vision - PDF Viewer

Simple Database Editor

Building a Database Table Editor

On this page ...

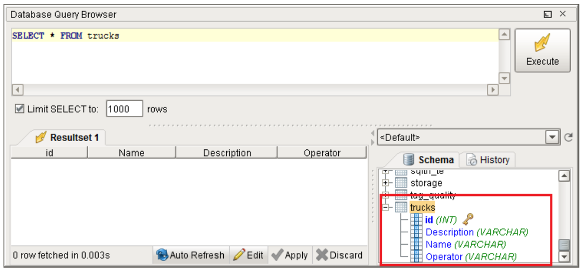
Building a Simple Database Table Editor in Ignition is actually quite easy. Using a few simple components and some scripting, we can easily make a window that will allow users to add, edit, and delete the data from a particular database table. Before we get started, there is a table that has already been created in a database that is used for this example. You can either create this table and follow along, or alter the example to fit your table.

The table is called **trucks**, and it has four columns.

id	Name	Description	Operator

- The **id** column is our primary key, it is non null, and it auto increments
- All other columns are varchar(45)

That is it. There is no need to add any data, as we can add data when testing our tool!



```
SELECT * FROM trucks
```

☑ Limit SELECT to: 1000 rows

id	Name	Description	Operator

0 row fetched in 0.003s

Adding the Components

This example consists of a Table component that will display the data, Button components to add and delete rows from the database table, Text Fields where we can enter in new values, and an Update Details Button that will push those values to the database table.

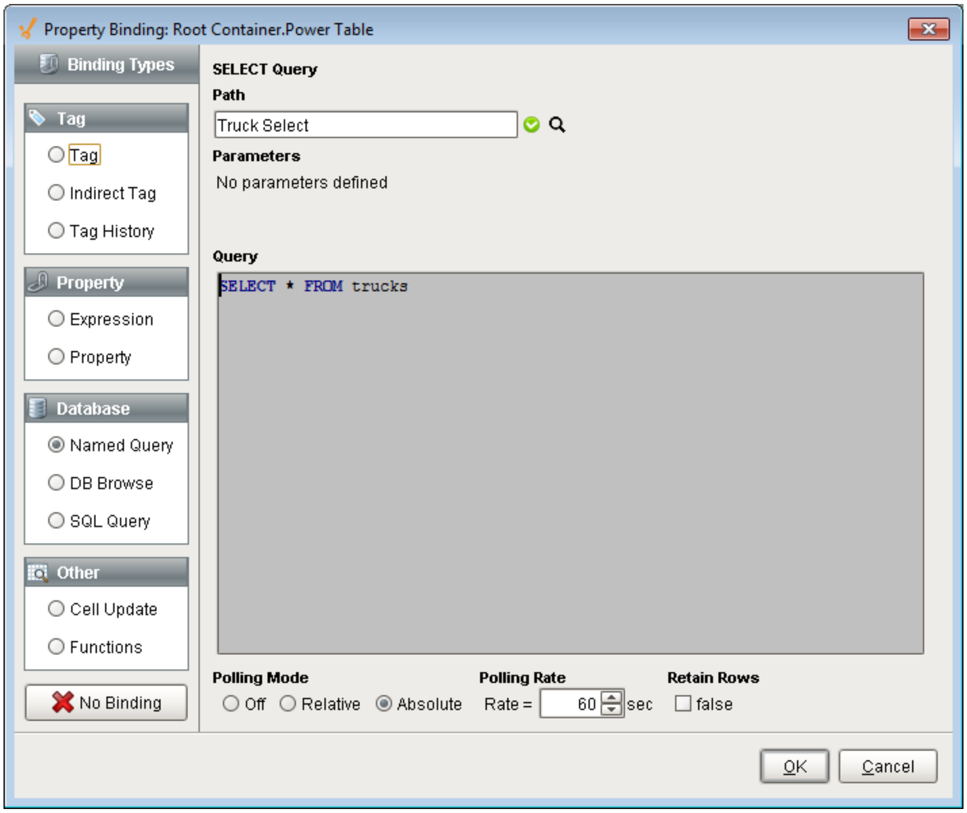
Let's start off by going to the **Designer** and adding the necessary components to the window.

1. Add a **Power Table** component to the window.
   a. We want to query the table in the database that we made earlier, so let's add a **SQL Query Binding** on the **Data** property of the Table.
   b. Add the following code under the **SELECT Query** area.

      SQL - Selects all Trucks

      ```
      SELECT * FROM trucks
      ```

   c. Set the **Polling Mode** to **Absolute** and the **Polling Rate** to **60 seconds**, so that it periodically updates if left open.
   d. Since we want to use the secure Named Query system, click the convert to Named Query button under the polling mode, and call the query "Truck Select"



SELECT Query

Path: Truck Select

Parameters: No parameters defined

Query

```
SELECT * FROM trucks
```

Polling Mode: ○ Off ○ Relative ● Absolute

Polling Rate: Rate = 60 sec

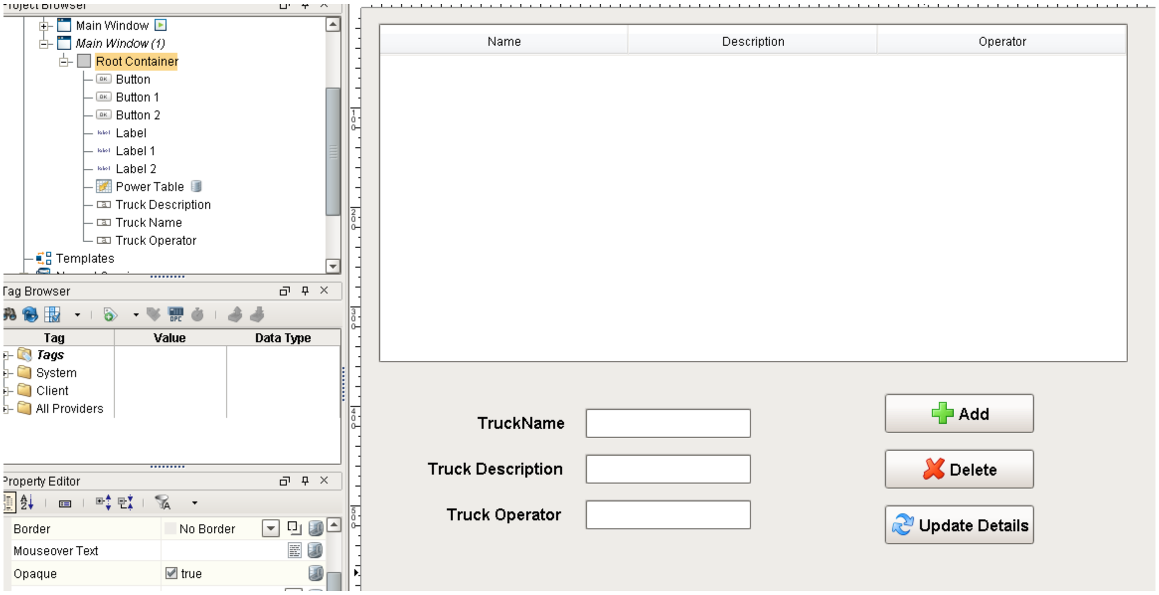
Retain Rows: □ false

e. With the Table still selected, open the **Table Customizer** by right clicking on the Power Table and going to **Customizers > Table Customizer**.
f. Set the **Hide** property of the **id** column to **true**. We don't need to show that column to our users. Click **OK** to close the Table Customizer.
g. In the **Property Editor**, make sure the **Selection Mode** property of the Power Table is set to **Multiple Interval**.

2. Add three **Button** components to the right of the **Table**.
   a. Change the **Text** property on one Button to say "**Add**."
   b. Change the **Text** property of the second Button to say "**Delete**".
   c. Change the **Text** property of the final Button to say "**Update Details**"
   d. Optionally, you can also add an image to the **Image Path** of each button such as a green plus and a red X, and resize them as necessary.

3. Add three **Label** components and three **Text Field** components under the **Table**.
   a. Pair up a Label with each Text Field.
   b. Rename the Text Fields:
      Truck Name, Truck Description, Truck Operator.
   c. Change the **Text** property of each Label to match the name of their corresponding Text Fields.

Excellent! Now we have our window configured with all of the components we need. Now we can start adding scripts that will alter the components.



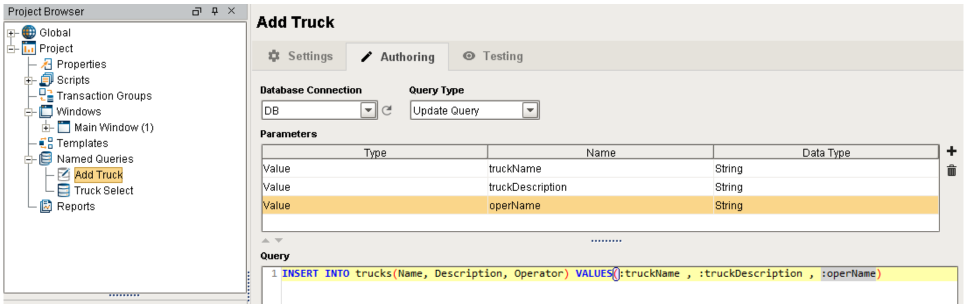
Add a Truck

First, we can add a script to our **Add** button that will allow us to add a row to the Table. We can use the Text Fields that we added so the user can enter in values to insert into the table.

1. Create a new Named Query which we will use to insert new data. Set up security to fit your needs and name it appropriately. For more information on creating Named Queries, see Using Named Queries - Example.
   a. Set the Query Type to Update Query.
   b. Create three Value type Parameters.
      i. The first will be a string data type and will be used for the name of the truck so it can be called "truckName".
      ii. The second will be a string data type and will be used for the description of the truck so it can be called "truckDescription".
      iii. The third will also be a string data type and will be used for the operators name so it can be called "operName".
   c. Create the insert query.

SQL - Adding a Truck

```
INSERT INTO trucks(Name, Description, Operator) VALUES(:truckName , :truckDescription , :
operName)
```



Database Connection: DB

Query Type: Update Query

Parameters

Type	Name	Data Type
Value	truckName	String
Value	truckDescription	String
Value	operName	String

Query

```
INSERT INTO trucks(Name, Description, Operator) VALUES(:truckName , :truckDescription , :operName)
```

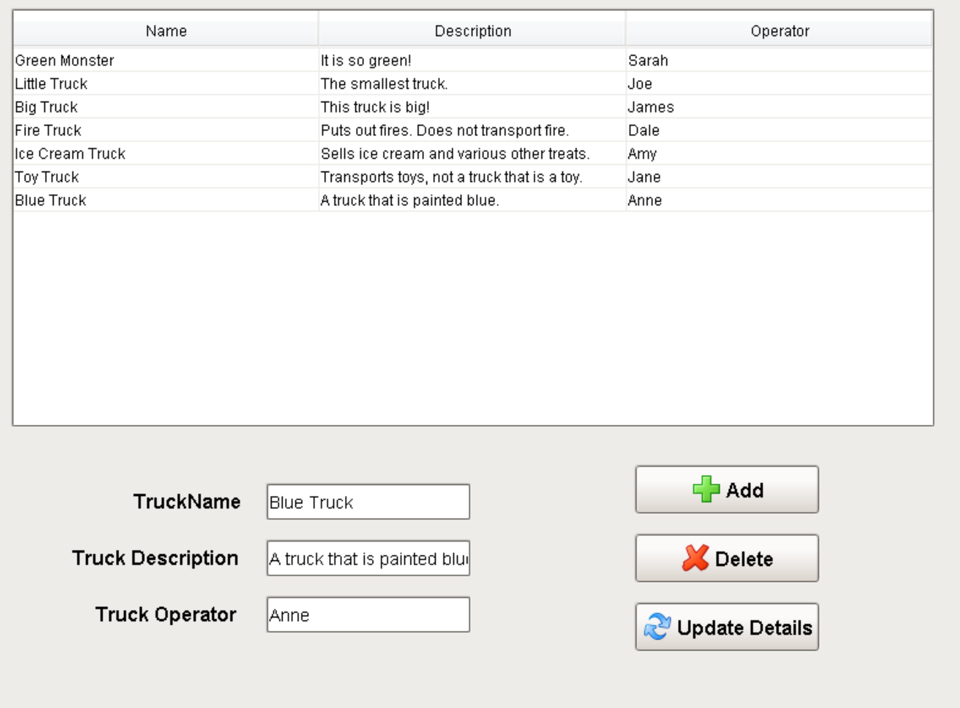
2. Back on our window with components, right click on the **Add Button** and select Scripting. Select the **actionPerformed** Event Handler and navigate to the **Script Editor** tab.
   a. Here we can add some code that will pull the values from the three Text Field components and then use those values in the Named Query we made in step 1.

Python - Adds a New Blank Row and Refreshes the Table

```
# Grab the values from the Text Fields.
name = event.source.parent.getComponent('Truck Name').text
description = event.source.parent.getComponent('Truck Description').text
operator = event.source.parent.getComponent('Truck Operator').text
```

```
# Use those values in the Add Truck Named Query.
system.db.runNamedQuery("Add Truck", {"truckName":name, "truckDescription":description,
"operName":operator})

# Refresh the table to immediately bring in the new row.
system.db.refresh(event.source.parent.getComponent("Power Table"), "data")
```

3. Now we can test out our script and query by putting the Designer into **Preview Mode**, adding some data to the **Text Fields**, and clicking the **Add Button**. You should see a row populate the table. Try adding a few rows to fill our table with data. Make sure you have the designer Communication Mode set to Read/Write to test. Go back to **Design Mode** before moving on.



Name	Description	Operator
Green Monster	It is so green!	Sarah
Little Truck	The smallest truck.	Joe
Big Truck	This truck is big!	James
Fire Truck	Puts out fires. Does not transport fire.	Dale
Ice Cream Truck	Sells ice cream and various other treats.	Amy
Toy Truck	Transports toys, not a truck that is a toy.	Jane
Blue Truck	A truck that is painted blue.	Anne

TruckName: Blue Truck

Truck Description: A truck that is painted blu

Truck Operator: Anne

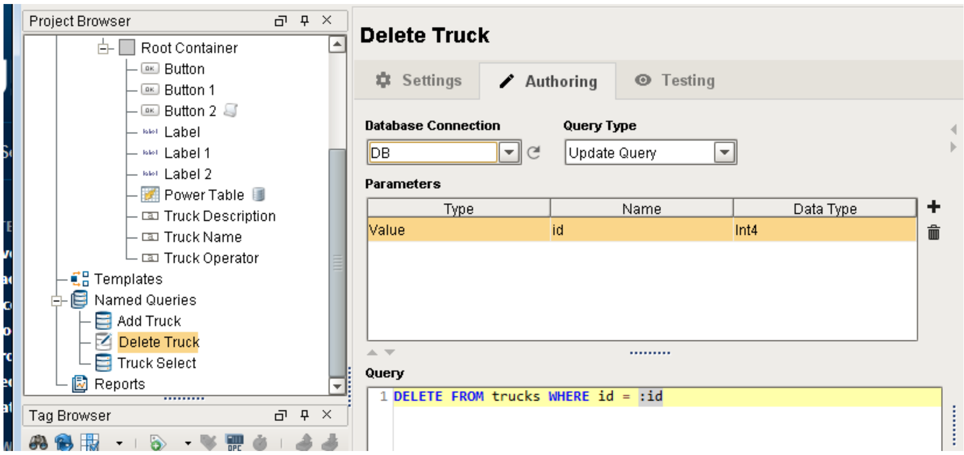
Delete a Truck

Now that we can add rows, let's add a script to our **Delete button** that will delete rows. Since our users can select multiple rows, our script needs to take that into account, and delete all of the rows that the user selected. It will also have a **Message Box** popup if there are no rows selected, informing the user that they need to select at least one row. We also want to add a confirmation before the rows actually get deleted.

1. Create a new Named Query which we will use to delete a row of data. Set up security to fit your needs and name it appropriately. For more information on creating Named Queries, see Using Named Queries - Example.
    a. Set the Query Type to Update Query.
    b. Create one Value type Parameters.
        i. It will be a Int4 data type and will be used for the id of row so it can be called "id".
    c. Create the delete query.

SQL - Deleting a Truck

```
DELETE FROM trucks WHERE id = :id
```



Type	Name	Data Type
Value	id	Int4

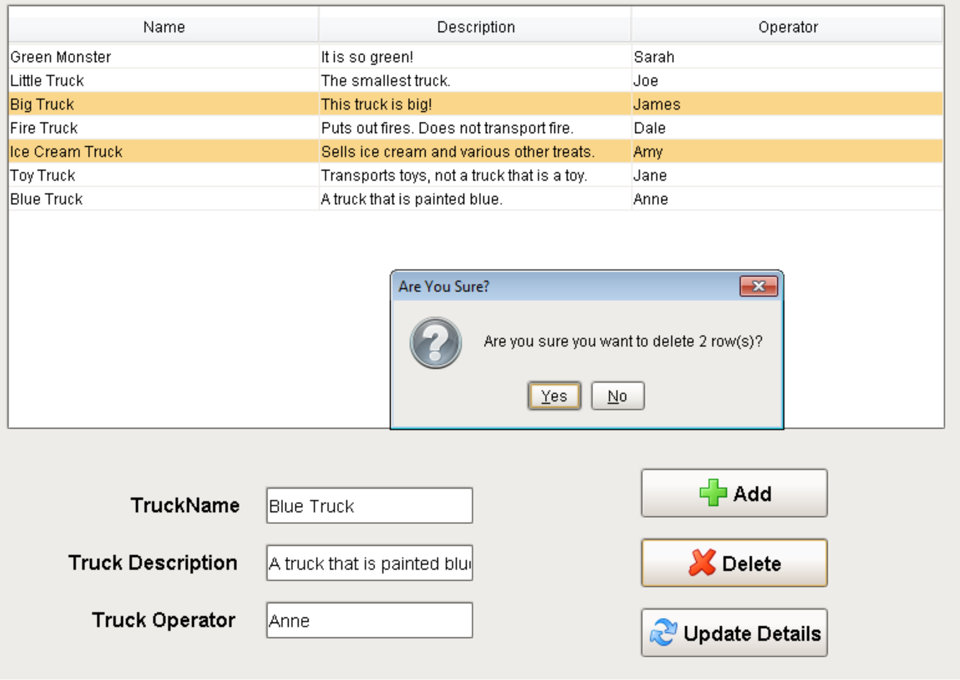
```
DELETE FROM trucks WHERE id = :id
```

2. Back on our window with components, right click on the **Delete Button** and select Scripting. Navigate to the **Script Editor** tab on the **actionPerformed** Event Handler.
   a. Here we can add some code that will delete the user's selected rows. First we grab the Power Table's selected rows, and check to make sure rows are actually selected. If there are selected rows, then we have a popup that confirms the user wants to delete rows, and it shows the number of rows that will be deleted. Once the user confirms, the script will then loop through the selected rows and delete each one.

Python - Deletes the Row or Rows

```
# Get the selected row or rows.
selRows = event.source.parent.getComponent('Power Table').getSelectedRows()

# Check to see that a row is actually selected.
if len(selRows) > 0:

        # If a row is selected, ask for confirmation before deleting the rows.
        if system.gui.confirm("Are you sure you want to delete " + str(len(selRows)) + " 
row(s)?", "Are You Sure?", 0):

                # If the user gave confirmation, we need to loop through all of them.
                for row in selRows:

                        # For each selected row, we need to get the value of the id column 
in that row.
                        id = event.source.parent.getComponent('Power Table').data.
getValueAt(row, "id")

                        # Use the id of the row to delete it from the database table.
                        system.db.runNamedQuery("Delete Truck", {"id":id})

                # Refresh the table after deleting all selected rows
                # to immediately remove the selected rows from the Power Table.
                system.db.refresh(event.source.parent.getComponent('Power Table'), "data")

        # If the user said no to the delete.
        else:
                system.gui.messageBox("User canceled the delete.", "Delete Canceled")

# If no row is selected, have a popup message that asks the user to select a row.
else:
        system.gui.messageBox("Please select at least one row.", "Select A Row")
```

3. Now we can test our **Delete button** by putting the Designer into **Preview Mode** and clicking the **Delete button**. When one or more rows are selected and the **Delete button** is pressed, a popup will confirm we want to delete the rows before deleting them. (You can select multiple rows with **Shift-Click** or **Control-Click**). When no rows are selected and the **Delete button** is pressed, a **Message Box** will pop up, informing the user to select a row first. You can deselect rows in the Table by clicking on the Table and pressing the **Escape** key. Be sure to go back to Design Mode before moving on.



Name	Description	Operator
Green Monster	It is so green!	Sarah
Little Truck	The smallest truck.	Joe
Big Truck	This truck is big!	James
Fire Truck	Puts out fires. Does not transport fire.	Dale
Ice Cream Truck	Sells ice cream and various other treats.	Amy
Toy Truck	Transports toys, not a truck that is a toy.	Jane
Blue Truck	A truck that is painted blue.	Anne

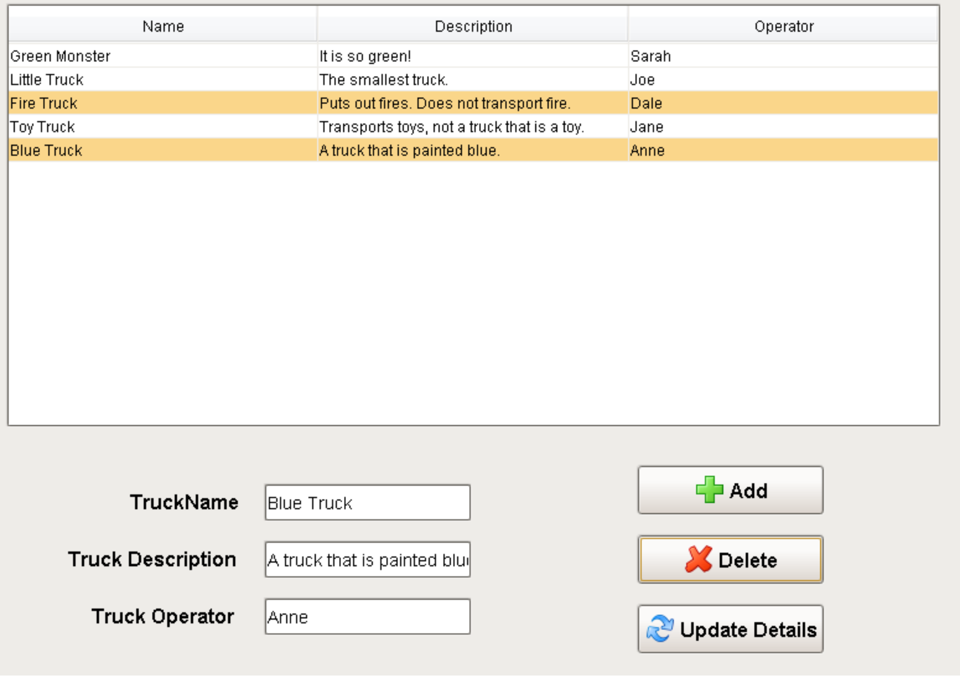
Are You Sure?

Are you sure you want to delete 2 row(s)?

TruckName: Blue Truck

Truck Description: A truck that is painted blu

Truck Operator: Anne



Name	Description	Operator
Green Monster	It is so green!	Sarah
Little Truck	The smallest truck.	Joe
Fire Truck	Puts out fires. Does not transport fire.	Dale
Toy Truck	Transports toys, not a truck that is a toy.	Jane
Blue Truck	A truck that is painted blue.	Anne

TruckName: Blue Truck

Truck Description: A truck that is painted blu

Truck Operator: Anne

Populate Text Fields

Now that we can add and delete rows from our Table, we need a way to edit the data contained in them. However, before we can create a script on our **Update Details button**, we first need to populate the Text Fields with the current values of the selected row. We can do this by adding a simple expression to each of the text fields.

1. On the Truck Name Text Field, add an expression binding to the Text property.
    a. The expression will grab the dataset and will grab the value at the selected row of the Name column. The expression is wrapped in a try so that if no rows are selected, it will return an empty string.

Expression - Grabbing the Selected Row's Name Value

```
try({Root Container.Power Table.data}[{Root Container.Power Table.selectedRow}, "Name"],
"")
```

2. On the Truck Description Text Field, add an expression binding to the Text property.

   a. The expression will grab the dataset and will grab the value at the selected row of the Description column. The expression is wrapped in a try so that if no rows are selected, it will return an empty string.

Expression - Grabbing the Selected Row's Description Value

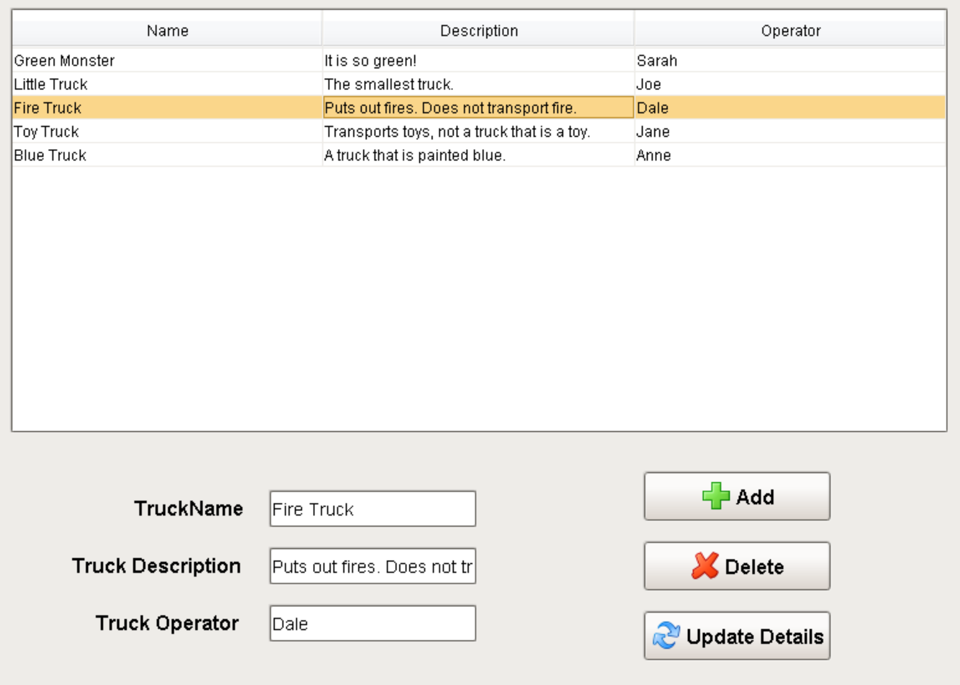
```
try({Root Container.Power Table.data}[{Root Container.Power Table.selectedRow},
"Description"], "")
```

3. On the Truck Operator Text Field, add an expression binding to the Text property.
   a. The expression will grab the dataset and will grab the value at the selected row of the Operator column. The expression is wrapped in a try so that if no rows are selected, it will return an empty string.

Expression - Grabbing the Selected Row's Operator Value

```
try({Root Container.Power Table.data}[{Root Container.Power Table.selectedRow},
"Operator"], "")
```

4. Click on a row to see the fields below populate. Press the ESC key to empty the selection.



Name	Description	Operator
Green Monster	It is so green!	Sarah
Little Truck	The smallest truck.	Joe
Fire Truck	Puts out fires. Does not transport fire.	Dale
Toy Truck	Transports toys, not a truck that is a toy.	Jane
Blue Truck	A truck that is painted blue.	Anne

TruckName: Fire Truck

Truck Description: Puts out fires. Does not tr

Truck Operator: Dale

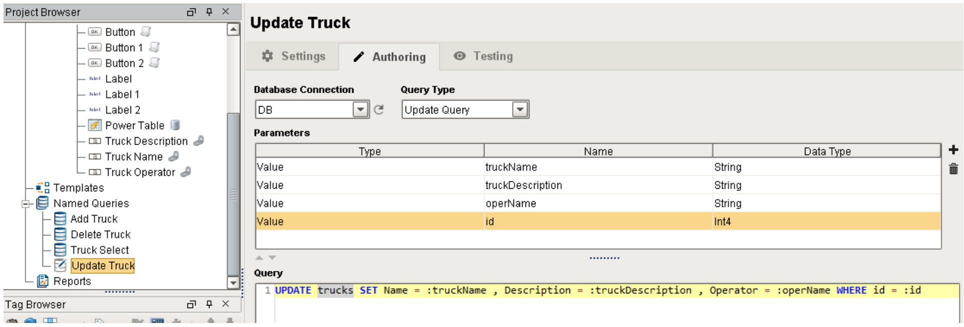
Update Data

The last part of setting up this window is to add a script to the **Update Details button** that will pull the data from the Text Fields and insert it into the selected row of the database Table. Additionally, the script will check to ensure that a row selected, and that only one row is selected. We wouldn't want to confuse users who may have accidentally selected multiple rows at a time, since our button will only update a single row.

1. Create a new Named Query which we will use to update a row of data. Set up security to fit your needs and name it appropriately. For more information on creating Named Queries, see Using Named Queries - Example.
   a. Set the Query Type to Update Query.
   b. Create four Value type Parameters.
      i. The first will be a string data type and will be used for the truck name of row so it can be called "truckName".
      ii. The second will be a string data type and will be used for the truck description of row so it can be called "truckDescription".
      iii. The third will be a string data type and will be used for the operator name of row so it can be called "operName".
      iv. The fourth will be a Int4 data type and will be used for the id of row so it can be called "id".
   c. Create the update query.

SQL - Deleting a Truck

```
UPDATE trucks SET Name = :truckName , Description = :truckDescription , Operator = :
operName WHERE id = :id
```



Update Truck

Database Connection: DB

Query Type: Update Query

Parameters

Type	Name	Data Type
Value	truckName	String
Value	truckDescription	String
Value	operName	String
Value	id	Int4

Query

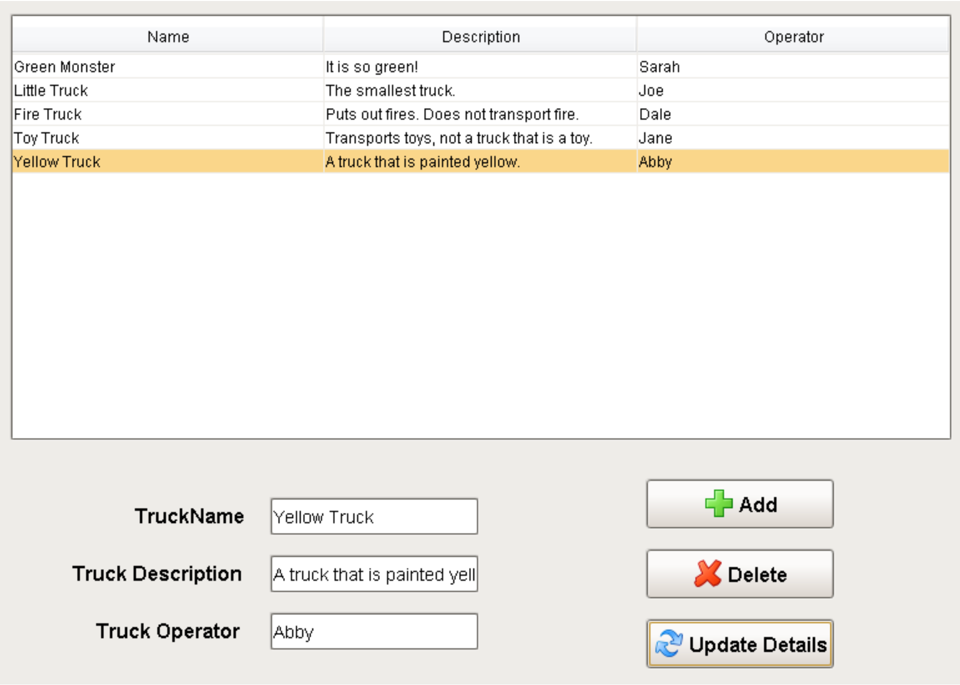
```
UPDATE trucks SET Name = :truckName , Description = :truckDescription , Operator = :operName WHERE id = :id
```

2. Back on our window with components, right click on the **Update Button** and select Scripting. Navigate to the **Script Editor** tab on the **actionPerformed** Event Handler.
    a. Here we can add some code that will update the users selected row with data pulled from the Text Fields. First we check to ensure that only one row is selected. Next the script asks them to confirm that the user wants to update the selected row. If the user confirms, the script then pulls in the values of all three Text Fields and uses those values along with the selected rows id to update the database table with new data. It will then refresh the table after updating so that the new data can be brought into the table.

Python - Deletes the Row or Rows

```
# Check to make sure only one row is selected.
if len(event.source.parent.getComponent('Power Table').getSelectedRows()) == 1:

        # If a row is selected, ask for confirmation before updating the row.
        if system.gui.confirm("Are you sure you want to update the selected row?", "Are 
You Sure?", 0):

                # Grab the values from the text fields.
                name = event.source.parent.getComponent('Truck Name').text
                description = event.source.parent.getComponent('Truck Description').text
                operator = event.source.parent.getComponent('Truck Operator').text

                # Grab the selected row.
                selRow = event.source.parent.getComponent('Power Table').selectedRow

                # Using the selected row, we need to get the value of the id column in 
that row.
                id = event.source.parent.getComponent('Power Table').data.getValueAt
(selRow, "id")

                # Run a query that will update the values of the row matching the id.
                system.db.runNamedQuery("Update Truck", {"truckName":name, 
"truckDescription":description, "operName":operator, "id":id})

                # Refresh the table to immediately show the updated data.
                system.db.refresh(event.source.parent.getComponent('Power Table'), "data")

        # If the user said no to the update.
        else:
                system.gui.messageBox("User canceled the update.", "Update Canceled")

# If there more than one row selected, or no rows selected.
else:

        # Have a popup asking the user to select one row.
        system.gui.messageBox("Please select one row to edit.", "Select One Row")
```

3. Now that we added this last bit of code, we can test it by selecting a row, adding data to each **Text Field**, and clicking the **Update Details button**. You should see the data populate the row of the Table, since the table in the database has been updated. Additionally, you should notice how the Text Fields will automatically update when selecting a new row. Try adding a few rows of valid data to your Table.



Name	Description	Operator
Green Monster	It is so green!	Sarah
Little Truck	The smallest truck.	Joe
Fire Truck	Puts out fires. Does not transport fire.	Dale
Toy Truck	Transports toys, not a truck that is a toy.	Jane
Yellow Truck	A truck that is painted yellow.	Abby

TruckName: Yellow Truck

Truck Description: A truck that is painted yell

Truck Operator: Abby

Related Topics ...

- Inserting Data into a Database
- Updating the Database through the Power Table
- Editing Multi-Selected Rows from Table
- Named Query Caching

Basic SQL Troubleshooting

Learning when something is wrong with your database queries and learning how to resolve the problem is key to getting the best possible use out of the database connection to Ignition. Typically, the biggest problem that users face with queries is the data taking too long to load on the window, but an error in a query can also be difficult to track down. This section details what to look for when the query fails to execute, as well as what to do when facing slow queries and how to optimize them within the project to help return data to the window as quickly as possible.

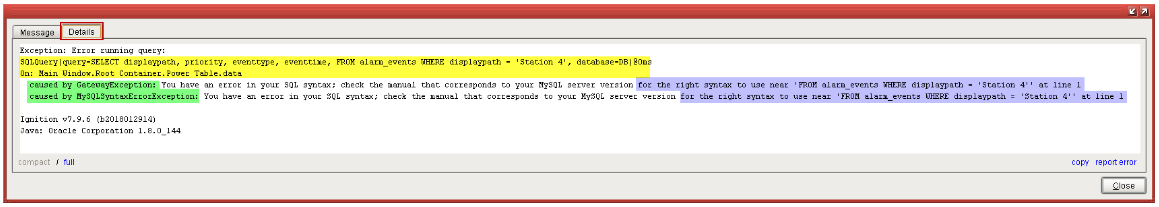
On this page ...

Error Message Box

If a query fails to execute, chances are there was an error box that popped up. Resist the urge to close it! The Error Box usually contains a lot of useful info that can help troubleshoot why the query is failing. When looking at the error message, there are a few things to look for. First, you will want to click on the **Details** tab in the upper left corner of the Error Box. This will bring up the details of the error, which contains the information we need to track down the error.

In the image below, you can see the yellow highlighted part shows us what our query is, as well as where it is executing from. This helps to ensure we are looking at the correct error message for the query we are trying to fix. Next, you want to look for the words '**caused by**' in the text of the error details, which tell us the reason for the error. You can see in the image below, there are two of '**caused by**' messages highlighted in green.The first one is from the Gateway, and the second one is from the database (in this case, a MySQL database).

After locating the '**caused by**', the following message will help to pinpoint what the error is. In this example, we have an error with our syntax in our SQL statement. The most helpful portion of the error is at the end of the message highlighted in purple. This lets us know where the syntax error is in the SQL statement. The trick is to look immediately before the quoted query. In the image below, it starts to quote the original query with '**FROM alarm_events WHERE.....**', so in my original query, I need to look at what was right before the '**FROM**'. You can see in my original query highlighted in yellow, there is a comma right before the '**FROM**', which is incorrect, as the last column in the **SELECT** statement should not have a comma after it.

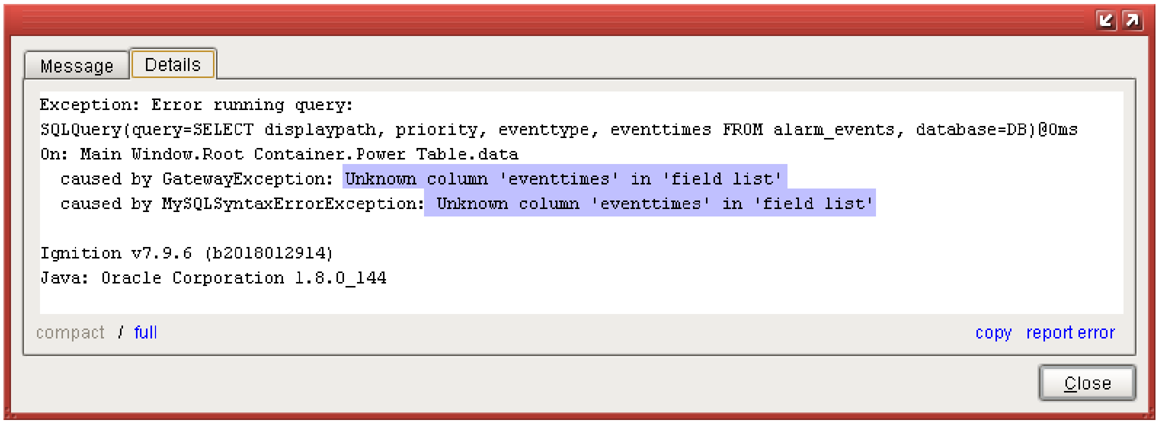


```
Exception: Error running query:
SQLQuery(query=SELECT displaypath, priority, eventtype, eventtime, FROM alarm_events WHERE displaypath = 'Station 4', database=DB)@0ms
On: Main Window.Root Container.Power Table.data
  caused by GatewayException: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'FROM alarm_events WHERE displaypath = 'Station 4'' at line 1
  caused by MySQLSyntaxErrorException: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'FROM alarm_events WHERE displaypath = 'Station 4'' at line 1

Ignition v7.9.6 (b2018012914)
Java: Oracle Corporation 1.8.0_144
```

Other Common SQL Errors

Unknown Column

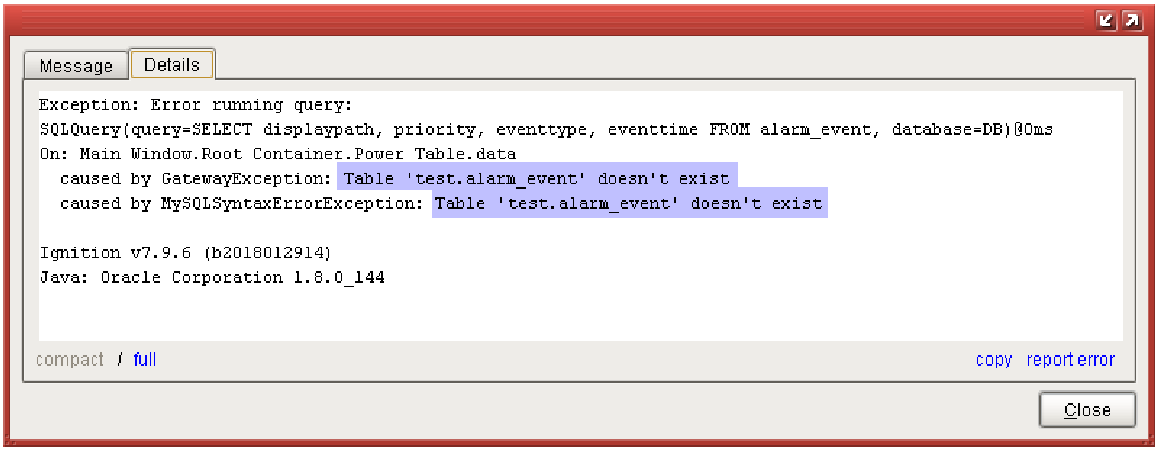
Typically, the name of the column is wrong because of a misspelling. In this case, the correct spelling was 'eventtime', but there was an accidental 's' added to the name.



```
Exception: Error running query:
SQLQuery(query=SELECT displaypath, priority, eventtype, eventtimes FROM alarm_events, database=DB)@0ms
On: Main Window.Root Container.Power Table.data
   caused by GatewayException: Unknown column 'eventtimes' in 'field list'
   caused by MySQLSyntaxErrorException: Unknown column 'eventtimes' in 'field list'

Ignition v7.9.6 (b2018012914)
Java: Oracle Corporation 1.8.0_144
```

Unknown Table

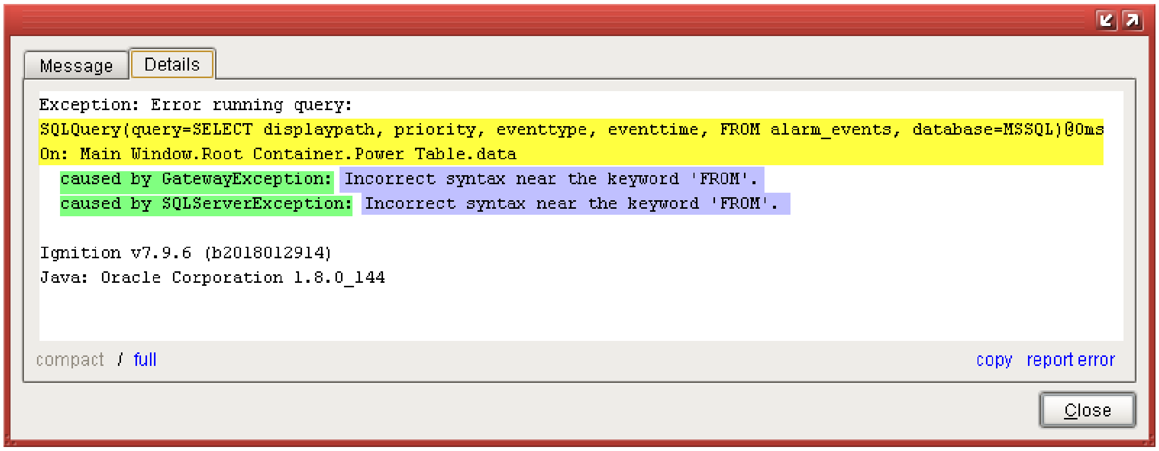
If the table is not found in the database, it may also be because of a misspelling. Here, the correct table name should be 'alarm_events'.



```
Exception: Error running query:
SQLQuery(query=SELECT displaypath, priority, eventtype, eventtime FROM alarm_event, database=DB)@0ms
On: Main Window.Root Container.Power Table.data
   caused by GatewayException: Table 'test.alarm_event' doesn't exist
   caused by MySQLSyntaxErrorException: Table 'test.alarm_event' doesn't exist

Ignition v7.9.6 (b2018012914)
Java: Oracle Corporation 1.8.0_144
```

Any Database Error

The previous error examples have all been from a MySQL database, but the same principles apply to any database. Simply locate the 'caused by', and look at the message afterwards. In this MSSQL syntax error, instead of displaying everything after the syntax error, it only displays the part immediately after. The problem in this query is an extra comma before the word 'FROM'.



```
Exception: Error running query:
SQLQuery(query=SELECT displaypath, priority, eventtype, eventtime, FROM alarm_events, database=MSSQL)@0ms
On: Main Window.Root Container.Power Table.data
   caused by GatewayException: Incorrect syntax near the keyword 'FROM'.
   caused by SQLServerException: Incorrect syntax near the keyword 'FROM'.

Ignition v7.9.6 (b2018012914)
Java: Oracle Corporation 1.8.0_144
```

Checking the Database Connection

Checking the Database Connection in the Gateway can also be useful to ensure there is a valid connection and there is nothing blocking the execution of queries. The Status page of the Gateway can be used to determine if queries are taking too long, while the Database Connections page can be used to alter the settings of the database to better handle the number of queries running from the Gateway to the database. See the Slow Queries page for more information on changing these settings.

Testing Query Results

Using the Database Query Browser is a good way to test out a query before actually running it on a component. This can help you see what results will be returned so that the query can be modified to narrow down how the query should be formatted.

Checking for Slow Queries

There are several ways that slow running queries can cause improper behavior. See Slow Queries for more details.

Related Topics ...

- SQL in Ignition

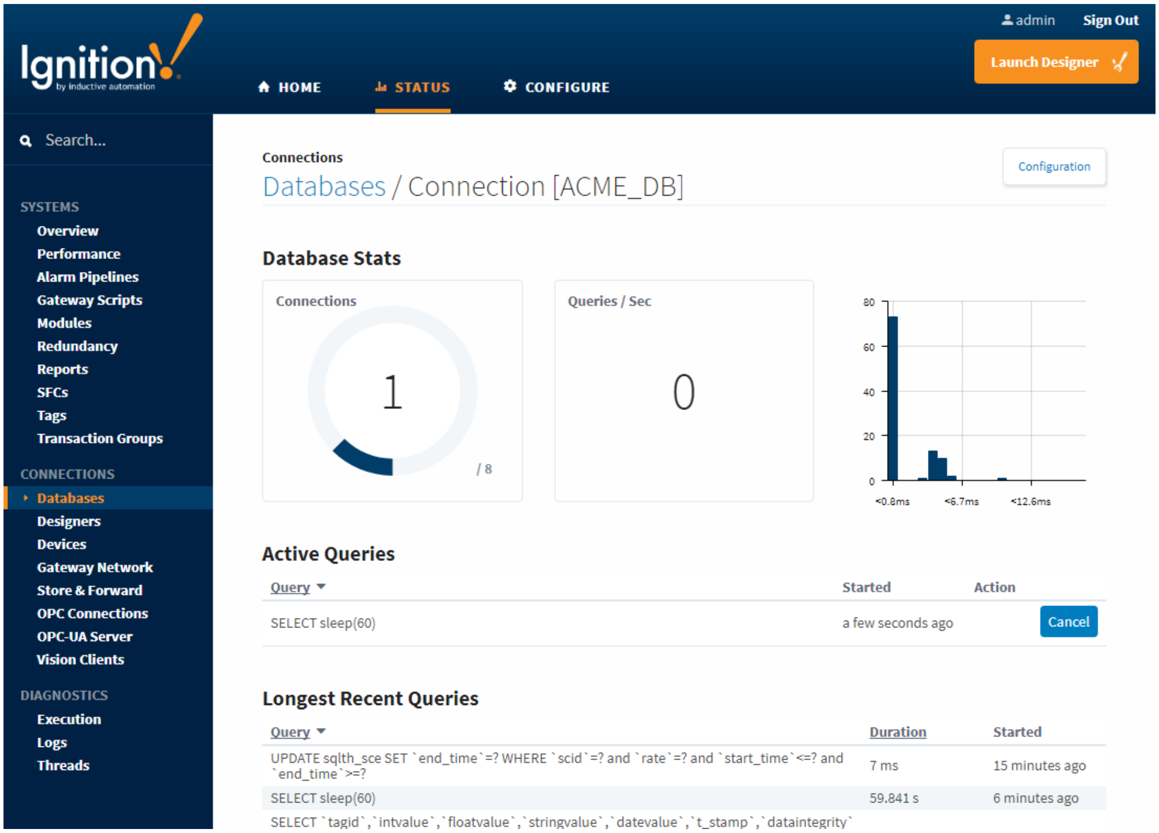
In This Section ...

Slow Queries

Slow running queries can be a big problem. Not only can data take a long time to display on the screen, but it can end up slowing down the whole client. Here, we take a look at some of the things that you can do when your project has a slow running query.

Identify the Slow Query

The first step in dealing with slow queries is identifying which query is actually running slowly. It is typically obvious when a window is opened and a component takes time before database data is displayed. Chances are, the query is on that component somewhere. However, it may also be a good idea to check the queries that are being run against that database connection in the Gateway. From the Gateway webpage, navigate to **Status** **Connections** **Databases** to see a list of all database connections. Clicking on the **Details** button to the right of the database connection will show all of the currently running queries, the most recent long running queries, as well as some basic metrics for that connection. Here, you can get a good idea of any queries that may be a little slower than the others.



Active Queries

Query	Started	Action
SELECT sleep(60)	a few seconds ago	Cancel

Longest Recent Queries

Query	Duration	Started
UPDATE sqlth_sce SET \`end_time\`=? WHERE \`scid\`=? and \`rate\`=? and \`start_time\`<=? and \`end_time\`>=?	7 ms	15 minutes ago
SELECT sleep(60)	59.841 s	6 minutes ago
SELECT \`tagid\`,\`intvalue\`,\`floatvalue\`,\`stringvalue\`,\`datevalue\`,\`t_stamp\`,\`dataintegrity\`		

Execute Against the Database Management Software

The next step in identifying the type of slow query we are dealing with is to run the query directly within the database management software. By cutting out Ignition, we can determine if the query is actually running slowly, or if there is a problem within Ignition that is making the query run slow.

Fast Query in the Database Management Software

If the query runs quickly in the database management software, then there are a few things we can take a look at that may help out.

Check the Database Connection

First, check to make sure the database connection is valid, and there are no warnings associated with it. You may want to go into the connection settings for that database and take note of the value of the **Max Active** property, which determines the maximum number of active connections to the database as well as the **Max Wait** property, which is the number of milliseconds to wait for a connection to come available. To get to the database connections settings, go to **Configure** page of the Gateway webpage and select **Databases Connections**, locate your database, and hit the **edit** button. Here, you can check the database settings. Open **Advanced Properties** at the bottom of the screen, and you'll find the **Max Active** property as well as the **Max Wait** property.

Check Currently Executing Queries

Back in the **Status** section of the Gateway Webpage on the **Database Connection** page mentioned above, we can see a list of currently running queries, as well as how many of the active connections are being used. If the max number of connections is being used, it may be that there are so many queries running that each query needs to wait for an active connection to open up. If this is the case, you may want to increase the amount of active connections to the database or take steps to reduce the query load on the database.

Slow Query in the Database Management Software

If the query also runs slow within the database management software, then the query is just a slow query. Unfortunately, nothing within Ignition can speed up the execution of that query, so you would want to instead take a look at what the query is doing. If the query is pulling in lots of data, you can try breaking the query down into smaller queries, or writing the query in a more efficient way.

For particularly large tables, it may also be helpful to add an index to one of the table's columns. Indexes are something that the user can't see, but help the database speed up data retrieval. However, adding an index to a table will increase the amount of time that an update to the table takes, because the index also needs to be updated. For this reason, it is recommended to only make indexes on columns that are frequently searched against.

To make an index, most database management software have built in interfaces that allow you to customize the index on each table. An index can be made for a single column, or a combination of columns in the table. Talk to your Database Administrator about adding or updating table indexes.

Alternately, you may need to take a look at the database system as a whole. As the size of the database grows, you may need to update the hardware resources available to it. If the database is installed on a server with another system like Ignition, be aware that although the two systems are now sitting next to each other, they now have to share the hardware resources available to them, which may cause issues for both systems. In many cases, it is often better to have the database run on a separate server, which gives it ample room to grow.

Related Topics ...

- SQL Query Volume Optimization
- Connections - Databases

SQL Query Volume Optimization

On this page

...

Overview

Leveraging a SQL Database can drastically improve the quality of a project, but improper database consideration while designing can lead to poor performance later on. This page contains some best practices and considerations when incorporating SQL queries.

Optimizing Individual Queries

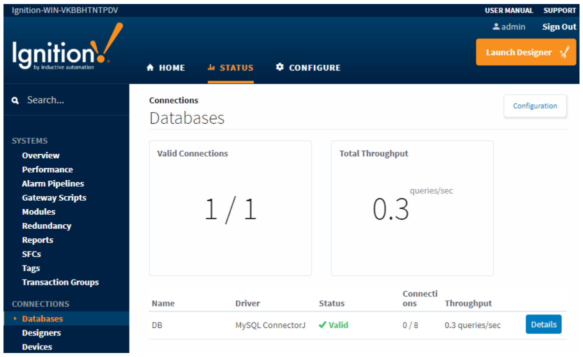
Optimizing individual queries to run faster and be more efficient is very difficult to do properly and can vary widely depending on a number of factors such as the way the tables are setup in the database and what data is being pulled out. Because each database can vary widely from another, there isn't any general way of improving the efficiency of your queries. Instead we recommend becoming more familiar with the SQL language as well as how the data is setup in your system and exactly what data you want to retrieve. This knowledge can help you build better queries.

Your company may also have a database administrator who would manage the database system for your company and would be familiar with figuring out the best way of retrieving data. They may be able to help you retrieve the data that you need.

Always Think Large

Consider how many instances of a query may be running at any given point in time. A single SQL Query Binding will be called for each instance of the window that is open, so if 50 clients are all looking at the same window, 50 separate queries will be called. If the binding is configured to poll, then 50 queries will poll at the rate specified for this single binding. This is already a fair amount of work without factoring in other systems, such as Tag Historian.

To provide context, you can always check the Status Section of the Gateway Webpage to check current throughput of each database connection.



Connections

Databases

Valid Connections 1 / 1

Total Throughput 0.3 queries/sec

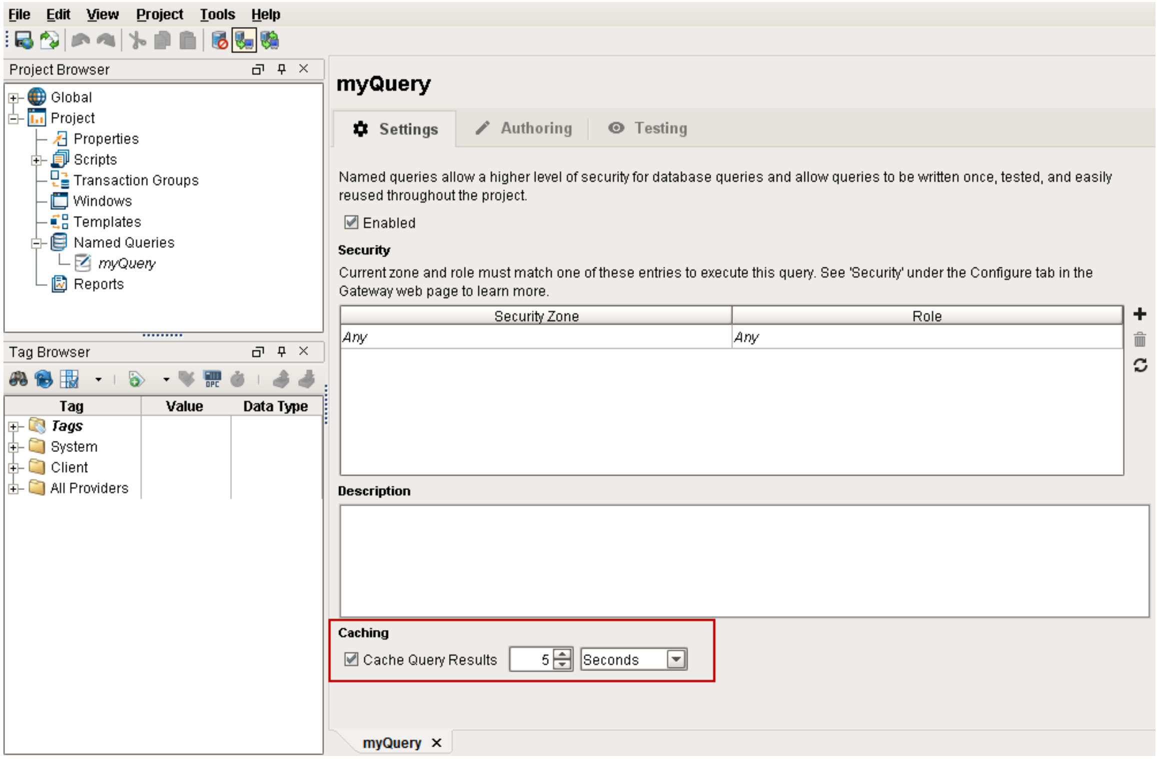
Name	Driver	Status	Connections	Throughput
DB	MySQL ConnectorJ	✓ Valid	0 / 8	0.3 queries/sec

Use Cached Named Queries When Possible

Most resources in Ignition that can request a query will not cache the results for use by other resources: a SQL Query Binding that returns a result set will only do so for the one component, and can't be utilized by other resources in the same project. Thus, if two clients have the same window, the same query must be fired twice for both bindings to receive information from the database which is wasteful.

Named Queries are the exception. They can cache the resulting dataset for use by other resources in the project, as well as other instances of the same project. In the previously mentioned scenario, one client would trigger the Named Query to execute, and the other client would simply utilize the cached result set, reducing the number of queries running against the database.

If your project contains queries that poll slowly, or results sets that aren't frequently modified, then a Named Query with caching enabled is an efficient alternative to a SQL Query or DB Browse binding.



myQuery

Named queries allow a higher level of security for database queries and allow queries to be written once, tested, and easily reused throughout the project.

☑ Enabled

Security

Current zone and role must match one of these entries to execute this query. See 'Security' under the Configure tab in the Gateway web page to learn more.

Security Zone	Role
Any	*Any*

Description

Caching

☑ Cache Query Results 5 Seconds

Use the Expression Language to Consolidate Multiple Queries

If multiple resources (such as multiple Tags, or multiple components) need separate values from the same database table, or a window contains multiple components that are all querying data from the same table, such as multiple Numeric Labels, it may be more efficient to have a single query run and fetch the large portions of the table, and then distribute the individual values to each component. This typically involves having some property or Tag query for several rows of data from a database table, and then using expression bindings or Expression Tags to retrieve individual values from the query.

Individual values may be retrieved from a dataset via the Expression Language: either an Expression binding on a property, or an Expression Tag. Here are two commonly used approaches to extracting a single value from a dataset in the Expression Language:

- The Expression Language's Dataset Access syntax. You may want to use the try() function in case the dataset is empty:

Pseudocode - Expression Language Dataset Access Syntax

```
{dataset}[rowIndex,columnIndex]
```

- The lookup function:

Pseudocode - Expression Language Lookup Function

```
lookup({dataset}, lookupValue, noMatchValue)
```

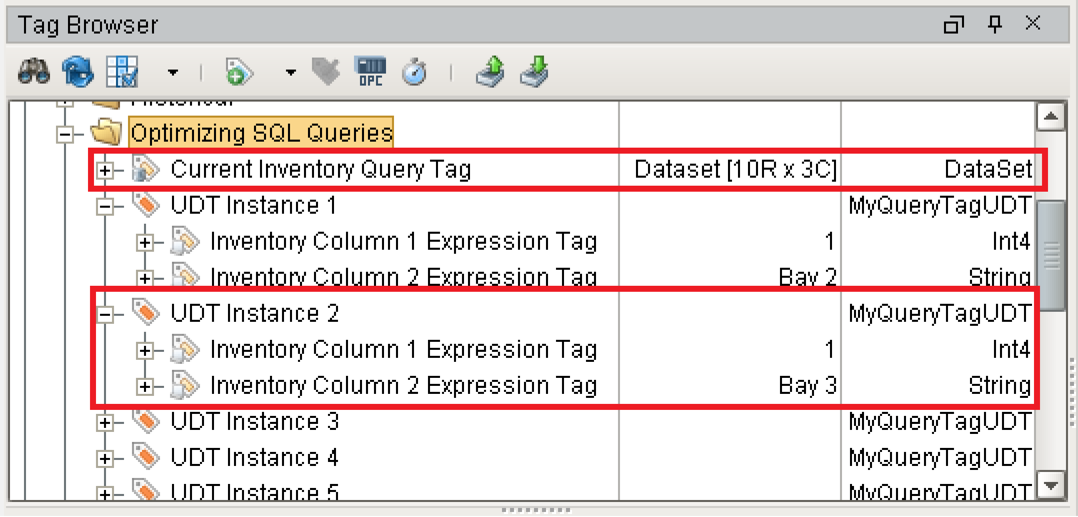
Restrict the Number of Query Tags in a UDT

Each Query Tag in a UDT will run a separate query per instance of the UDT. Assuming a scan class of one second, if a UDT definition contains 5 Query Tags, and there are 5 instances of that UDT, then there will be 25 queries executing every second.

As mentioned on this page, the Expression Language can be used to reduce load on the database if multiple Query Tags are retrieving data from the same database table. Furthermore, UDT parameters can be utilized in the Expression Tags, so new UDT Instances can easily be configured to look up the appropriate values.

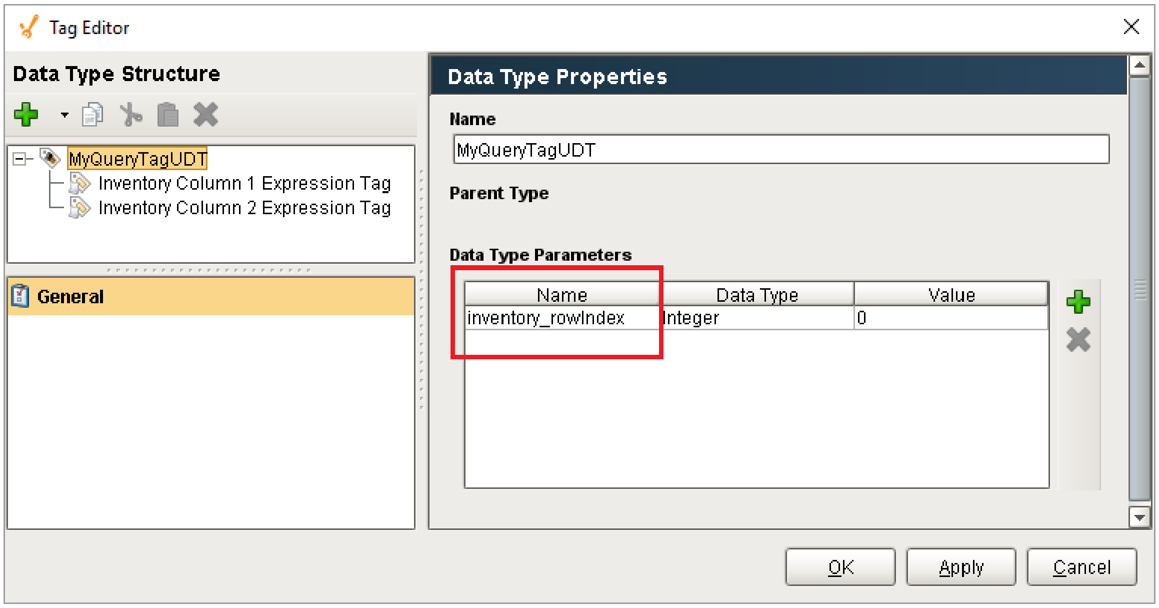
Single Database Tables

Below, the Tag named **Current Inventory Query Tag**, as the name implies, is a Query Tag retrieving multiple rows of data from a database table. We see that the highlighted **UDT Instance 2** contains two members: **Inventory Column 1 Expression Tag** and **Inventory Column 2 Expression Tag**, which are simply Expression Tags that are referencing individual cells from the **Current Inventory Query Tag**.



Optimizing SQL Queries		
Current Inventory Query Tag	Dataset [10R x 3C]	DataSet
UDT Instance 1		MyQueryTagUDT
Inventory Column 1 Expression Tag	1	Int4
Inventory Column 2 Expression Tag	Bay 2	String
UDT Instance 2		MyQueryTagUDT
Inventory Column 1 Expression Tag	1	Int4
Inventory Column 2 Expression Tag	Bay 3	String
UDT Instance 3		MyQueryTagUDT
UDT Instance 4		MyQueryTagUDT
UDT Instance 5		MyQueryTagUDT

The UDT definition can use a parameter to specify an individual row in the Query Tag that each instance should focus on.



MyQueryTagUDT
Inventory Column 1 Expression Tag
Inventory Column 2 Expression Tag

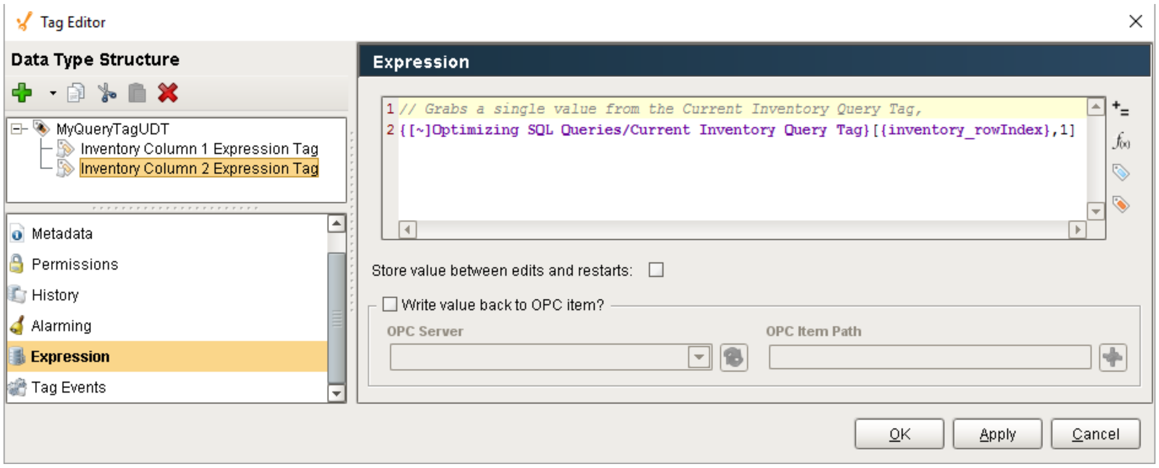
Name: MyQueryTagUDT

Parent Type

Data Type Parameters

Name	Data Type	Value
inventory_rowIndex	Integer	0

Each Expression Tag could use an expression like the following to look up individual values:



Tag Editor

Data Type Structure

- MyQueryTagUDT
  - Inventory Column 1 Expression Tag
  - Inventory Column 2 Expression Tag

Metadata
Permissions
History
Alarming
Expression
Tag Events

Expression

```
// Grabs a single value from the Current Inventory Query Tag,
{[~]Optimizing SQL Queries/Current Inventory Query Tag}[{inventory_rowIndex},1]
```

Store value between edits and restarts: □

□ Write value back to OPC item?

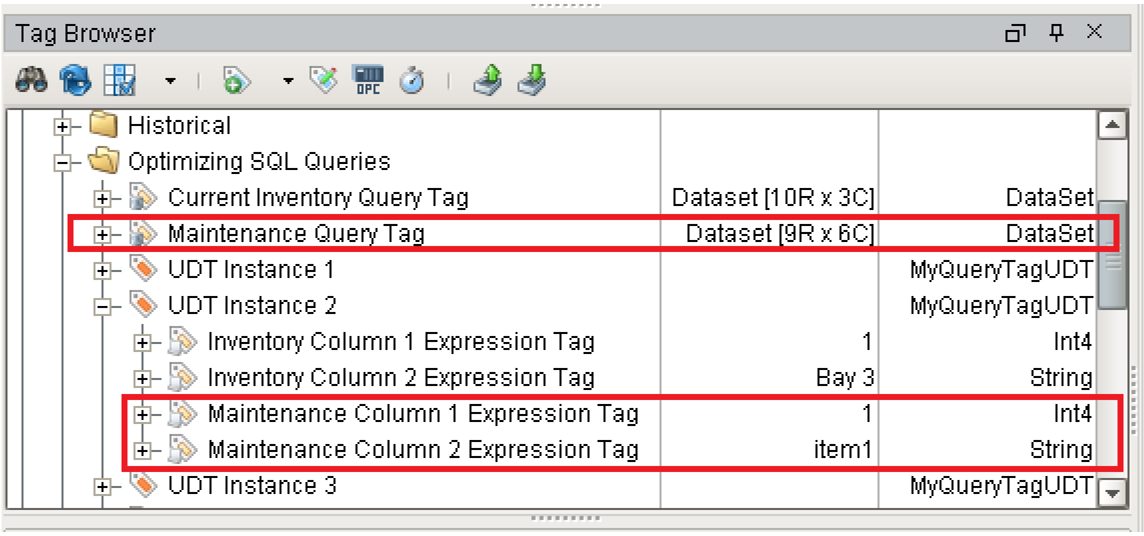
OPC Server

OPC Item Path

Multiple Database Tables

To add values from a separate database table, we simply need a separate Query Tag. In the image below, a new Tag named **Maintenance Query Tag** has been added, which is querying from a separate Database table. To incorporate this new data into our UDT instances, new Expression Tags have been added (**Maintenance Column 1 Expression Tag**, and **Maintenance Column 2 Expression Tag**) that simply reference specific values in the new Query Tag. Now, regardless of how many UDT instances exist in the Tag provider, we only have two Tags that are executing queries against the database.

For each separate table, we need to incorporate a single new Query Tag to collect all of the rows we want to show, add index parameters to the UDT definition, and add Expression Tags to our UDTs.



Tag Browser

Historical		
Optimizing SQL Queries		
Current Inventory Query Tag	Dataset [10R x 3C]	DataSet
Maintenance Query Tag	Dataset [9R x 6C]	DataSet
UDT Instance 1		MyQueryTagUDT
UDT Instance 2		MyQueryTagUDT
Inventory Column 1 Expression Tag	1	Int4
Inventory Column 2 Expression Tag	Bay 3	String
Maintenance Column 1 Expression Tag	1	Int4
Maintenance Column 2 Expression Tag	item1	String
UDT Instance 3		MyQueryTagUDT

Related Topics ...

- Named Query Caching
- Slow Queries

Scripting

What Is Scripting?

Most of the time when we talk about "scripting" in Ignition we are talking about Python scripting, or writing code in the Python language. Python is a general purpose programming language that was developed in the early 1990s and has gained significant popularity in the 2000s. It is extremely readable, elegant, powerful, and easy to learn. As an added bonus, it gracefully interacts with Java, giving programmers an extremely powerful tool when paired with Ignition, which is written in Java.

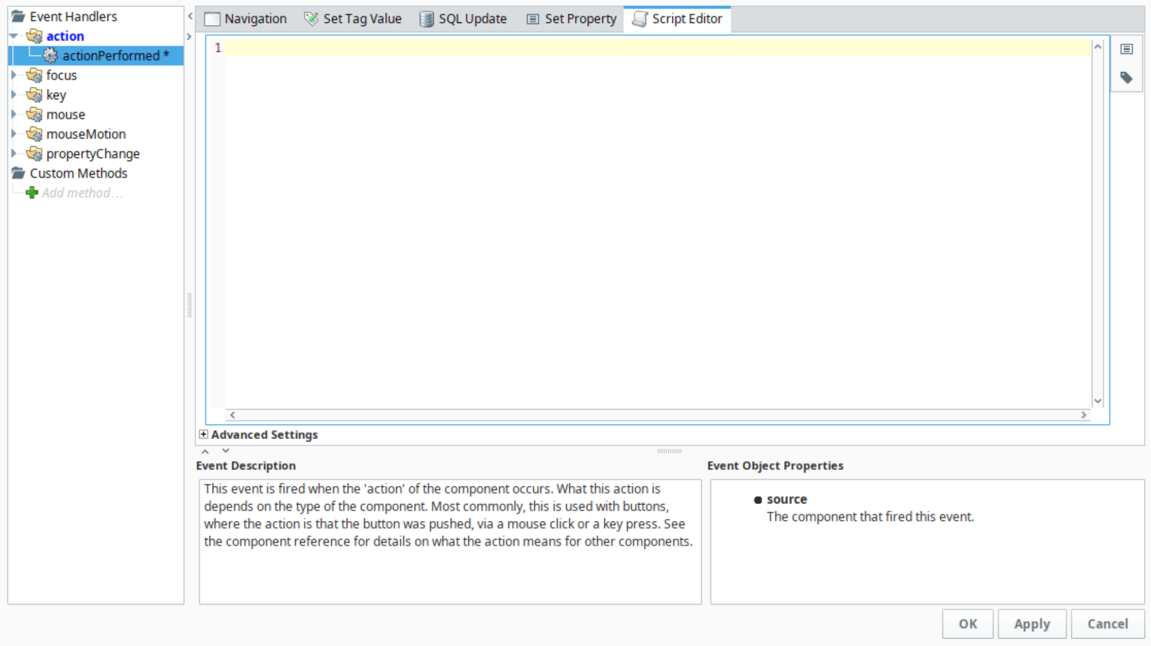
In Ignition, you will be mixing the core Python language with references to other components and a variety of our built-in system functions.

On this page

...

Is it Easy to Use?

Luckily, Python is a simple language to get started with. Using it in an event-driven system takes away a lot of the extra code that normally makes programming time consuming. For those that are already familiar with scripting (and those of you that are learning), we have a huge list of functions inside Ignition to do some of the common tasks in a single line of code. These System Functions are available while typing. Just start with "**system**" and press "ctrl-space" to see a list of available functions. The list will filter itself as you continue typing.



Event Description

This event is fired when the 'action' of the component occurs. What this action is depends on the type of the component. Most commonly, this is used with buttons, where the action is that the button was pushed, via a mouse click or a key press. See the component reference for details on what the action means for other components.

Event Object Properties

- **source**
  The component that fired this event.

Where Is Scripting Used?

Python is used in many places in Ignition. Each location has its own events that trigger your scripts to run, and add functionality to your projects in different ways.

- **Components** - Add actions to components like buttons, customize the look and feel of charts and tables, and even setup a custom navigation schema.
- **Tags** - Create a script that runs on a Tag change and when an alarm goes active!
- **Reports** - Use scripting to create a customized datasource, or create your own unique action to use with the scheduling system.
- **Alarm Notification** - Create custom rosters using scripting to dynamically change who gets notified with each new alarm event.
- **Client, Session, and Gateway** - Add a script that will run when certain events happen, such as when the Client or Gateway starts up, or on certain keystrokes.

Users that are new to Ignition focus mainly on the component binding system in Vision or Perspective, and for good reason. It's simple, flexible, and generally easy to understand without much of a computer background. However, Ignition has a complete scripting system built into every place you can think of. Using it is not a requirement, but it can add a significant degree of flexibility and customization to your projects. It allows you to create exactly what you need, giving you total control where pre-canned options fall short.

The majority of your scripting will be done in Event Handlers inside of components. This system makes it very easy to get started scripting with little to no experience. With the script builders, basic scripting like Navigation and setting Tag values takes just a few clicks. See also Script Builders in Vision and Component Events and Actions.

Related Topics ...

- Getting Started with Scripting in Ignition
- Gateway Event Scripts
- Client Event Scripts
- Tag Event Scripts
- Scripting in the Report Module

In This Section ...

Python Scripting

About Python

While it is entirely possible to create a complete and powerful project in Ignition without writing a line of script, many designers will find that in order to complete projects with specific requirements, they need to learn at least a little Python. In our experience, most industrial projects involve lots of very complex and specific requirements.

The good news is that learning Python is easy and enjoyable. Python is one of the most beautiful programming languages we've ever encountered. It is very easy to read - even if you don't know it at all, you will probably be able to understand a basic Python script. It is frequently called "executable pseudocode". If you find yourself doing a lot of scripting, you may want to pick up a basic reference book about Python, or go through a simple online tutorial course.

This section is a short tutorial specifically for python, which should help get you started. It goes over all of the core concepts you will need for scripting in Ignition but then next section (Scripting in Ignition) goes over using Python directly inside Ignition.

- Python Variables, Datatypes, and Operators: Learn what a variable is and how to create it, what the various datatypes are, and what operators you can use on them.
- Conditions and Loops: Learn the common if/else type of statements, as well as loops.
- Functions: Learn about the built-in functions that can do complex work with a simple command, as well as user defined functions, which help compartmentalize the code.
- Libraries: Learn about our built-in system functions, as well as pulling in outside libraries.

On this page

...

Python or Jython?

You'll often hear Python referred to as "Jython" by advanced users of Ignition. Python is the language, Jython is the implementation of the language that we use. Most users of Python use the implementation called "CPython" - they just don't realize it. See http://en.wikipedia.org/wiki/Python_(programming_language)#Implementations.

One of the powerful things about using Jython is that your script has access to the entire Java standard library. In the Client, this will be Java 6 or above. When running under the Gateway, this will be Java 8 and above. For more information, see Accessing Java.

Many scripting users are blown away by their script's speed. We can't take credit for this - the Jython engine hot-compiles (compiles the code when it is run) your Jython code to Java bytecode, which means it runs natively in the JVM, which in turn can hot-compile it to machine code. It's fast.

Which Version of Python is Used?

Ignition uses Jython 2.7. Jython is the Python programming language implemented over the Java Virtual Machine. When looking at outside documentation, such as on www.python.org, verify that you are looking at the correct version.

Jython 2.7 allows us to use the standard functions and tools in Python 2.7, so if you want to look up something in the Python docs, make sure to use version 2.7 (https://docs.python.org/2/).

Scripting Basics

Python is very easy to learn, and with some understanding of its basic syntax, you can get started making your own scripts.

INDUCTIVE UNIVERSITY

Basic Python - Variables and Comments

Hello World

Let's get right to everyone's favorite example, "Hello World." The following script will print out `"Hello World"` to the output console.

The `print` keyword is a handy tool in Python, allowing you to write text to the output console. This is useful for debugging your scripts. You can print multiple things by separating them with commas.

Watch the Video

Variables

Variables are created by simply assigning a value to them. Variables do not need to be declared, because Python has a dynamic type system. That means Python figures out the type of the variable on the fly when the script is executed.

The following script would print out: 15

Python - Using Variables

```
x=5
y=3
print x*y
```

Quotations

Python makes limited distinction between single and double quotes. As long as they are used consistently then there are few times when the type of quotation mark you use matters. Some of the rules are shown here:

Python - Using Single and Double Quotes

```
print "This is my text"           # Using double quotation marks
print 'This is my text'                # Using single quotation marks
print "This is my text'                # This will not work because
python does not allow mixing the single and double quotation marks
print "My name is 'David'"        # This will print: My name is
'David'
print 'My name is "David"'        # This will print: My name is
"David"
print 'My name is Seamus O\'Malley' # This will print: My name is
Seamus O'Malley
```

Whitespace

Perhaps Python's most unique feature is logical blocks which are defined by an indentation in Python. A colon (:) starts a new block, and the next line must be indented (typically using a tab or 4 spaces). The block ends when the indentation level returns to the previous level. For example, the following will print out `"5 4 3 2 1 Blast-off" with each value on a new line.` The final print is not part of the while loop because it isn't indented.

Python - Logical Blocks / Indentation

```
countdown = 5
while countdown > 0:
        print countdown
        countdown = countdown - 1
print "Blast-off!"
```

Multi-Line Strings

You can start and end a section of comments with a triple quote (''' or """).

Python - Multiple Lines of Comments Using a Triple Quote

```
'''
This is a lot of text
that you want to show as multiple lines of
comments.
Script written by Professor X.
Jan 5, 1990
'''
print 'Hello world'
```

Comments

Comments are a way to document your Python script. There are several ways to use comments, but the best advice we can give is to use them as much as possible! There are a few ways to make a comment in Python.

Individual Lines

You can start a line with a pound/hash (#) sign, or put one anywhere in a normal line of code.

Python - Document Scripts Using Comments

```
# this is a comment
print 'Hello world' # this is also a valid comment
```

Blocks of Lines

While Python doesn't explicitly have a way to block comment (comment out multiple lines), multi-line strings are functionally similar.

Comment Many Lines with a Keyboard Shortcut

In Ignition, you can use the Ctrl-/ keyboard shortcut to comment several lines of code at once. Just highlight one or more lines of code and hold the Ctrl key and press "/". This will prepend all of the selected lines of code with the pound/hash (#) sign. Press Ctrl-/ again to remove the pound/hash sign.

Control Flow

Control Flow statements, that is the ifs and loops, make the language do things differently based on the various conditions. Python has all of the basic control flow statements that you'd expect from a programming language.

Basic Python - Flow Control

Watch the Video

If and Else

An if statement allows you to check if a condition is true or not true. Depending on the condition, you can either execute a block of code, or do something else. Many of these can be chained together to determine under what conditions should certain code execute.

Pseudocode - If Statement

```
if condition == True:
        print value1
```

For and While

Looping can be done with a for, which executes a block of code a set number of times, or a while, which executes a block of code until a certain condition is met. Both can be incredibly useful.

Pseudocode - For Statement

```
for item in myList:
        print item
```

Related Topics ...

- Scripting in Ignition
- Getting Started with Scripting in Ignition

In This Section ...

Variables, Datatypes, and Objects

Overview

Without incorporating any Ignition-specific objects, it is important to understand the basics of Python. This section seeks to introduce three main Python principles: Variables, Data Types, and Operators. Additionally, this page and the sub-pages herein will provide plenty of examples to get you started.

This section of the manual attempts to introduce these principles so that they may later be used in conjunction with Ignition. The information in this section is far from comprehensive, as Python's official documentation is a better reference for all things Python. However, this page serves as a great way to jump right into the action.

Variables

Variables are created by simply assigning a value to them. Variables do not need to be declared, because Python has a dynamic type system. That means, Python figures out the type of the variable on the fly when the script is executed.

The following script would print out: 15

Python - Declaring and Assigning Variables

```
x=5
y=3
print x*y
```

A space may be included on either side of the assignment operator ('='), but is not required. Thus, the following example would be functionality identical as the example above.

Python - Declaring and Assigning Variables with Spaces

```
x = 5
y = 3
print x * y
```

INDUCTIVE UNIVERSITY

Python Variables

Watch the Video

Built-in Datatypes

Python features several built-in datatypes. Below is an overview and links to appropriate pages where applicable.

INDUCTIVE UNIVERSITY

Python Datatypes

Watch the Video

None

There is a special value in Python called `None` (with a capital N). This is simply a special value that means no value. This value is equivalent to Java's `null` value. None can be used to initialize a variable, but is best when checking to see if something exists before doing extra work:

Python - None

```
# If some value is not equal to None...
if something != None:
        # ...then do some work
        doWork()
```

Booleans

Python has two built-in values to represent true and false values: `True` and `False`, respectively (note the capital letters). These can be used when testing for truth, and are implicitly returned when the comparison operator is used:

Python - Booleans

```
# Prints True
print 1 == 1

# Prints False
print 1 == 0
```

When using Booleans, suchs as predicates in an if-statement, you typically don't have to use True or False directly. Instead, many other values are considered True or False. The following values are also considered False:

- None
- Numeric values of 0, such as `0`
- Empty Sequences and Dictionaries, like [] or {}

Values aside from the ones mentioned above are considered true, so you can easily utilize the existence of a non-zero value as a True. Note that for integers both positive and negative numbers are True, only a value of 0 is False.

Working with Different Datatypes

Watch the Video

Strings

Literal strings can be typed in using either double quotes or single quotes. This can be handy when your string contains one quote or the other. You can also use the backslash character to escape special characters including these quotes. See the Strings page for more information.

Numeric Types

Numbers can just be typed in normally, like `42` or `3.14159` . Adding a decimal point differentiates an Integer from a Float. More information on Numeric types can be found on the Numeric Types page.

Colors

Working with colors in Python is remarkably easy. You can simply use any tuple of 3 or 4 integers to represent a color in RGB or RGBA. For example, to set a label's text color to red, you can simple do something like this:

Python - Tuple as Color

```
label = event.source
label.foreground = (255,0,0) #(red,green,blue)
```

Additionally, the system.gui.color function allows you to pick a color in a similar fashion:

Python - System Function as Color

```
newColor = system.gui.color(255,0,0)
```

Lists and Tuples

Python offers a variety of sequence types: most notably Lists and Tuples. These are ordered collections, meaning they are indexed and the sorted order is maintained. More information on these types can be found on the Lists and Tuples page.

Python - System Function as Color

```
newList = [1,2]
newTuple = (2,5,7)
```

Dictionaries

A Dictionary is a very useful type that holds a set of key-value pairs. Unlike sequences, they are not ordered, so there is no need to sort them. Instead you give each Value in the dictionary a Key, which handles as a reference to Value. You may have used these in other languages and know them as hashmaps, maps, associative memories, or associative arrays. More information on Dictionaries can be found on the Dictionaries page.

Python - System Function as Color

```
newDictionary = {"itemName":5}
```

JSON

JSON stands for JavaScript Object Notation. While it is not a datatype, it is a way of defining data in a human readable format, and is commonly used in many applications. It comes from the Javascript programming language, but it is language independent. Each JSON object contains lists and objects. A list is a series of ordered values separated by commas that is commonly used within Python. The object works like a dictionary, using any number of name/value pairs, where each value could be any basic datatype, a list, or even another object with its own name/value pairs. Because JSON is just a way of defining data, it can be used in many different programming languages, including Python. This makes it a useful tool for defining data in a way that humans can easily read.

Ignition also has two scripting functions that allow you to convert between a JSON string and a native Python object: system.util.jsonEncode and system.util.jsonDecode.

Official Documentation

For more information on JSON, see http://www.json.org.

Datasets

A Dataset is a multidimensional collection of values, stored in a manner similar to how values on a spreadsheet appear. Python does **not** natively have a Dataset type. Instead these datasets were created for use inside of Ignition. There are two types of datasets:

- Dataset: Sometimes called a "Standard Dataset", this type is commonly used on many Vision Components, such as the Power Table or Chart, to display multiple values simultaneously.
- PyDataset: Short for "Python Dataset", these datasets act in a manner very similar to a Python Sequence when it comes to accessing specific values, or iteration (see Lists and Tuples). Ignition's built-in system functions that interact with the database typically return a PyDataset.

While there are two types of datasets, you can easily convert one type of dataset to the other. Additionally, you can easily create a dataset from a script.

Python - Creating a Dataset

```
header = ["The Only Column"]
rows = [[1],[2],[3]]

myDataset = system.dataset.toDataSet(header, rows)
```

Dates

Dates and times can be created in Python with the `datetime` and `time` libraries. However, the Ignition's built-in system.date functions can also be used instead without having to import either library.

Python - Show the Current Datetime

```
currentTime = system.date.now()

print "The current time is: %s" % currentTime
```

Using the In-Keyword

Python's `in` keyword can be used to check the contents of something for a specific instance of another object. One use is to check the contents of a string for a certain substring:

Python - Using In Keyword to Look for a Substring

```
myString = "Hello World"
subString = "World"

# Here we are using an if-statement to look for the substring. If the substring exists inside of
myString, the expression returns True.
if subString in myString:
        # This will print out, because the substring "World" exists in the string "Hello World"
        print "The word 'World' appears in myString"
```

The `in` keyword can be used to look for a certain object inside of a sequence:

Python - Check for the Presence of an Object

```
myList = [1,2,3,4]

# If the integer 3 exists in myList, then this will print True, otherwise it will print False.
print 3 in myList
```

Python - Check for the Presence of an Object

```
# Check for the Administrator role.
if "Administrator" in system.security.getRoles():
        print "Administrator found"
```

Additionally, the `in` keyword can be used in an expression on a `while` loop:

Python - Check for the Presence of a Keyword in an Expression

```
myList = [1,2,3,4]

# As long as the integer 4 is in myList, the while loop will continue to iterate.
while 4 in myList:
        # pop(0) will remove the first element in the list
        myList.pop(0)
else:
        print "all done!"
```

Basic Operators

Python has many common operators as you would expect, or are at least familiar with if you've worked with other scripting languages.

These are just the basics. There are other operators, like bit shift operators and more. Read about them at: http://docs.python.org/library/stdtypes.html

Operator Reference

Arithmetic Operators

In regards to Arithmetic Operators, the precision of the returned value depends on the data types of the arguments: including at least one **float** will return a **float**, otherwise, an **integer** is returned.

Operator	Meaning	Example	Output
+	Addition. Note that the data type of the returned object depends on the datatype of the arguments being used:	`print 5 + 9`	14

		`print 5.0 + 9`	`14.0`
`-`	Subtraction	`print 5 - 9`	`-4`
`*`	Multiplication	`print 5 * 9`	`45`
`/`	Division	`print 10 / 4.5`	`2.2222222222222223`
`//`	Floored Division: will divide and return just the nearest integer value, even if dividing floats.	`print 10 // 4.5`	`2.0`
`%`	Modulo: returns just the remainder of the dividend (the left argument) after being divided by the right argument. In the example on the right, the 4.5 divides evenly twice, leaving 1.0 as a remainder.	`print 10 % 4.5`	`1.0`
`**`	Power: raise the the number on the left to the power of the number on the right.	`print 5 ** 3`	`125`
Boolean Operators			
Operator	**Meaning**	**Example**	**Output**

`or`	Returns True if either argument is True. If both are False, then returns False.	`x = False`
		`y = True`
		`print x or y`
			`True`

`and`	Returns True only if both arguments are True. Otherwise, returns False.	`x = False`
		`y = True`
		`print x and y`
			`False`

not	Returns a boolean value that represents that opposite value of the trailing expression: False becomes True, and True becomes False.	`x = True`
		`print not x`
			`False`

Comparison Operators			
Operator	**Meaning**	**Example**	**Output**
`<`	Less than	`print 10 < 5`	`False` `False`

		`print 10 < 10`	

`<=`	Less than or equal to	`print 10 <= 5`
		`print 10 <= 10`
			`False`
			`True`

`>`	Greater than	`print 10 > 5`
		`print 10 > 10`
			`True`
			`False`

`>=`	Greater than or equal to	`print 10 >= 5`
		`print 10 >= 10`
			`True`
			`True`

`==`	Equal	`print 10 == 10`
		`print 10 == 0`
			`True`
			`False`

`!=`	Not equal	`print 10 != 10`
		`print 10 != 0`
			`False`
			`True`

`is`	Returns True if both arguments are referring to the same object, otherwise, returns False.	`x = 10`
		`y = 5`
		`print x is y`
			`False`

`is not`	Returns True if both arguments are referring to different objects.	`x = 10`
		`y = 5`
		`print x is not y`
			`True`

Order of Operations

Note that there is an order of operations for Arithmetic and Boolean operators that can be modified with parenthesis. For arithmetic they are grouped in tiers and evaluated left-to-right within a given tier:

- Parenthesis
- Exponents
- Multiplication, Division, Modulo
- Addition, Subtraction

Python - System Function as Color

```
6+10*2   # produces 6+(10*2) = 26
6*10+2   # produces (6*10)+2 = 62

10*6%2   # produces (10*6)%2 = 0
10%6*2   # produces (10%6)*2 = 8
```

For Boolean operators:

- Not
- And
- Or

Related Topics ...

- Conditions and Loops

In This Section ...

Numeric Types

The Differences Between the Types

Within Python, there are a few numeric types that we will often use within the scope of Ignition: integers, floats and booleans. **Integers** have at least 32 bits of precision and consist of non decimal values (1, 2, 50, 246). When working with larger numbers, Python also supports a **long** type that has unlimited precision. **Floats** consist of numbers with decimal values (1.0, 2.4, 50.7, 246.8734). **Booleans** are a unique subset of integers. They are either **True** or **False**, but can also be represented using a 1 or 0, respectively.

On this page

...

Numbers in Arithmetic

When using numeric values in expressions or equations, Python is very flexible in how it handles multiple types in the same operation. When using two of the same type in an expression, the outcome will always be of that type. Adding two integers together yields an integer, while multiplying two floats will yield a float. With division, using integers will always result in a floor division so that the answer remains an integer. Floor division will simply remove the decimal, and will not round up at any point.

When using different types, the more precise type will be used for the outcome. Doing an operation with one integer and one float will produce an outcome that is a float. Adding a float and an integer together will yield a float.

Below, we have some examples that show off some of the different ways that Python handles expressions. You can try these out in the interactive interpreter of the Script Console.

Type of Arthimetic	Description	Example	Output

Addition and Subtraction	Adding/Subtracting integers will return an Integer.
	If at least one of the numbers is a float, then the resulting number will also be a float. Otherwise the result will be an integer
		`4 + 5 # int + int = int`
		`4.0 + 5 # float + int = float`
		`4.0 + 5.3 # float + float = float`
			`9`
			`9.0`
			`9.3`

Multiplication	Similar to Addition and Subtraction, multiplication will only return a float if at least one of the numbers is a float	`4 * 5 # int * int = int`
		`4 * 5.0 # int * float = float`
			`20`
			`20.0`

Division	Dividing an integer by another integer results in floor division: meaning the remainder of division is never returned. Note that this is not the same as rounding to the nearest integer.
	To include the remainder, at least one of the numbers needs to be a float
		`4 / 5 # int / int = int`
		`4 / 5.0 # int / float = float`
			`0`
			`0.8`

Operators and Functions

Python offers a wide range of operators that can be used to perform calculations on numeric values. Additionally, there are many functions that can perform specific operations to numeric values. The tables below contain some of the most common operations and functions, but this is not an exhaustive list.

Operators

Operator	Description	Example	Types that use the Operator
+	Used to find the sum of two numeric values Can also be used to concatenate string values	5 + 1 = 6 "Hello " + " World"	Numeric Types, Strings
-	Used to find the difference of two values	5 - 1 = 4	Numeric Types
*	Used to find the product of two values	5 * 2 = 10	Numeric Types
/	Used to find the quotient of two values	4 / 2 = 2	Numeric Types
%	Modulus operator. When used with numeric types, returns the remainder of the quotient of two values The operator is also seen in string formatting.	5 % 2 = 1	Numeric Types, Strings
**	Used to find a value to the power of another value	5 ** 2 = 25	Numeric Types
Functions			
Function	**Description**	**Example**	
int(x)	Converts x to an integer	int(4.8) = 4	
long(x)	Converts x to a long	long(5.7) = 5	
float(x)	Converts x to a float	float(4) = 4.0	
abs(x)	Absolute value of x	abs(-4.6)=4.6	
round(x[, n])	Rounds x to n digits. If n is not specified, default is 0	round(5.674)=6.0 round(5.674, 2)=5.67	

Related Topics ...

- Strings
- Lists and Tuples
- Dictionaries
- Datasets
- Dates

Strings

What are Strings?

In Python, as well as most other programming languages, a string is a grouping or string of characters that are used literally instead of as names of variables or other objects. In Python, they can be enclosed in either single quotes or double quotes:

Python - Print Statements

```
# These two print statements will both print out a sentence that
looks the same as the other.
print 'These will both print the same!'

print "These will both print the same!"
```

Strings are also considered sequences, which means they can be iterated through much like a list.

On this page ...

String Escape Character

Strings use a special character called an "escape" character to denote when something different should be happening within the string. In Python, the escape character is the backslash (\ ). The escape character is mainly used in a few different ways:

The Escape Character and Quotation Marks - "" and ''		
Comments	**Example**	**Output**
Here, we use the escape character to let Python know we want to ignore the second single quote and directly print it. This works with either single or double quotes.	`print 'doesn\'t'`	`doesn't`
Alternately, we can enclose the string in the opposite quotes that we are using in the string. Because we enclose the string in double quotes here, the single quote prints normally.	`print "doesn't"`	`doesn't`
This also works in reverse, where we can enclose the string in single quotes and use double quotes within.	`print '"Test", he said.'`	`"Test", he said.`
Of course, we can also use the escape character with double quotes enclosing the string.	`print "\"Test\", he said."`	`"Test", he said.`
New Lines and Tab Spacing		

A new line may be specified with the escape character and "n".
 	
Alternatively, a multi-line string may be used to add line breaks to a string literal.
	`print "First line\nSecond line"`	`First line`
		
		`Second line`

Tab-spacing may be added to a string literal with the escape character and "t"	`print "Hello \tWorld"`	`Hello` `World`
Specifying a Backslash		
If you need to place a backslash inside of a string literal, simply use the escape character twice.	`print "Folder\\file.txt"`	`Folder\file.txt`

Multi-Line Strings

If a string literal should contain multiple lines, you can use a multi-line string, which is created by using three quotation marks.

Example	Output

`print '''This`
`is`
`on`
`multiple`
`lines'''`
	`This`
	`is`
	`on`
	`multiple`
	`lines`

This is useful when using the system.db functions, as full SQL queries are typically passed to these functions, so using a multi-line string allows you to write the query in an easy to read format.

Python - Multi-line String

```
query = '''        SELECT * FROM myTable
                        WHERE id < 10'''

system.db.runPrepQuery(query,[])
```

Raw Strings

Sometimes, it is necessary to print the raw string without allowing escaped characters. This is done by placing the letter "r" in front of the string.

Python - Raw String

```
print r'This string \n will print out directly as written.'
```

This is especially useful in cases where a file path needs to be hard coded as a string literal:

Python - File Path as a Raw String

```
# Specifying a Windows file path
myPath = r"C:\Folder\Another Folder\file.txt"
print myPath

# Specifying a Linux file path
myPath = r"/home/Directory/Another Directory/file.txt"
print myPath
```

Unicode Strings

Strings that contain characters beyond 7-bit ASCII, such as é or need to be marked as unicode strings by placing the letter u in front of the string.

Python - Unicode String

```
print u'été'
```

Combining Strings

Two different string type variables can actually be combined or concatenated using the plus (+) sign. It is important to understand that they are concatenated exactly.

Python - Concatenated String Type Variables

```
a = "this"
b = "that"

# Will print 'thisthat' because there was no space at the end of a or the beginning of b.
print a + b

# Will print 'this that' because we added a space between them.
print a + " " + b
```

String Indexing

Strings in Python are actually indexed, with the first character having an index of 0. To grab a value at a specific index, you place a value within square brackets ([ ]) after the string variable.

Python - Indexed Strings

```
a = "Ignition"

# Will print out 'I', since it is in the zero position.
print a[0]

# Will print out 't', since it is in the fourth position.
print a[4]
```

You can also use a negative index, which will start from the right side of the string with the last character having a index value of -1. This is useful for getting the last character when you aren't sure how long the string is.

Python - Negative Indexed String

```
a = "Ignition"

# Will print out 'o', since it is the second to last character.
print a[-2]
```

Slicing Strings

Using the string index values, we can actually grab slices or parts of the string. Similar to grabbing an individual character, we place two values separated by a colon within square brackets. You can think of these numbers as the slices between characters, with 0 before the first character, 1 before the second, and so on. It should look like this: string[4:7] The first value is the start location, while the second value is the end location.

Python - Sliced Strings

```
a = "Inductive Automation"

# Will print out 'ctive Auto'. Note how the space is counted as a character,
# and the 14th location is before the m, which is not included.
print a[4:14]
```

If left blank, the first value will default to 0, while the second value will default to the length of the string. Additionally, negative values can be used as well, just like with the index.

Python - Negative Value Sliced Strings

```
a = "Inductive Automation"

# From index 0 to index 12.
print a[:12] # Will print out: Inductive Au

# From index 12 to the end of the string.
print a[12:] # Will print out: tomation
```

```
# From the 4th character from the end to the 5th index. Note this prints nothing because the start
character is after the end character.
print a[-4:5] # Will print out:

# From the 8th character from the end to the 2nd character from the end.
print a[-8:-2] # Will print out: tomati

# From index 5 to the 5th character from the end.
print a[5:-5] # Will print out: tive Autom
```

You can use all of these together (and with integer variables or functions) to have a very flexible way to format strings.

String Formatting

Also known as string substitution, it allows you to enter in values into a string, similar to having a variable inside of the string. This is useful when doing a database query where you can build the query as a string, and then add in the query parameters when executing the query.

Doing this requires the use of the percent (%) sign followed by the parameter type within the string. After the string, you then use the percent sign followed by the parameter values within parentheses in order.

Python - String Substitution

```
a = "I have %i apples."

# Will print exactly as written, because we are not substituting any values in.
print a

# Will print with an 8 substituted in for the %i. I have 8 apples.
print a % (8)
```

Instead of using literal values, you can also use variables as parameter values instead. This is useful when pulling the value from a function. Each string can also have multiple values substituted, as long as the parameters are in the order that they are in the string.

Python - String Substitution using Variables as Parameters

```
apples = 10
oranges = 14
peaches = 5

a = "I have %i apples, %i oranges, and %i peaches."

# Will print with the variable values substituted in for the %i. I have 10 apples, 14 oranges, and 5
peaches.
print a % (apples, oranges, peaches)
```

There are a few different types used with string formatting, the most common of which are listed here. The character values are what go after the percent sign.

Type	Character
Signed Integer	i
Floating Point Decimal Format	f
Floating Point Exponential Format	e
String	s

String Search Example

You can bring all this together for a simple way to print out just the last word in a string.

Python - Find the Last Word of a String

```
# Find just the last word of a string.
myString = "Inductive Automation"
```

```
lastSpaceIndex = myString.rfind(" ")

# Add one to not include the space.
print myString[lastSpaceIndex+1:] # Will print out: Automation
```

String Functions

Python strings have many functions available that can manipulate the string or give information on the string. The most common are in the table below.

Function	Description	Example	Output
len (string)	Returns the length of the **string**.	a = "Inductive Automation" print len(a)	20

x in string	Will return **True** if **x** is within the string, **False** if not.
	
	Can also be used to iterate through the string.
		string = "Inductive Automation"
		
		if 'd' in string:
		
		print "There is a d in the string"
		
		for letter in string:
		
		print letter
			There is a d in the string
			
			I
			
			n
			
			d
			
			u
			
			c
			
			t
			
			i
			
			...

string. find(x, [, start[, end]])	Returns the first location of the substring **x** from the string. Returns -1 if the substring is not found.	string = "Inductive Automation" print string. find(" ")	9
string. rfind(x, [, start[, end]])	Similar to find, but returns the last location of the substring **x**. Returns -1 if not found.	string = "Inductive Automation"	17

<table>
<tr><td></td><td></td><td>print string.find("i")</td><td></td></tr>
<tr><td>string.upper()</td><td>Returns a copy of the string with all characters uppercase. This is useful when comparing user input to a string value, as the user may use a different case for certain letters.</td><td>userInput = "administration"

print userInput.upper()</td><td>ADMINISTRATION</td></tr>
<tr><td>string.lower()</td><td>Returns a copy of the string with all characters lowercase. This is useful when comparing user input to a string value, as the user may use a different case for certain letters.</td><td>userInput = "ADminIStration"

print userInput.lower()</td><td>administration</td></tr>
<tr><td>string.capitalize()</td><td>Returns a copy of the string with the first character capitalized and all other characters lowercase.</td><td>string = "here is my sentence."

print string.capitalize()</td><td>Here is my sentence.</td></tr>
<tr><td>string.title()</td><td>Returns a copy of the string with the first letter of each word capitalized and all other characters lowercase.</td><td>string = "here is my sentence."

print string.title()</td><td>Here Is My Sentence.</td></tr>
<tr><td>string.strip([x])</td><td>Returns a copy of the string with leading and trailing characters removed, where <b>x</b> is the string of characters.. If <b>x</b> is omitted, then removes whitespace from the leading and trailing edges of the string.

lstrip() and rstrip() may instead be used to strip characters from the leading or trailing edge</td><td>string = " This string has some empty space"

print string
print string.strip()

fencedString = "|||||___My String__</td><td>This string has some empty space
This string has some empty space
My String
My String___|||||
|||||___My String</td></tr>
</table>

		`_|||||"`
		
		`print fencedString.strip("|_")`
		
		`print fencedString.lstrip("|_")`
		
		`print fencedString.rstrip("|_")`	

string.count(x[, start[, end]])	Returns the number of occurrences of **x** in the string. A **start** and **end** can be specified that will limit the count to that area.	`string =" Inductive Automation"`
		
		`print string.count('i')`
		
		`print string.count('i', 4, 9)`
			2
			
			1

string.split([delimiter[, maxsplit]])	Returns a list of the words in the string. Optionally, specifying a **delimiter** will split the string on the delimiter string. Specifying a **maxsplit** will split the string a maximum number of times, with the remainder of the string as the final list object. The number of items in the list will not be more than maxsplit + 1	`sentence = "This is an example of split"`
		
		`print sentence.split()`
		
		`dashedSentence = "Yet-another-sentence-here"`
		
		`print dashedSentence.split("-",2)`
			`['This', 'is', 'an', 'example', 'of', 'split']`
			`['Yet', 'another', 'sentence-here']`

string.rsplit([delimiter[, maxsplit]])	Similar to split, but splitting is performed from right to left. Thus if **maxsplit** is smaller than the total number of **delimiter** in the string, only the rightmost words will be split off as separate items.	`sentence = "This is an example of split"`
		
		`print`
			`['This', 'is', 'an', 'example', 'of', 'split']`
			`['Yet-`

		`sentence.rsplit()`
		
		`dashedSentence = "Yet-another-sentence-here"`
		
		`print dashedSentence.rsplit("-",2)`
			`another',`
			`'sentence',`
			`'here']`

string.join(x)	Returns a copy of the string that is the concatenation of the strings in the iterable **x**. The string calling the join is what will separate the values of the iterable.	`a = '-'`
		`b = 'abcdef'`
		
		`print a.join(b)`
			a-b-c-d-e-f

string.replace(old, new [, count])	Returns a copy of the string where occurrences of the **old** substring are replaced with the **new** substring. Optionally, if count is specified, will only replace the first **count** number of occurrences.	`string =" Ignition is good!"`
		
		`print string.replace("good", "awesome")`
			Ignition is awesome!

Related Topics ...

- Numeric Types
- Lists and Tuples
- Dictionaries
- Datasets
- Dates

Lists and Tuples

On this page

...

Sequences

Like Strings, Python has two other common sequence types: Lists and Tuples. While lists and tuples are similar to strings in that they share many of the same functionality, they are unique in that they are used to group other values together. Both lists and tuples define a number of values separated by commas, but lists are enclosed in square brackets **[ ]** and are mutable, meaning their contents can change, while tuples are enclosed in parentheses **( )** and are immutable, meaning their contents can't change.

Lists

As stated above, lists are groups of comma separated values enclosed in square brackets, but can utilize many of the features available to other sequences like strings.

Python - Lists

```
# Lists are very simple to create.
myList = [1, 2, 3]

print myList # Will print out: [1, 2, 3]

# Empty lists can also be created, to have items added to them later.
myList = []

# Lists are not confined to hold values of a single data type either.
myList = [1, "hello", 3.3]

# Lists can even hold other lists! In this case, myList would
actually hold 5 elements:
# 1, a list, 4, 'that', and another list. The last list also contains
a list as well.
myList = [1, ["this", 3.3], 4, 'that', [6, [7.7, 'other'], 9]]
```

Basic Python - Lists and Dictionaries

Watch the Video

List Concatenation

Like strings, lists also support concatenation and can be combined to form a single larger list.

Python - List Concatenation

```
a = [1, 2, 3]
b = ['four', 'five', 'six']

print a + b # Will print out [1, 2, 3, 'four', 'five', 'six']
```

List Indexing

Like strings, lists are also indexed, which allows you to grab single values or splice them to get ranges of values. Just like everything else in Python, lists start with 0.

Python - List Indexing

```
myList = ['a', 'b', 'c', 'd', 'e', 'f', 'g']

print myList[3] # Will print out: d

print myList[-2] # Will print out: f

print myList[2:5] # Will print out: ['c', 'd', 'e']
```

When trying to index a nested list, we simply need to add a second index after the first.

Python - Index a Nested List

```
myList = [1, 2, [3, [4, 5, 6, 7], 8], 9, 10]

print myList[2][1][3]  # Will print out: 7
```

Appending to Lists

You can add values directly to the end of a list with the append() function. You can add anything that is normally allowed in a list.

Python - Appending to a List

```
myList = []

myList.append('Hello')
myList.append('World')

print myList # Will print out ['Hello', 'World]
```

List Functions

Below is a list of common list functions. Some of them are similar to other sequences like strings, while others are unique to lists because lists are mutable.

Function	Description	Example	Output
len(list)	Returns the length of the **list** .	list = [1, 2, 3, 4, 5] print len(list)	5

x in list	Will return **True** if **x** is within the list, **False** if not.
	
	Can also be used to iterate through the list.
		list = [1, 2, 3, 4, 5]
		
		if 4 in list:
		    print "There is a 4 in the list"
		
		for value in list:
		    print value
			There is a 4 in the list
			
			1
			
			2
			
			3
			
			4
			
			5

list.index (x)	Will return the index number of item **x**. Throws an error if item is not found.	list = [1, 2, 3, 4, 5] print list.index (3)	2
min(list)	Returns the smallest item of **list**.	list = [1, 2, 3, 4, 5] print min(list)	1
max(list)	Returns the largest item of **list**.	list = [1, 2, 3, 4, 5] print max(list)	5
list.count	Will return the number of times **x** appears in the list.		2

(x)		`list = [1, 2, 3, 4, 5, 4]`
		`print list.count(4)`	

list.append(x)	Will add **x** to the end of the list.	`list = [1, 2, 3, 4, 5]`
		`list.append(6)`
		`print list`
			[1, 2, 3, 4, 5, 6]

list.insert(i, x)	Will insert **x** at position **i**.	`list = [1, 2, 3, 4, 5]`
		`list.insert(1, 5)`
		`print list`
			[1, 5, 2, 3, 4, 5]

list.remove(x)	Will remove the first **x** from the list.	`list = [1, 2, 3, 2, 4, 5]`
		`list.remove(2)`
		`print list`
			[1, 3, 2, 4, 5]

list.pop([i])	Will remove the item at index **i**, and return it. If no index is specified, it will remove and return the last item in the list.	`list = [1, 2, 3, 4, 5]`
		`list.pop(2)`
		`print list`
		`list.pop()`
		`print list`
			3
			[1, 2, 4, 5]
			5
			[1, 2, 4]

list.reverse()	Will reverse the items in the list.	`list = [1, 2, 3, 4, 5]`
		`list.reverse()`
		`print list`
			[5, 4, 3, 2, 1]

Tuples

Tuples look similar to lists in that they are defined as a group of comma separated values, but they are enclosed by parenthesis and they are immutable like strings, meaning they can't be altered. Besides that, they are sequences, like lists and strings. This means that tuples have the functionality that other sequences have, such as concatenation, indexing and slicing, and even nesting.

Python - Tuples

```
# Tuples are very simple to create.
myTuple = (1, 2, 3)

print myTuple # Will print out: (1, 2, 3)

# Empty tuples can also be created, to have items added to them later.
myTuple = ()
```

```
# Like lists, tuples are not confined to hold values of a single data type either.
myTuple = (1, "two", 3.3)

# Tuples can even hold other tuples!
myTuple = (1, ("two", 3.3), 4, 'five', (6, (7.7, 'eight'), 9))

a = (1, 2, 3)
b = (4, 5, 6)

# Combine two tuples to make a new tuple
print a + b # Will print out: (1, 2, 3, 4, 5, 6)

myTuple = ('a', 'b', 'c', 'd', 'e', 'f', 'g')

print myTuple[2:5] # Will print out: ('c', 'd', 'e')
```

Lists and tuples can also be nested within each other!

Tuple Functions

All of the functions that work on tuples work on lists, so these should look familiar. However, because they are immutable, not all of the list functions work on tuples. Check out the common tuple functions below.

Function	Description	Example	Output

len(tuple)	Returns the length of the **tuple** .	`tuple = [1, 2, 3, 4, 5]`
		
		`print len(tuple)`
			5

x in tuple	Will return **True** if **x** is within the tuple, **False** if not.
	
	Can also be used to iterate through the tuple.
		`tuple = [1, 2, 3, 4, 5]`
		
		`if 4 in tuple:`
		`        print "There is a 4 in the tuple"`
		
		`for value in tuple:`
		`        print value`
			`There is a 4 in the tuple`
			
			1
			
			2
			
			3
			
			4
			
			5

min (tuple)	Returns the smallest item of **tuple** .	`tuple = [1, 2, 3, 4, 5]`
		
		`print min(tuple)`
			1

max (tuple)	Returns the largest item of **tuple** .	`tuple = [1, 2, 3, 4, 5]`
		
		`print max(tuple)`
			5

Related Topics ...

- Numeric Types
- Strings
- Dictionaries
- Datasets
- Dates

Dictionaries

On this page ...

Mapping Type

A Dictionary is a mapping object. Where sequences are indexed with a numeric index value, dictionaries are indexed using keys. These keys then have a matching value pair that is associated with a particular key. For example, with a list we can extract the object at index 0, which we may have decided is the name, whereas with dictionaries, I can instead extract the a value using a key "name". Because of how they work, dictionaries are sometimes known as associative arrays in other programming languages.

Dictionaries are created using braces { } , where each key/value pair is separated by a comma (,) and keys are separated from their values using a colon (:). In the example below, I created a dictionary with two keys: name, id.

Python - Creating a Dictionary

```
# In this dictionary, I associated the name John Smith to the key
"name",
# and the id number 12345 to the key "id".
myDictionary = {"name":"John Smith", "id":12345}
```

Basic Python - Lists and Dictionaries

Watch the Video

Using a Dictionary

The keys in a dictionary can be numbers, strings, or tuples, but typically a string is used to make a key that best describes the value. Any given key may only appear once in a dictionary, so trying to set another value for a key that already exists will overwrite the previous value for that key. Alternately, attempting to access the value of a key that does not exist will throw an error, while setting a value to a key that does not exist will create a new key/value pair within the dictionary.

To access a value in a dictionary works much like accessing a value in a list; simply place brackets containing the key after the dictionary object.

Python - Accessing Values in a Dictionary

```
# Creates a dictionary with three key/value pairs.
myDictionary = {'Bob': 89.9, 'Joe': 188.72, 'Sally': 21.44}

print myDictionary['Joe'] # Will print out: 188.72

# Adds a key for 'Amir', and alters the value associated with the key 'Sally'.
myDictionary['Amir'] = 45.89
myDictionary['Sally'] = 146.23

print myDictionary # Will print out the whole dictionary: {'Joe': 188.72, 'Amir': 45.89, 'Bob': 89.9,
'Sally': 146.23}
```

It is also easy to loop through all of the values of a dictionary using the keys() function. For example:

Python - Keys Function

```
# The keys() function provides us with a list of keys, which we can iterate through and print out in
addition to using in the value lookup.
for key in myDict.keys():
   print key, myDict[key]
```

There are many use cases for dictionaries, but they are commonly used in Ignition when passing values into a Message Handler or creating a dynamic roster for alarms.

Dictionary Functions

Dictionaries have a few functions that allow for greater control over the dictionary object and the values contained within.

Function	Description	Example	Output

len(dictionary)	Returns the number of items in the dictionary.	`myDictionary = {"name":"John Smith", "id":12345}`
		`print len (myDictionary)`
			2

del dictionary[key]	Will remove the named key.	`myDictionary = {"name":"John Smith", "id":12345}`
		`del myDictionary["id"]`
		`print myDictionary`
			{'name': 'John Smith'}

key in dictionary	Will return True if the dictionary has that key.
	Can also use "key not in dictionary"
		`myDictionary = {"name":"John Smith", "id":12345}`
		`if "name" in myDictionary:`
		`        print myDictionary["name"]`
			John Smith

dictionary.clear()	Remove all of the items in the dictionary.	`myDictionary = {"name":"John Smith", "id":12345}`
		`myDictionary.clear()`
		`print myDictionary`
			{}

dictionary.keys()	Returns a list of the dictionary's keys.	`myDictionary = {"name":"John Smith", "id":12345}`
		`print myDictionary.keys()`
			['name', 'id']

dictionary.values()	Returns a list of the dictionary's values.	`myDictionary = {"name":"John Smith", "id":12345}`
		`print myDictionary.values()`
			['John Smith', 12345]

Related Topics ...

- Client Message Handler
- Gateway Message Handler
- Numeric Types
- Strings
- Lists and Tuples
- Datasets
- Dates

Datasets

Datasets and PyDatasets

A dataset can be thought of as a two dimensional list, or rather a list where each object is another list of objects. Datasets are not normally native to Python, but are built into Ignition because of their usefulness when dealing with data from a database. It is very common to deal with datasets in scripting, as datasets power many of the interesting features in Ignition, like charts and tables.

The main confusion when dealing with datasets is the difference between the dataset object and the PyDataset object. Dataset is the kind of object that Ignition uses internally to represent datasets. When you get the data property out of a component like a Table, you will get a dataset. The PyDataset is a wrapper type that you can use to make datasets more accessible in Python. The biggest differences are seen in how we access the data in the two different objects. However, you can easily convert between the two with system.dataset.toDataSet and system.dataset.toPyDataSet, making it simple to use the object that you find easier to use.

On this page ...

INDUCTIVE UNIVERSITY

Working with Datasets

Watch the Video

Creating Datasets

Because datasets are not native to Python, there is no way to naturally create them within scripting. Instead they must be created using the system.dataset.toDataSet function, which also allows you to convert a PyDataset to a Dataset. It requires a list of headers and a list of each row's data. Each data row list must be the same length as the length of the headers list.

Python - Creating a Dataset

```
# First create a list that contains the headers, in this case there
are 4 headers.
headers = ["City", "Population", "Timezone", "GMTOffset"]

# Then create an empty list, this will house our data.
data = []

# Then add each row to the list. Note that each row is also a list
object.
data.append(["New York", 8363710, "EST", -5])
data.append(["Los Angeles", 3833995, "PST", -8])
data.append(["Chicago", 2853114, "CST", -6])
data.append(["Houston", 2242193, "CST", -6])
data.append(["Phoenix", 1567924, "MST", -7])

# Finally, both the headers and data lists are used in the function
to create a Dataset object
cities = system.dataset.toDataSet(headers, data)
```

⚠ All code snippets on this page will reference the cities dataset we created above, so place that code at the beginning of every code snippet.

Accessing Data in a Dataset

To access the data inside of a dataset, each dataset has a few functions that can be called on to access different parts of the dataset. These are listed in the table below.

Function	Description	Example	Output
`data.getColumnAsList(colIndex)`	Returns the column at the specified index as a list.	`print cities.getColumnAsList(0)`	`[New York, Los Angeles, Chicago, Houston, Phoenix]`

`data.getColumnCount()`	Returns the number of columns in the dataset.	`print cities.getColumnCount()`	4
`data.getColumnIndex(colName)`	Returns the index of the column with the name colName.	`print cities.getColumnIndex("Timezone")`	2
`data.getColumnName(colIndex)`	Returns the name of the column at the index colIndex.	`print cities.getColumnName(1)`	`Population`
`data.getColumnNames()`	Returns a list with the names of all the columns.	`print cities.getColumnNames()`	`[City, Population, Timezone, GMTOffset]`
`data.getColumnType(colIndex)`	Returns the type of the column at the index colIndex.	`print cities.getColumnType(3)`	`<type 'java.lang.Integer'>`
`data.getColumnTypes()`	Returns a list with the types of all the columns.	`print cities.getColumnTypes()`	`[class java.lang.String, class java.lang.Integer, class java.lang.String, class java.lang.Integer]`
`data.getRowCount()`	Returns the number of rows in the dataset.	`print cities.getRowCount()`	5
`data.getValueAt(rowIndex, colIndex)`	Returns the value at the specified row and column indexes.	`print cities.getValueAt(1, 2)`	PST
`data.getValueAt(rowIndex, colName)`	Returns the value at the specified row index and column name.	`print cities.getValueAt(2, "Population")`	2853114

Looping Through a Dataset

Oftentimes you need to loop through the items in a dataset similar to how you would loop through a list of items. You can use the functions above to do this.

Python - Looping Through a Dataset

```
# We use the same cities dataset from above. Using the range function, we can come up with a range of
values that represents the number of columns.
```

```
for row in range(cities.getRowCount()):
        for col in range(cities.getColumnCount()):
                print cities.getValueAt(row, col) # Will print out every item in our cities dataset,
starting on the first row and moving left to right.
```

Accessing Data in a PyDataset

> PyDatasets can be accessed in the same ways that Datasets can. This means that all of the above functions (getColumnCount(), getValueAt(), etc) can be used with PyDatasets too.

PyDatasets are special in that they can be handled similarly to other Python sequences. Any dataset object can be converted to a PyDataset using the function system.dataset.toPyDataSet. All of the functions listed above can be used on a PyDataset, but the data can also be accessed much easier, similar to how you would a list.

Python - Accessing Data in a PyDataset

```
# First convert the cities dataset to a PyDataset.
pyData = system.dataset.toPyDataSet(cities)

# The data can then be accessed using two brackets at the end with row and column indexes. This will
print "PST"
print pyData[1][2]
```

Looping Through a PyDataset

Looping through a PyDataset is also a bit easier to do, working similar to other sequences. The first for loop will pull out each row, which acts like a list and can be used in a second for loop to extract the values.

Python - Looping Through a PyDataset

```
# Convert to a PyDataset
pyData = system.dataset.toPyDataSet(cities)

# The for loop pulls out the whole row, so typically the variable row is used.
for row in pyData:
        # Now that we have a single row, we can loop through the columns just like a list.
        for value in row:
                print value
```

Additionally, a single column of data can be extracted by looping through the PyDataset.

Python - Extract a Column of Data by Looping Through a PyDataset

```
# Convert to a PyDataset
pyData = system.dataset.toPyDataSet(cities)

# Use a for loop to extract out a single row at a time
for row in pyData:
        # Use either the column index or the column name to extract a single value from that row.
        city = row[0]
        population = row["Population"]
        print city, population
```

PyRow

A PyRow is a row in a PyDataset. It works similarly to a Python list.

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

In 8.0.14, we added the following methods to PyRows. The examples and outputs are based on the results in the table below. In addition, "print" commands are used, but should be replaced by appropriate logging methods (such as system.util.getLogger) depending on the scope of the script.

A	B
Apple	Orange
Banana	Orange
Apple	Apple

Method	Description	Syntax	Example	Output

index()	Returns the index of first occurrence of the element. Returns a ValueError if the element isn't present in the list.	index(element)	`for row in pyDataset:`
			`    try:`
			`        print row.index("Apple")`
			`        except:`
			`        print "No apples in this row"`
				`0`
				`No apples in this row`
				`0`

count()	Calculates total occurrence of given element in the row.	count(element)	`for row in pyDataset:`
			`        print row.count("Apple")`
				`1`
				`0`
				`2`

Repeating Elements

8.0.14 also introduces the concept of repeating elements in a row:

Example	Output

`for row in PyDataset`
`        print row * 2`
	`[u'Apple', u'Orange', u'Apple', u'Orange']`
	`[u'Banana', u'Orange', u'Banana', u'Orange']`
	`[u'Apple', u'Apple', u'Apple', u'Apple']`

Altering a Dataset

Technically, you cannot alter a dataset. Datasets are immutable, meaning they cannot change. You can, however, create new datasets. To change a dataset, you really create a new one and then replace the old one with the new one. There are system functions that are available that can alter or manipulate datasets in other ways. Any of the functions in the system.dataset section can be used on datasets, the most common ones have been listed below:

- `system.dataset.addRow`
- `system.dataset.deleteRow`
- `system.dataset.setValue`
- `system.dataset.updateRow`

The important thing to realize about all of these datasets is that, again, they do not actually alter the input dataset. They return a new dataset. You need to actually use that returned dataset to do anything useful.

For example, the following code is an example of the setValue function, and would change the population value for Los Angeles.

Python - Altering a Dataset Using the setValue Function

```
# Create a new dataset with the new value.
newData = system.dataset.setValue(cities, 1, "Population", 5000000)

# The cities dataset remains unchanged, and we can see this by looping through both datasets.for row in
range(cities.getRowCount()):
for row in range(cities.getRowCount()):
        for col in range(cities.getColumnCount()):
                print cities.getValueAt(row, col)

for row in range(newData.getRowCount()):
        for col in range(newData.getColumnCount()):
                print newData.getValueAt(row, col)
```

Related Topics ...

- Numeric Types
- Strings
- Lists and Tuples
- Dictionaries
- Dates

Dates

Overview

Dates can normally be tricky since they generally require very specific formats. Furthermore, some functions/objects require a date object instead of a string. Fortunately, there are several ways to create and alter date objects with scripting in Ignition.

Python has some built-in libraries to create and manipulate dates and times. However, most users find both Ignition's built-in system functions and even Java's Calendar class easier to use. Regardless, this section will demonstrate some examples from each approach.

On this page ...

Ignition's System Functions

Ignition's system.date library has a large number of functions that provide easy access to datetime creation and manipulation. This page has just a few simple examples. Additional examples and functions can be found in the scripting appendix.

Creating Dates

New datetimes can be created by using either the system.date.now or system.date.getDate. The system.date.getDate function returns a datetime, but the time is set to midnight. However, we can use system.date.setTime to change the time.

Python - System Functions - Creating Dates

```
# Get the current datetime.
print system.date.now()

# Create a date. The time will be set to midnight.
newDate = system.date.getDate(2018, 10, 28)
print newDate

# Change the time on the new date to 11:30 am.
print system.date.setTime(newDate, 11, 30, 0)
```

INDUCTIVE UNIVERSITY

Basic Python - Dates, Colors, and JSON Strings

Watch the Video

Formatting Dates

When printed, datetimes default to a format like the following: **Sun Jan 1 00:00:00 TZ 2018**. However, this can be manipulated by using special characters in the system.date.format function:

Python - System Functions - Date Formatting

```
rightNow = system.date.now()

# Demonstrating the standard format.
print rightNow

# Demonstrating the modified format.
print system.date.format(rightNow, "yyyy-MM-dd HH:mm:ss")
```

Date Arithmetic

The system.date.add* functions can be used to add to or subtract some amount of time from a date. See the system.date.add* functions for more information.

Python - System Functions - Date Arithmetic

```
# Get the current datetime.
newDate = system.date.now()

# Change the time on the new date to 30 minutes ago.
print system.date.addMinutes(newDate, -30)
```

Date Formatting Characters

The following is a reference of date formatting characters that can be used by system.date.format or Java's DateFormat class. Additionally, there are many other non-scripting uses in Ignition (such as the Vision - Calendar component's Format String property) that can utilize this reference.

Symbol	Description	Presentation	Example	Other Notes
`G`	Era designator	Text	`G=AD`	
`y`	Year	Year	`yyyy=1996`; `yy=96`	Lowercase y is the most commonly used year symbol
`Y`	Week year	Year	`YYYY=2009`; `YY=09`	Capital Y gives the year based on weeks (ie. changes to the new year up to a week early)
`M`	Month in year	Month	`MMMM=July`; `MMM=Jul`; `MM=07`	
`w`	Week in year	Number	`27`	If Dec31 is mid-week, it will be in week 1 of the next year
`W`	Week in month	Number	`2`	
`D`	Day in year	Number	`189`	
`d`	Day in month	Number	`10`	
`F`	Day of week in month	Number	`2`	2nd Sunday of the month
`E`	Day name in week	Text	`EEEE=Tuesday`; `E=Tue`	
`u`	Day number of week	Number	`1`	(1 = Monday, ..., 7 = Sunday)
`a`	Am/Pm marker	Text	`PM`	
`H`	Hour in day (0-23)	Number	`0`	
`h`	Hour in am /pm (1-12)	Number	`12`	
`k`	Hour in day (1-24)	Number	`24`	
`K`	Hour in am /pm (0-11)	Number	`0`	
`m`	Minute in hour	Number	`30`	
`s`	Second in minute	Number	`55`	
`S`	Millisecond	Number	`978`	
`z`	Time zone	General time zone	`zzzz=Pacific Standard Time`; `z=PST`	
`Z`	Time zone	RFC 822 time zone	`Z=-0800`	

X	Time zone	ISO 8601 time zone	X=-08; XX=-0800; XXX=-08:00	

Java's Calendar Class

Ignition's System Functions vs Java's Calendar Class

While Java's Calendar class if useful, in many cases Ignition's built-in system.date functions are simpler to use. Furthermore, the system.date functions typically use the Calendar class to retrieve the current time, so you are not losing any functionality by using the system functions.

It is highly advisable to use Ignition's system functions to generate and manipulate dates. The information on this page pertaining to the Calendar class is maintained in the interest for legacy installations.

Creating Dates

To create an arbitrary date, you can use the `java.util.Calendar` class. It has various functions to alter the calendar fields, like Calendar.HOUR, Calendar.MONTH, and so on. After you're done manipulating the Calendar, you can use its `getTime()` function to retrieve the Date represented by the calendar. It also has a handy `set()` function that takes the common parameters of a Date. The one major "gotcha" here is that January is month zero, not month one. For example:

Python - Calendar Class - Creating Dates

```
from java.util import Calendar
cal = Calendar.getInstance()

# set year, month, day, hour, minute, second in one call
# This sets it to Feb 25th, 1:05:00 PM, 2010
cal.set(2010, 1, 25, 13, 5, 0)
myDate = cal.getTime()
```

Date Arithmetic

Often you'll have a Date object from a component like the Popup Calendar and want to alter it programmatically. Say, subtracting 8 hours from it, or something like that. The `java.util.Calendar` class is used for this as well. Following the example above, this code would subtract 8 hours from the variable `myDate`.

Python - Calendar Class - Date Arithmetic

```
from java.util import Calendar
cal = Calendar.getInstance()
cal.setTime(myDate)
cal.add(Calendar.HOUR, -8)
myNewDate = cal.getTime()
```

Python's Time and Datetime Libraries

Ignition's System Functions vs Python's LIbraries

Many components in Ignition that contain a Date property actually expect a Java calendar object. Creating a datetime object using Python's built-in libraries and passing them to a date property on a component will result in an exception.

As a result, it is highly recommended to use Ignition's built-in system.date functions.

Creating Dates - Python's Time Library

The time library can be use to return dates as well as time. Times are created as a tuple of integers. The integers represent the following values: year, month, day of the month, hour, minute, second, weekday, day of the year, daylight savings time)

Check out Python's time library documentation for more information.

Python - Python Library - time

```
import time

# Finds the current local time. The time is returned as a tuple of integers.
myTime = time.localtime()

# Print the time into a 24-character string with the following format: Sun Nov 20 12:00:00 2017
print time.asctime(myTime)

# Alternatively, we can reformat the time in a custom manner, then print it
print time.strftime('%H:%M:%S %b %d %Y', myTime)
```

Creating Dates - Python's Datetime Library

Python's datetime library offers a bit more flexibility since arithmetic can easily be applied. Additional information on the datetime library can be found in Python's official documentation.

Note the double use of 'datetime' in the example below. This is because the 'datetime' library has a class named 'datetime.'

Python - Python Library - datetime

```
import datetime

# Returns the current datetime.
print datetime.datetime.now()
```

However, we can clean up the above by importing in the datetime class from the library:

Python - Python Library - datetime

```
# Imports the class named 'datetime' from the 'datetime' library, so we don't have to state it twice.
from datetime import datetime

# Returns the current datetime.
print datetime.now()
```

If you need to create a specific datetime, instead of just using the current, you can pass in the values directly when creating an instance of datetime:

Python - Python Library - Creating a New Time

```
from datetime import datetime

# Prints out the following datetime: 2018-01-02 03:04:05.000006
print datetime(2018,1,2,3,4,5,6)
```

Finding the difference between two datetime objects can easily be accomplished by using the '-' character

Python - Python Library - Date Difference

```
from datetime import datetime

rightNow = datetime.now()
someTime = datetime(2018,1,1,1,1,1,1)

# Find the difference between the two dates.
print someTime - rightNow
```

Date Arithmetic

With Python's built-in libraries, the timedelta class provides the simplest way to perform arithmetic on a date: It simply creates an object that effectively represents a duration. The duration can then be applied to a datetime.

Python - Python Library - Date Arithmetic

```
# We're including the timedelta class here
from datetime import datetime, timedelta

rightNow = datetime.now()

# Creating a timedelta object, and setting the hours to 8
offset = timedelta(hours = 8)

# Print the current time, and then print the time minus the offset.
print rightNow
print rightNow - offset
```

Related Topics ...

- Numeric Types
- Strings
- Lists and Tuples
- Dictionaries
- Datasets

Conditions and Loops

On this page ...

If-Statements

The **if** statement should be familiar to anyone with a passing knowledge of programming. The idea of an **if** is that you want your script to execute a block of statements only when a certain condition is true. Python's **if** is simple to use, and has some additional keywords to provide more flexibility.

Simple If-Statement Example

The syntax for **if** is as follows:

Pseudocode - If Statement

```
# Note that 'if' uses lowercase characters.
# Additionally, a colon is placed after the expression.
if expression:

        # The statements that should execute when the expression is
true
        # MUST be indented.
        statement
```

Example

```
x = 5
z = 15
if x < 10:
        # Since the condition "x < 10" is
true,
        # the following line will execute
        print "'x' is less than 10"
if z < 10:
        # This condition "z < 10" is false,
        # so the following line will not
execute
        print "this will never show"
```

Output

```
'x' is less than
10
```

Control Flow Logic

Watch the Video

If and Else

You can use the **if...else** form of an **if** statement to do one thing if a condition is true, and something else if the condition is false.

Example

```
x = 15
if x < 10:
   print "x is less than 10"
else:
   print "x is not less than 10"
```

Output

```
x is not less than
10
```

Elif (Else If)

Lastly, you can use the **if...elif** form. This form combines multiple condition checks. `elif` stands for "`else if`". This form can optionally have a catch-all `else` clause at the end. For example, this script will print out **`three`**:

Example

```
x = 3
if x == 1:
```

Output

three

```
   print "one"
elif x == 2:
   print "two"
elif x == 3:
   print "three"
else:
   print "not 1-3"
```

You can use as many elif items as you want, and the else is not required at the end.

For-Loops and While-Loops

Control Flow Loops

Watch the Video

For-Loop

Python's **for** loop may be a bit different than what you're used to if you've programmed any C. The **for** loop is specialized to iterate over the elements of any sequence, like a list. A **for** loop uses an iterator variable to reference each item as it steps through the sequence. This means it's very simple to write a loop!

Note that the syntax of the for-loop requires use of the in-keyword.

Pseudocode - For Loop

```
# In this example, "item" is a variable created specifically by the
"for" loop to act as an iterator.
# The name "item" is not a keyword, and a different variable name may
be used.
# Additionally, note that "for" and "in" are lowercase, and a colon
is present at the end of the line.
for item in sequence:
        # All statements that should execute each iteration must be
indented after the "for" statement
        statement
```

Example

```
listOfFruit = ['Apples', 'Oranges', 'Bananas']
for fruit in listOfFruit:
   print fruit
```

Output

```
Apples
Oranges
Bananas
```

You don't need to manually create a sequence to repeat a task several times in a for loop. Instead, the built-in function **range()** function can generate a variable-size list of integers starting at zero. For example, calling **range(4)** will return the list [0, 1, 2, 3].

Example

```
# Even though this example isn't using the value of "x",
# the print statement will still be executed once for each item
# in the list returned by range().
for x in range(4):
   print "this will print 4 times"
```

Output

```
this will print 4 times
this will print 4 times
this will print 4 times
this will print 4 times
```

While-Loop

A while loop will repeat a block of statements as long as a condition is true. This code will print out the contents of the items in the list.

Pseudocode - While Loop

```
# A while loop simply needs the keyword "while", the condition that
# determines when we should stop iterating, and a colon at the end of the line.
while condition:
        # All statements that should be repeated each iteration must be indented after the "while"
statement
        statement.
```

This code uses a function called **len()** , which is a built-in function that returns the length of a sequence.

Example

```
listOfFruit = ['Apples', 'Oranges', 'Bananas']
x = 0
while x < len(listOfFruit):
   print listOfFruit[x]
   x = x + 1
```

Output

```
Apples
Oranges
Bananas
```

The Break and Continue Statements

You can stop a loop from repeating in its tracks by using the **break** statement. This code will print out " Loop " exactly two times, and then print " Finished ".

Example

```
for x in range(10):
   if x >= 2:
      break
   print "Loop"
print "Finished"
```

Output

```
Loop
Loop
Finished
```

You can use the **continue** statement to make a loop stop executing its current iteration and skip to the beginning of the next iteration. The following code will print out the numbers 0-9, skipping 4

Example

```
for x in range(10):
   if x == 4:
      continue
   print x
```

Output

```
0
1
2
3
5
6
7
8
9
```

Infinite Loops

It is incredibly easy to create an infinite loop when using a **while** statement. Depending where the infinite loop was created, it could cause you to lose your work in the Designer, or create a large amount of overhead on the Gateway.

Python - Infinite Loop Created by While Statement

```
x = 0
while x < 10:
        x += 1 # Forgetting to add a way to increment "x" will cause an infinite loop
        print x
```

In many cases, a **for** loop could be used instead of a **while**, but this is not always possible. When using **while**, the best way to avoid an infinite loop is to make sure you always have a way to exit the loop: a simple approach involves using a counter that can eventually trigger a **break** statement, or add the counter as an additional condition to the **while**.

Python - Preventing Infinite Loops Using the Break Keyword

```
###Example 1: using the break keyword
# The counter variable will be used as a guaranteed way out of the While.
counter = 0

# Normally, using True as a condition in a While would be a quick
# way to generate an infinite loop, but the counter helps prevent that
while (True):

        # Increase the counter
        counter += 1

        # Check the value of the counter. If it's at the point where we can assume we're going to be
looping indefinitely...
        if counter >= 1000:

                # Break out of the loop
                break
```

Python - Preventing Infinite Loops Using an Additional Condition

```
###Example 2: using an additional condition
# Again, the counter variable will be used as a guaranteed way out of the While.
counter = 0

# Instead of using nested logic, we can simply add counter's value as an additional condition with "and"
while (True and counter < 1000):

        # Increase the counter. Once counter >= 1000, the while loop will be forced to end.
        counter += 1
```

The Pass Keyword

When using conditional statements and loops, the pass keyword can be especially useful when writing a new script. When called, the pass keyword does nothing, which may seem useless. However it is great when you find yourself in a situation where you need a line of code to meet a syntax requirement, but don't want the code to do any additional work.

Python - Pass Keyword

```
myVar = system.tag.read(tagPath).value

if myVar == 0:
        firstFunction()
elif myVar == 1:
        secondFunction()
elif myVar == 2:
        # I haven't implemented the thirdFunction() yet, so I can use pass here as a placeholder
        pass
```

Related Topics ...

- Error Handling
- Getting Started with Scripting in Ignition

Error Handling

What is Error Handling

The concept of error handling is recognizing when an error might occur in a block of code, and instead of throwing the error, handling it gracefully. This can involve giving the user a more distinct error message, letting the user know that their attempt to run the code failed, or even undoing something that your code set it motion so that it can be started again from the same starting point.

On this page ...

Error Handling in Python

Within Python, we can use the **try** and **except** blocks to handle errors. We would first use `try:` and write the code we would like to try indented underneath it. We then must have an `except:` with code that will run if there is an error.

Pseudocode - Error Handling

```
# With try, we can attempt any amount of code
try:
        some code
        more code
        even more code

# If any of lines in the try block were to throw an error, then we move down and run the block under except.
# The except statement is NOT optional: you must define what your code should do in the event an exception occurs.
except:
        failover code
```

When running the code above, there is a specific order to the way it executes.

1. The try block is run, and the code within is executed line by line.
   a. If an error occurs, the code in the try block will immediately stop and the except block will begin to execute.
   b. If no error occurs after all code in the try block has been executed, the try block will be finished and the code in the except block will be skipped.
2. After either outcome, any code after the except will then execute.

Because of the way the try and except blocks work, it is very useful on situations that require user input, where something may have been incorrectly entered which would usually result in an error.

The Pass Keyword

The **pass** keyword is unique in that it does nothing except to fill in a spot where code is required. This is useful if we want to handle the exception so that it doesn't throw an error message, but we don't actually want to do anything with it. In this case, we can use pass to tell the script to continue.

Pseudocode - Pass Keyword

```
try:
        some code
        more code
        even more code

except:
        # The error will bring the code to the exception, and then the exception will simply do nothing.
        pass
```

Error Handling Example

An easy way to demonstrate how error handling works is with a division example, since it is easy to cause an error by dividing by 0. Take the code below:

When we run it, we get a printed value of 50. There was no error in the division, and the try block finished successfully. However, if we were to change the value of x to 0, we can see that "An error occurred!" is printed.

Python - Error Handling Division

```
# We start with a value, which could represent some user input.
x = 2

# We use that value in our try block, and have a simple print statement if there is an error.
try:
        value = 100/x
        print value
except:
        print "An error occurred!"
```

Exception-Specific Failover

While each try block must be followed by at least one except block, there can be multiple except blocks to handle different types of errors. This is done by listing the error object after the except keyword and before the colon. Looking back at the example above, I know that my most common error is going to be a divide by zero error. So I can make an exception that is specific to divide by zero errors.

Python - Exception-Specific Failover

```
# We start with a value, which could represent some user input.
x = 0

# Use the user input in division in our try block.
try:
        value = 100/x
        print value

# If the exception that would occur is of type ZeroDivisionError, we can run a specific block of code
except ZeroDivisionError:
        print "Dividing by zero is not allowed. Please stop trying to divide by zero"
# We can then have a second exception without a specific error. This except acts as a catch-all;
# if the user caused an exception we didn't account for above, we can failover to the block of code
below.
except:
        print "An error occurred!"
```

The except blocks in the code above cover all possible errors as represented in the table below. Now, we have a more tailored approach to how we handle errors while still catching all of them that may occur.

Inputs (x value)	Output
2	50
0	Dividing by zero is not allowed. Please stop trying to divide by zero
'a'	An error occurred!

Each try block can have many except blocks, and each except block can also name multiple types of errors as shown below. However, an error that happens within a try block will only ever trigger a single except block.

Pseudocode - Try Block with Except Blocks

```
try:
        some code
except (ZeroDivisionError, RuntimeError, TypeError):
        failover code
```

Displaying Error Text

Sometimes you want to get the actual text from an error in addition to protecting your script. There's an easy way to fetch that information that's built into Python. When you are inside an except section of code, sys.exc_info() gives you access to the error text as a list of values. You can use this to print out your message, display it on the screen, send it to the database, or anything else.

This example should be put on a button, and will write the error text to a Label component that is a sibling to the button. This is useful in Perspective to get error messages out of views and event actions.

```
try:
        # cause an error
        x=[1,2]
        val = x[5]
except:
        # push the error text to a sibling label
        self.getSibling("Label").props.text = sys.exc_info()
```

You can also use the Ignition loggers to push these error messages out to the Gateway console. You can find these in the Gateway Webpage under the Status section, on the **Logs** page.

```
# This code would log an error to the gateway
try:
        # cause an error
        100/0
except:
        # push the error text to the logger
        logger = system.util.getLogger("myLogger")
        # convert the sys.exc_info() to a string and log it
        logger.info(str(sys.exc_info()))
```

Determining the Error Object

To determine the name of the error object that will be thrown from certain errors, we can take a look at the error to figure that out. We already mentioned that dividing by zero gives a ZeroDivisionError, but what about when we divide by a string? If I divide 100 by the letter a without error handling, this is the error I get:

```
Traceback (most recent call last):
File "<buffer>", line 3, in <module>
TypeError: unsupported operand type(s) for /: 'int' and 'str'
```

The last line of that error gives the name of the error object "TypeError" followed by the cause of that error. This makes sense, because the string 'a' is the wrong type to be using in division. However, not all errors are so simple and may require a bit more to find the name of the error. For a list of python error names check out this page in the python docs: https://docs.python.org/2.7/library/exceptions.html#Exception

Additionally, some exceptions may be returned by Java. In these cases, Oracle's documentation on the Exception class is more useful: https://docs.oracle.com/javase/8/docs/api/java/lang/Exception.html
A list of common exceptions are listed below.

Exception	Description	Exception Demonstration

ArrayIndexOutOfBoundsException	This typically occurs when a line of code attempts to access an index in a collection,
	but the index specified doesn't exist. This exception is unique to datasets in Ignition. Lists,
	and other built-in Python objects will return the IndexError below.
		`myDataset = system.dataset.toDataSet(["colName"],[[0]])`
		
		`# This will fail because the dataset only has a single row,`
		`# so trying to read the value at`

		`row index 5 means we're trying to`
		`# read something that doesn't exist.`
		`print myDataset.getValueAt(0,5)`

AttributeError	Typically seen when a script tries to access a property that doesn't exist.
	Example: a script tries to get the value of a property on a component, but the property name is misspelled.
		`myVar = 10`
		`# integers do not natively have a name variable, so this will fail with an AttributeError:`
		`# there isn't an 'attribute' on an integer by the name of 'name'`
		`print myVar.name`

IndexError	Similar to ArrayIndexOutOfBoundsException above, but occurs when Python specific object, such as a list.	`myList = [1,2,3]`
		`# There isn't an element in the list at index 4, so this will fail.`
		`print myList[4]`

NameError	Occurs when the local or global name is not found. Typically happens when referencing a variable that hasn't been assigned a value.	`# We haven't given a value to the variable myValue, so it will fail.`
		`print "The value of the variable is: " , myValue`

TypeError	A TypeError occurs when a function or similar operation is applied to another object of an incorrect type.	`myList = [1,2,3]`
		`# The first argument for pop() expects an integer, so passing a string value is innappropriate.`
		`# Passing a string to pop() will return a TypeError.`
		`print myList.pop("0")`

ValueError	A ValueError is returned when the value of an argument is inappropriate. Typically the exact issue is returned in the returned exception.	`myVar = "Hello"`
		`# Strings can be passed to the int() function, but the value needs to be something that`
		`# can be coerced into an integer, like "0".`
		`# Because "Hello" can't be easily converted, the line below will fail.`
		`print int(myVar)`

Related Topics ...

- Conditions and Loops

Built-In Functions

Python Built-In Functions

Functions are code that can be called repeatedly from other places. Functions can have parameters passed into them and may return a resulting value. Some functions, like len(), are built-in. Some functions, like system.gui.messageBox(), are part of the scripting libraries provided by Ignition. Some functions, like math.sqrt(), are provided by the Python Standard Library.

Functions are invoked by using their name followed by an argument list surrounded in parentheses. If there are no arguments, you still need an open and close parenthesis.

This section details several useful Built-in Functions, along with some simple examples. See the official docs for more information.

On this page

...

Type Casting Functions

Python has many functions to convert between data types. Some common type casting functions are listed below

Function	Notes	Example	Output

`bool()`	When casting a numeric value to a boolean, a zero value is false, while all non-zero numbers are True
	
	When casting a String or Unicode value to a boolean, an empty string is False, any other string is True.
		`# Results in False`
		`print bool("")`
		
		`# Results in True`
		`print bool("Test")`
			`False`
			`True`

`int()` and `long()`	When casting a float, rounding will not occur automatically to the decimal value. Instead, the round() function should be called.
	
	When casting a String or Unicode value, the string literal needs to be a valid integer or long: decimal values contained in the string will result in a ValueError
	
	Integers have at least 32 bits of precision, while Longs have unlimited precision.
		`# Float to Integer`
		`print int(123.8)`
		
		`# Float to Long`
		`print long(321.8)`
		
		`# String to Integer`
		`print int("400")`
		
		`# ValueError: the value is not base 10`
		`print int("400.5")`
			`123`
			`321`
			`400`
			`ValueError`

`float()`	When casting a string literal as a float, non-numeric characters in the string will result in an exception, except for a decimal point (".").	`# Integer to Float`
		`print float(123)`
		
		`# String to Float`
		`print float("400.5")`
			`123.0`
			`400.5`

	Most objects can be cast as a string representation of some sort, including sequences.	`print "First line:" + str`	

`str()` and `unicode()`		`(80)`
		
		`# Even sequences can be cast as a string,`
		`# making for easy concatenation`
		`myList = [1,2,3]`
		`print str(myList)`
			`80`
			`[1, 2, 3]`

Checking an Object's Type

Checking the data type of an object can easily be done with both the `type()` and `isinstance()` functions.

Function	Description	Example	Output

`type(object)`	When passed a single parameter, this function returns the type of the `object`.	`var = 10`
		`print type(var)`
		`print type(str(var))`
			`type 'int'>`
			`<type 'str'>`

`isinstance(object, classinfo)`	Returns True if the `object` is an instance or subclass of `classinfo`, otherwise, returns false.
	
	If checking for a string or unicode type, a `classinfo` of "basestring", which is the base class for both strings and unicode types, would return True.
		`var = 10`
		`print isinstance(var,int)`
		
		`strVar = "string"`
		`print isinstance(strVar, basestring)`
			`True`
			`True`

Python - Type Validation: type vs isinstance

```
## type() Example
# This example attempts to validate the type of a variable. As written, this will evaluate as True, and thus the print statement would execute.
var = "My String"
if type(var) == type(""):
        print "Variable 'var' is a string"

## isinstance() Example
# The isinstance() function can offer the same functionality as above.
var = "My String"
if isinstance(var, str): # Note the lack of quotation marks around the classinfo parameter. We want to reference the class str, not the string "str".
        print "Variable 'var' is a string"
```

Generating a Range of Values

In some cases, it is useful to generate a range of integers for iteration. Python's `range()` function will return a list of integers.

Function	Description	Example	Output

`range([start,] stop[, step])`	Returns a list of progressively greater integers.
	`start` - Integer value denoting the initial value in the list. If omitted, defaults to 0. This parameter is inclusive.
	`stop` - Integer value, determines when to cease generating integers. This parameter is exclusive.
	`step` - Integer value to increment each new integer by. If omitted, step defaults to 1.
	If `step` is positive, integers will be generated as long as `( start + i*step < stop)` is true.
	If `step` is negative, integers will be generated as long as `( start + i*step > stop)` is true.
		`print range(5)`
		`print range(1, 5)`
		`print range(1, 10, 3)`
		`print range (15, 0, -3)`
			`[0, 1, 2, 3, 4]`
			`[1, 2, 3, 4]`
			`[1, 4, 7]`
			`[15, 12, 9, 6, 3]`

Assume we need to read from five separate Tags with a nearly identical Tag path in a single script:

Pseudocode - Tag Path

```
[Provider]Folder/Sub_Folder_1/Tag
[Provider]Folder/Sub_Folder_2/Tag
[Provider]Folder/Sub_Folder_3/Tag
[Provider]Folder/Sub_Folder_4/Tag
[Provider]Folder/Sub_Folder_5/Tag
```

Instead of manually typing each path, we could use `range()` in a for loop that would write the paths automatically.

Python - Range in a For Loop

```
# Initialize an empty list that will ultimately hold all the Tag paths.
tagPaths = []

# Use range to repeatedly append 5 tag paths to the tagPaths list: starting with a value of 1, and ending with a value of 5.
for num in range(1,6):
        # Use String Formatting to create a Tag path with the iterator's (num) value.
        tagPaths.append("[Provider]Folder/Sub_Folder_%i/Tag" % num)

# Now that tagPaths contains all our tag paths, we can use the list to interact with the tag, such as by reading their values simultaneously.
tagValues = system.tag.readAll(tagPaths).value
```

Rounding Numbers

You can round numbers inside Python with a few simple functions.

Function	Description	Example	Output

`round (number[, digits])`	When passed a single parameter, this function returns a rounded integer value of a number. If the decimal is greater than or equal to .5, the it rounds up, less than .5 rounds down.
	If the optional digits argument is used, then it rounds to that many decimal places.
		`var = 10.236981`
		`print round(var)`
		`prin`
			`10`
			`10.237`

		t round(var,3)	

`math.floor(number)`	Returns a truncated integer from the number given. The largest integer value less than or equal to *number*.
	
	Note that the example needs to import that math library before being able to call floor().
		`import math`
		
		`var = 100.938`
		`print math.floor(var)`
			`100.0`

math.ceil (number)	Returns the ceiling integer from the number given. The smallest integer value greater than or equal to *number*.
	
	Note that the example needs to import that math library before being able to call ceil().
		`import math`
		
		`var = 100.138`
		`print math.ceil(var)`
			`101.0`

Python - Simple Casting

```
stringVar = "40"

# Without using int(), this line would cause an exception. However int() is able to cast the type of
stringVar's value to an integer.
print 20 + int(stringVar)

# Type casting is also useful in cases where a string parameter is required, but a numerical value
should be given,
# such as the message parameter in system.gui.messageBox().
intVar = 60
# Note that this could also be accomplished with String Formatting instead of using str().
system.gui.messageBox(str(intVar))
```

Related Topics ...

- User Defined Functions
- Libraries

Libraries

The System Library

Ignition comes with a group of system functions, called the System Library. Using a system function is simple. For example, the following code will access the value of a Tag.

Pseudocode - Reading a Tag Value

```
value = system.tag.read("tagPath").value
```

The scripting appendix is full of built-in functions such as this.

On this page

...

System Library

Watch the Video

Python Libraries

Python Libraries are packages of extra functions that expand the functionality of the code and can be imported into a script. We do this by using the `import` keyword:

Pseudocode - Import a Library

```
# This pseudo code will import a library, and then call a function of that library.
import myLibrary

myLibrary.specialFunction()
```

The import keyword imports that entire library and allows you to use all of the functions inside of it by calling them off of the imported library. You can also import a piece of a library:

Pseudocode - Import a Function of a Library

```
# This pseudo code will import a function from a library, and then call that function.
from myLibrary import specialFunction

specialFunction()
```

Note, that since we are directly importing in the function, we can directly call it instead of having to call it off of the library.

Python Standard Library

Python has an extensive standard library that provides a host of new functionality to the scripting language. The python documentation goes over all of the libraries in its standard library as well as how to use them here: https://docs.python.org/2/library/index.html

Let's take a look at an example of using a common library:

Python - Accessing Files in a Python Standard Library

```
# The csv library provides an easy way to read csv files, regardless of how they are formatted.
import csv

# We first grab our filepath, and feed it into the open function, which opens the file.
filepath = "C:\\test.csv"
csvFile = open(filepath, 'r')

# We then pass our opened csv file object into the csv.reader function, which will read the file.
# This can be looped through in a for loop to print every row of the csv.
reader = csv.reader(csvFile)
for row in reader:
        print row
```

Importing 3rd Party Libraries

In addition to the standard libraries, 3rd party libraries can also be imported into Ignition's scripting environment. A Python Library or individual module file will consist of a python file (.py) that contains the code that implements the functions of the library. You can often find python libraries built by other users on the web, or can even create your own. These files can then be placed into a folder within your Ignition server.

- **Windows folder**: C:\Program Files\Inductive Automation\Ignition\user-lib\pylib
- **Linux folder**: /var/lib/ignition/user-lib/pylib
- **Mac OS X folder**: /usr/local/ignition/user-lib/pylib

Once the python file is in that folder, you can then import the library into a script just like any of the standard libraries.

> Ignition uses Python version 2.7. This means that any imported libraries must be compatible with Python 2.7.

Accessing Java

Scripting in Ignition executes in the java based implementation of Python called Jython. (See Python or Jython?). While this doesn't have any great effect on the Python language itself, one of the great side benefits is that your Python code can seamlessly interact with Java code as if it were Python code. This means that your Python code has access to the entire Java standard library.

To use Java classes, you simply import them as if they were Python modules. For example, the following code will print out all of the files in the user's home directory. This code uses the Java classes `java.lang.System` and java.io.`File` to look up the user's home directory and to list the files. Notice that we can even use the Python-style for loop to iterate over a Java sequence.

Python - Accessing Java

```
# Importing the appropriate java libraries.
from java.lang import System
from java.io import File

# Used to look up the files in the users home directory.
homePath = System.getProperty("user.home")
homeDir = File(homePath)

# Loops through the list of files and prints them.
for filename in homeDir.list():
   print filename
```

You can find the reference documentation for the Java standard class library (also known as, the "*JavaDocs*") at: http://docs.oracle.com/javase/8/docs/api/

Subclassing Java

You can also create Python classes that implement Java interfaces. You do need some understanding of Java and object-oriented programming concepts, which are outside the scope of this manual. To create a Python class that implements a Java interface, you simply use the interface as a superclass for your Python class. For example, we could augment the example above to use the overload java.io.File.list(FilenameFilter). To do this, we'll need to create a `FilenameFilter`, which is an interface in Java that defines a single function:

```
boolean accept(File dir, String name)
```

To implement this interface, we create a Python class that has `java.io.FilenameFilter` as its superclass, and implements that Java-style function in a Python-esque way.

Python - Implementing Java Interfaces

```
# Importing the appropriate java libraries.
from java.lang import System
from java.io import *

# This sets up an extension filter that can check the file extension. Txt is the default.
class ExtensionFilter(FilenameFilter):
   def __init__(self, extension=".txt"):
      self.extension=extension.lower()
   def accept(self, directory, name):
      # make sure that the filename ends in the right extension
      return name.lower().endswith(self.extension)

# Used to look up the files in the users home directory.
homePath = System.getProperty("user.home")
homeDir = File(homePath)

# Prints out all .txt files. Txt is provided if nothing is specified.
for filename in homeDir.list(ExtensionFilter()):
   print filename

# Prints out all .pdf files.
for filename in homeDir.list(ExtensionFilter(".pdf")):
   print filename
```

Related Topics ...

- Built-in Functions

User Defined Functions

Functions

A function is code that can be called repeatedly from other places. Functions can have parameters passed into them, and may return a resulting value. Some functions, like `len`, are built-in. Some functions, like `system.gui.messageBox()`, are part of the scripting libraries provided by Ignition. Some functions, like `math.sqrt()`, are provided by the Python standard library. However, functions can also be defined in a script that can be used later on in the script. In these user defined functions, you give the function a name and some code that will run when the function is called. Then later on in the script, you can call the function by its name and it will run the code specified in the function. This is useful, because it allows you to run a segment of code many times without having to repeat it within the script.

Functions are invoked by using their name followed by an argument list surrounded in parentheses. If there are no arguments, you still need an open and close parenthesis.

On this page

...

Defining Functions

Functions are defined using the `def` keyword. A function needs a name and a list of the arguments that it can be passed. For example, this code defines a function that prints "Hello World!".

Python - Defining a Function

```
# First we define our function.
def printHW():
   print "Hello World!"

# We can then call our function.
printHW()
```

INDUCTIVE UNIVERSITY

Functions

Watch the Video

Functions Arguments

When a function accepts arguments, the names of those arguments become variables in the function's namespace. Whatever value was passed to the function when it was invoked becomes the value of those variables. Arguments can have default values, which makes them optional. If an argument is omitted, then its default value will be used. The following code defines a function called **cap**, which will take a number, and make sure it is within an upper and lower limit. The limits default to 0 and 100.

Python - Defining Arugments

```
# We first define our function. Notice that we have 3 different arguments.
# x, min, and max. The min and the max are set to equal 0 and 100 respectively.
def cap(x, min=0, max=100):

        # Check if x is less than the min, return the min if true.
        if x < min:
                return min

        # Check if x is greater than the max, return the max if true.
        elif x > max:
                return max

        # Return the value if it is within the bounds.
        else:
                return x

# We can then see the outcome by running our function with a few different parameters.
# This will print out a 40, since it is within the bounds.
print cap(40)

# This will print out "0", since it is less than the min of 0.
```

```
print cap(-1)

# This will print out "100", since it is greater than the max of 100.
print cap(150)

# This will print out "150", because it uses a max of 200 instead of the default 100.
print cap(150, 0, 200)
```

Keyword Arguments

In Ignition, some complicated script functions are designed to take keyword arguments instead of normal parameters. In the description for those functions, you may see the following info box in this User Manual:

> This function accepts keyword arguments.

Arguments can also be specified by keyword instead of by position. In the example above, the only way someone would know that the 200 in the last call to `cap` specified the max is by its position. This can lead to hard-to-read function invocations for functions with lots of optional arguments. You can use keyword-style invocation to improve readability. The following code is equivalent to the last line above, using 200 for the `max` and the default for the `min`.

Python - Using Keyword Arguments

```
print cap(150, max=200)
```

Because we used a keyword to specify that 200 was the `max`, we were able to omit the `min` argument altogether, using its default. However, using a keyword argument before a non-keyword or positional argument is not allowed.

Python - Non-keyword Argument

```
# This would fail, because the function isn't sure what 150 is being used for.
print cap(max=200, 150)
```

Functions Are Objects

Perhaps one of the most foreign concepts for new Python users is that in Python, functions are first-class objects. This means that functions can be passed around to other functions (this concept is similar to the idea of function pointers in C or C++).

Suppose we wanted a general way to filter a list. Maybe sometimes we want the odd entries, while other times we want even ones. We can define a function called `extract` that takes a list and another function, and returns only entries that "pass" through the other function.

Python - Functions Passed to Other Functions

```
# We define a function that checks if the value passed in is odd.
def isOdd(num):
        return num % 2 == 1

# We define a function that checks if the value passed in is even.
def isEven(num):
        return num % 2 == 0

# We define a function that inserts our list into the appropriate function and returns valid values.
def extract(filterFunction, list):
        newList = []
        for entry in list:
                if filterFunction(entry):
                        newList.append(entry)
        return newList

# Prints out [0, 2, 4, 6, 8]
# Notice that isEven as not invoked, but passed to the filter function.
print extract(isEven, range(10))
```

Where Can Functions Be Defined

User Defined Functions can be defined anywhere that a script is used. As stated before, they are useful to run segments of code multiple times without having to repeat it. They are also used extensively in project scripts where multiple functions can be defined in a single script module. Finally, some special Ignition System functions like system.gui.createPopupMenu or the runScript Expression function use functions as arguments.

Function Scope

The concept of scope is very important in all programming, and Python is no exception. Scope defines what names are directly accessible without any qualifiers. Another way to put this is that the scope determines what variables are defined. In Python, a variable is defined at the time that it is assigned. What scope it belongs to is also defined by where the assignment occurs.

Pseudocode - Defining a Function for Scope

```
# On this line, there is no variable 'x' in scope.
doSomeWork()

# Now 'x' is defined in our scope, because we've assigned a value to it
x = 5

# This will work because x is in scope.
print x
```

When you define a function, that function gets its own scope. Variables that are assigned within that function body will not be available outside of the function.

Python - Variables Defined within a Function Not Available Outside Scope

```
# x is local to myFunction() because this is where it is defined.
def myFunction():
   x = 15
   print x

# This will fail, because x is not available in the outer scope
y = x + 10
```

Related Topics ...

- Built-in Functions
- Libraries

Scripting in Ignition

On this page

...

Where Is Scripting Used?

Python is used in many places in Ignition. Each location has its own events that trigger your scripts to run, and add functionality to your projects in different ways. The most apparent place is in event handlers on components and other objects in Vision Clients and Perspective Sessions.

Script Scope

One important thing to keep in mind before scripting in Ignition, is to understand the concept of scope. Within Ignition, there are different scopes: the Gateway Scope, the Perspective Session scope, and the Client Scope. Where a script is run from determines what scope it is running in. This is important because it determines what system functions can be run, what resources the script can interact with, and where the output will be written to. For example, running a script on a Tag is run in the Gateway Scope and the output is sent to the Gateway console (i.e., wrapper.log file) because Tags are stored in the Gateway. This means that the script will not be able to access any client level resources such as windows or components that you may have open in the Client. Additionally, some of the system functions like system.gui.errorBox only work in the "Client Scope," so you will not be able to use them in the script on the Tag.

"Client Scope" scripts, however, execute on the running client (and also in Designer when testing, but only in Preview Mode). For example, if a component on a window is running a script, its values are isolated to the client, and the output will be displayed on the Designer/Client output console.

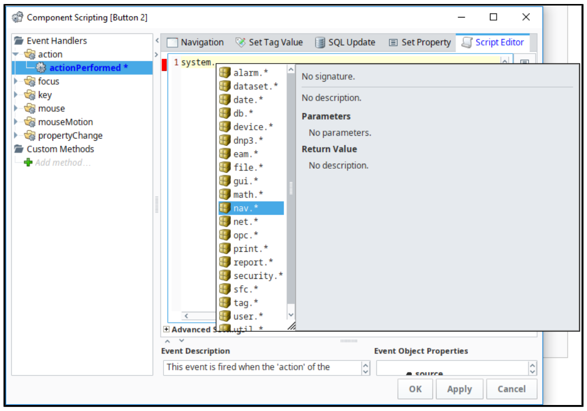
System Functions

Ignition comes with a group of system functions, called the System Library. Using a system function is simple. For example, the following code will access the value of a Tag.

Python - Simple Script Using a System Function

```python
value = system.tag.read("tagPath").value
```

A complete list of these functions (with their definitions) is available wherever you can add a script. Just type **system.** and then press **Ctrl+Space** to get a list of all the functions available. If you keep typing, the list will even be automatically narrowed down for you! Additionally, the Scripting Functions page in the appendix contains complete documentation for the built-in system functions.



```
system.
```

```
alarm.*
dataset.*
date.*
db.*
device.*
dnp3.*
eam.*
file.*
gui.*
math.*
nav.*
net.*
opc.*
print.*
report.*
security.*
sfc.*
tag.*
user.*
util.*
```

No signature.

No description.

Parameters

No parameters.

Return Value

No description.

This event is fired when the 'action' of the

Scripting in Ignition

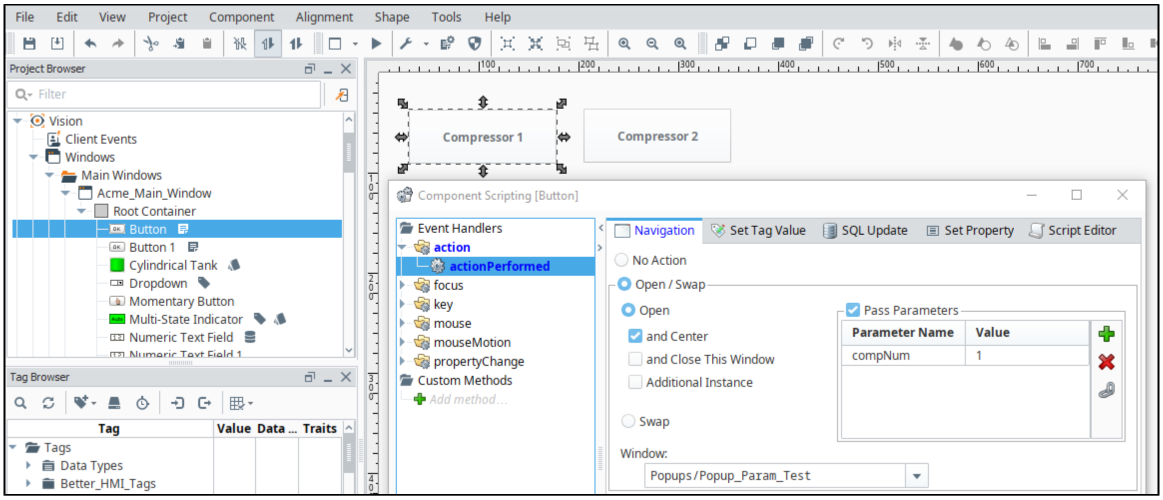
Watch the Video

System Library

Watch the Video

Components

Both Perspective and Vision offer component based scripting triggers, providing a means to execute a script under a number of different situations, such as a user interacting with a component or a component property value changing. For more information on how both module handle component based scripts, take a look at the Scripting in Perspective and Scripting in Vision sections.



○ No Action

● Open / Swap

● Open

☑ and Center

□ and Close This Window

□ Additional Instance

○ Swap

☑ Pass Parameters

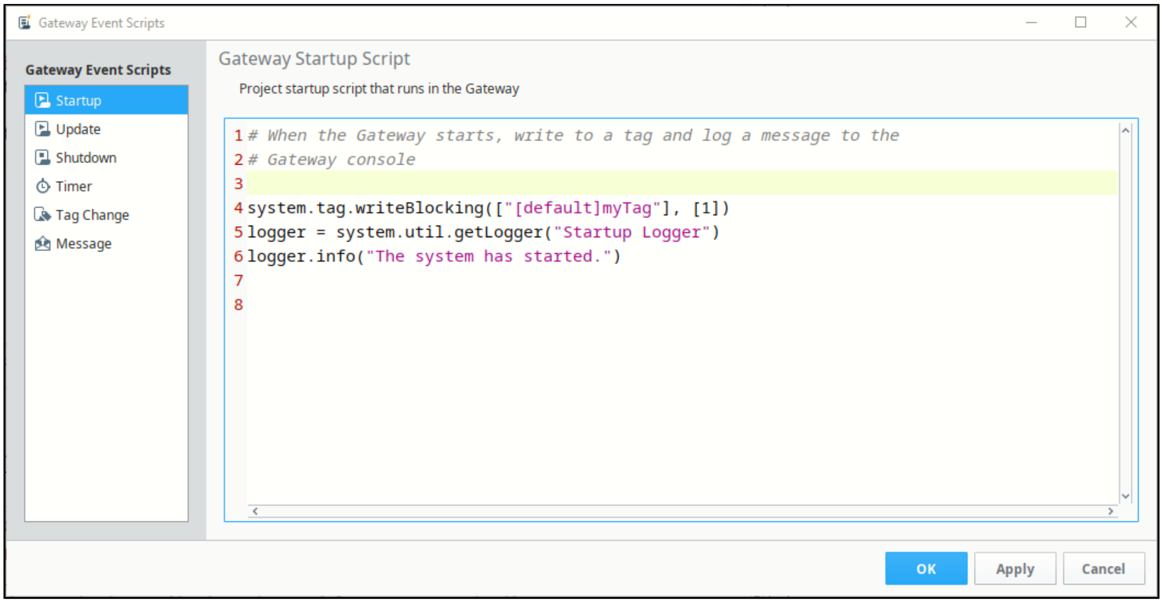
Parameter Name	Value
compNum	1

Window: Popups/Popup_Param_Test

Client, Gateway, and Session Event Scripts

Scripts can be set to activate on specific events that occur during runtime. For example, you can triggering a script to run when a vision client starts, or on certain time intervals.

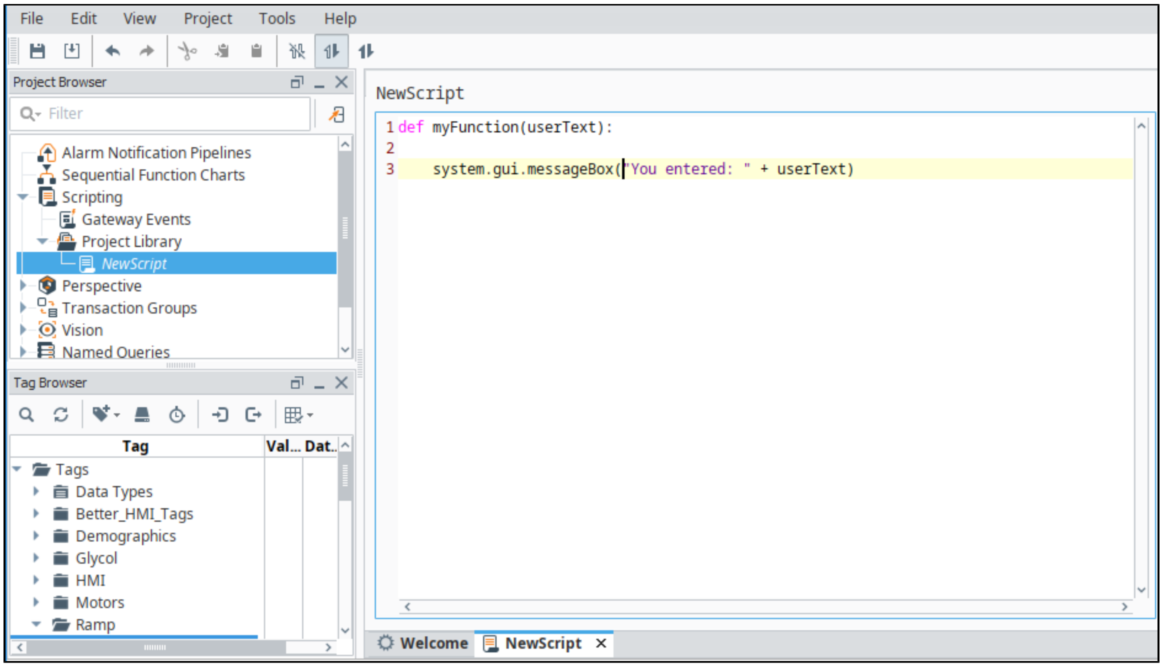
More information on these events can be found on the Client Event Scripts, Gateway Event Scripts, and Perspective Session Event Scripts pages.



```
# When the Gateway starts, write to a tag and log a message to the
# Gateway console

system.tag.writeBlocking(["[default]myTag"], [1])
logger = system.util.getLogger("Startup Logger")
logger.info("The system has started.")
```

Project Scripts

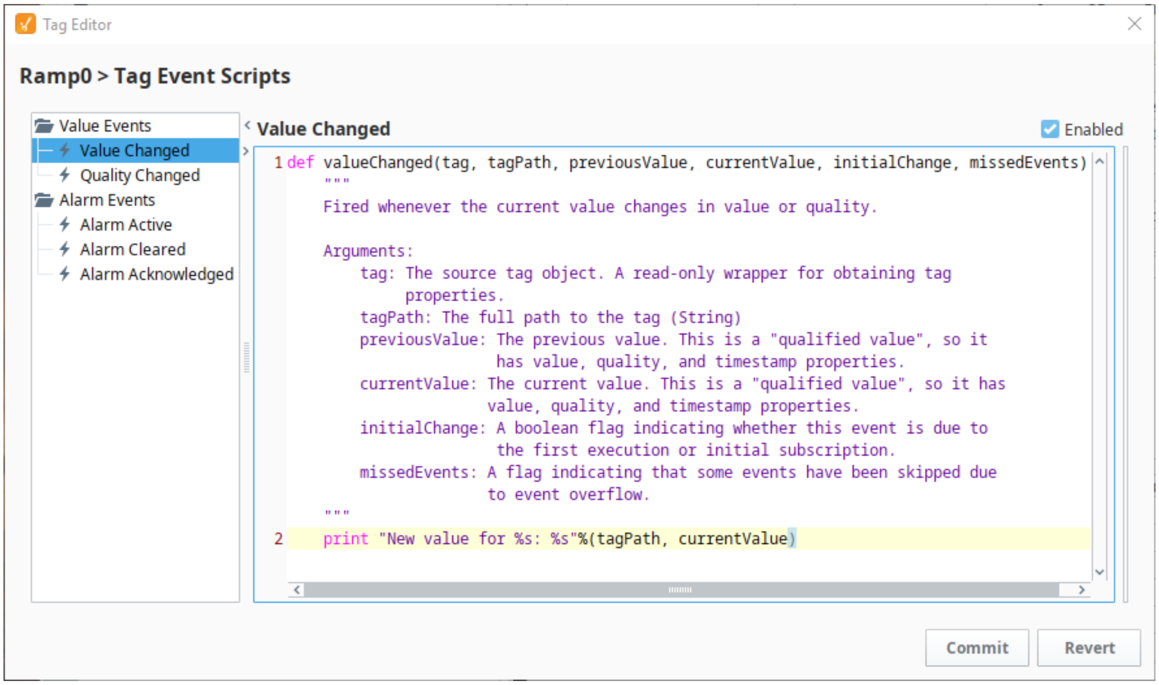
You can create your own reusable blocks of code in the Project Library. Once configured, these functions can be called from anywhere in a project, just like our system.* functions.



```
def myFunction(userText):

    system.gui.messageBox("You entered: " + userText)
```

Tag Scripts

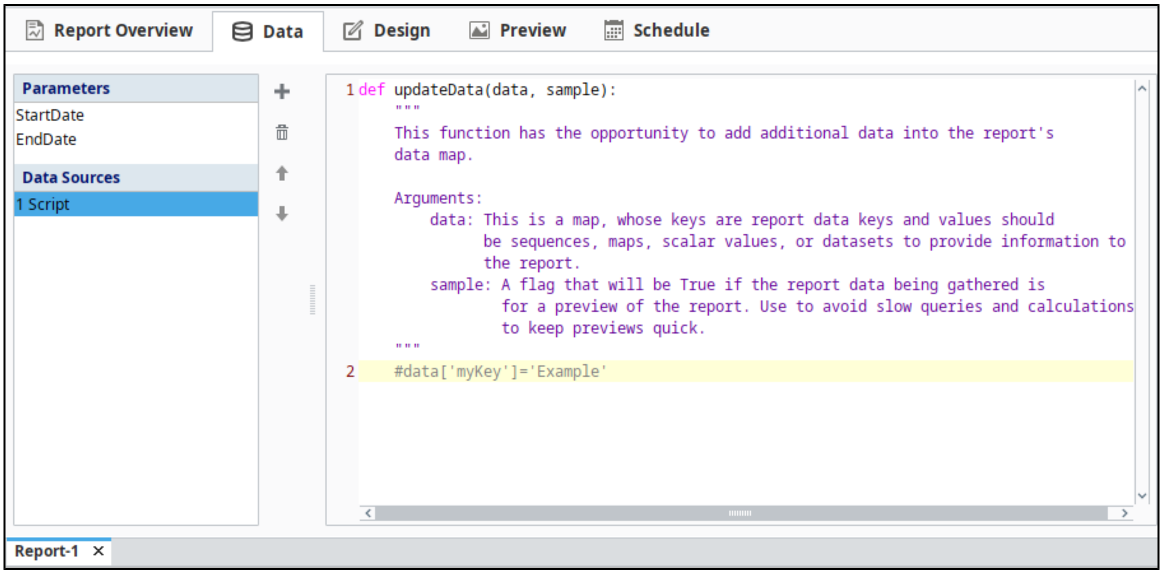
Once Enabled, these scripts are fired whenever a Tag value changes or an alarm event happens. You can use them for additional diagnostics, to set additional Tags, or to react to an alarm event. Because these events are on Tags, they are Gateway Scoped.



```
def valueChanged(tag, tagPath, previousValue, currentValue, initialChange, missedEvents):
	"""
	Fired whenever the current value changes in value or quality.

	Arguments:
		tag: The source tag object. A read-only wrapper for obtaining tag
		     properties.
		tagPath: The full path to the tag (String)
		previousValue: The previous value. This is a "qualified value", so it
		               has value, quality, and timestamp properties.
		currentValue: The current value. This is a "qualified value", so it has
		              value, quality, and timestamp properties.
		initialChange: A boolean flag indicating whether this event is due to
		               the first execution or initial subscription.
		missedEvents: A flag indicating that some events have been skipped due
		              to event overflow.
	"""
	print "New value for %s: %s"%(tagPath, currentValue)
```

Reporting

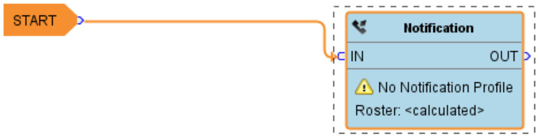
Reporting uses scripting in many different ways to help increase the effectiveness of the report. Scripting in Reports is used to create and modify data sources, manipulate charts, and set up a script as a scheduled report action.

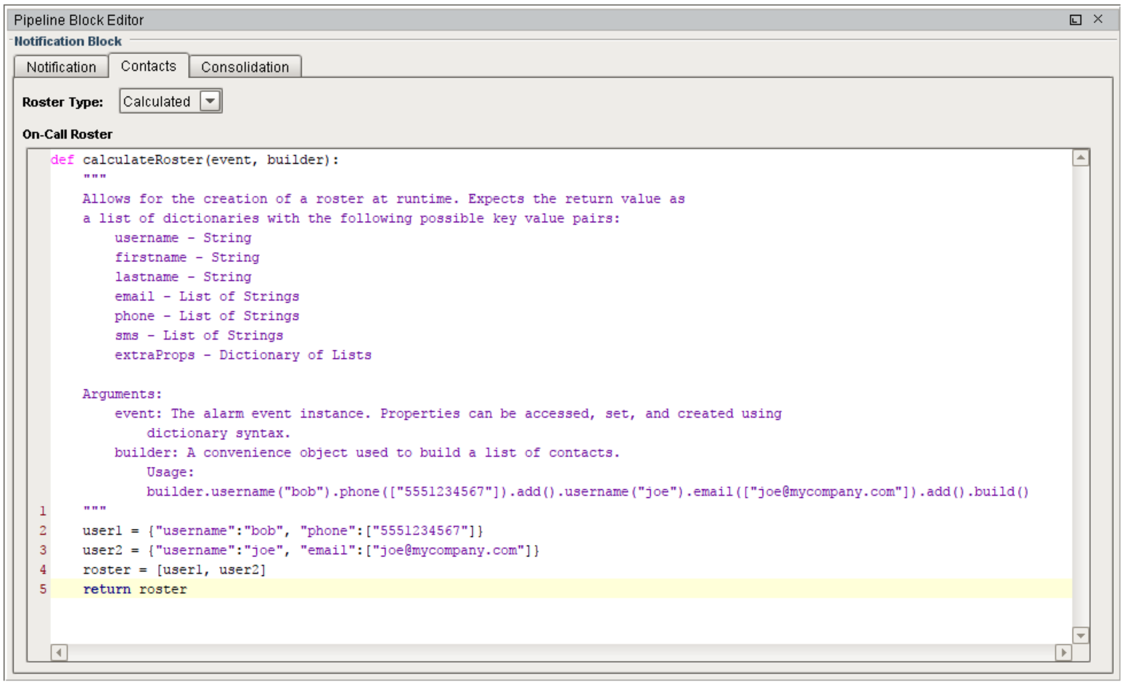


```
def updateData(data, sample):
    """
    This function has the opportunity to add additional data into the report's
    data map.

    Arguments:
        data: This is a map, whose keys are report data keys and values should
              be sequences, maps, scalar values, or datasets to provide information to
              the report.
        sample: A flag that will be True if the report data being gathered is
                for a preview of the report. Use to avoid slow queries and calculations
                to keep previews quick.
    """
    #data['myKey']='Example'
```

Alarming

The Alarm Notification system can also use scripting to great effect. A script block allows a script to be run within the pipeline, allowing data to be manipulated as the alarm event travels through the pipeline. Additionally, scripting can be used to generate a custom roster of users at runtime, giving full customization to who gets notified by the alarm event.





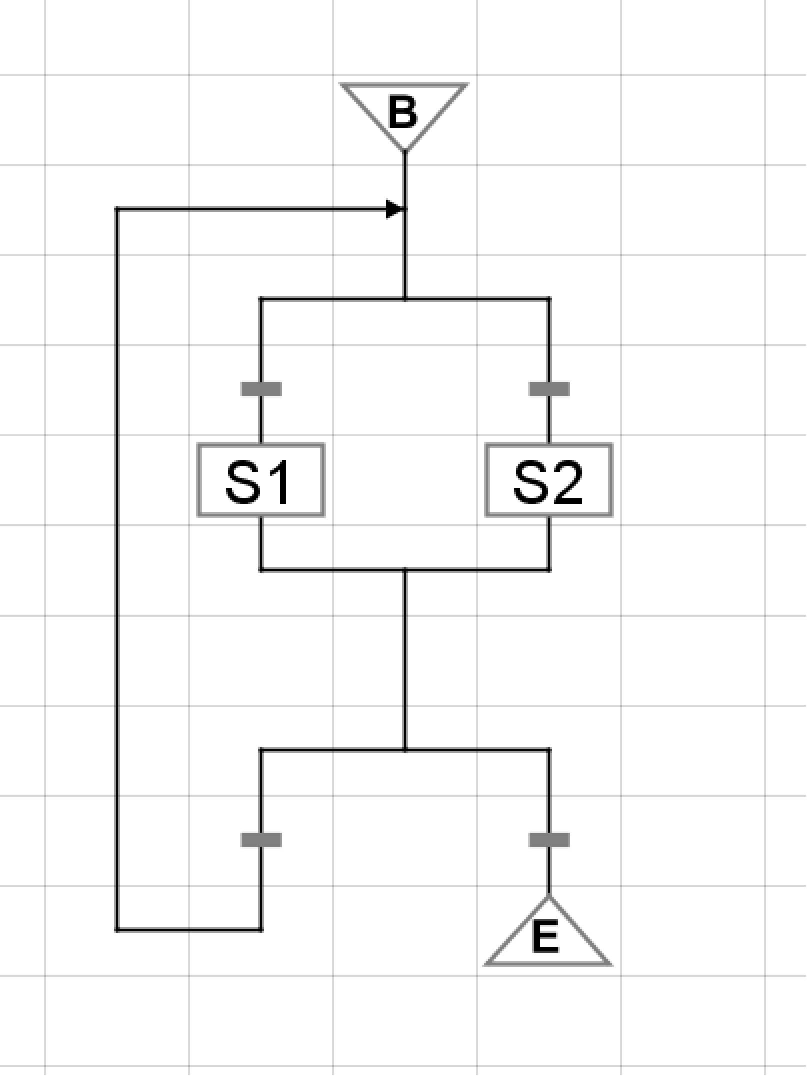
Roster Type: Calculated

On-Call Roster

```
def calculateRoster(event, builder):
    """
    Allows for the creation of a roster at runtime. Expects the return value as
    a list of dictionaries with the following possible key value pairs:
        username - String
        firstname - String
        lastname - String
        email - List of Strings
        phone - List of Strings
        sms - List of Strings
        extraProps - Dictionary of Lists

    Arguments:
        event: The alarm event instance. Properties can be accessed, set, and created using
            dictionary syntax.
        builder: A convenience object used to build a list of contacts.
            Usage:
            builder.username("bob").phone(["5551234567"]).add().username("joe").email(["joe@mycompany.com"]).add().build()
    """
    user1 = {"username":"bob", "phone":["5551234567"]}
    user2 = {"username":"joe", "email":["joe@mycompany.com"]}
    roster = [user1, user2]
    return roster
```

Sequential Function Charts

Sequential Function Charts (SFCs) are a flowchart of blocks that run scripts. They are executed in a specific sequential order along with some logic to potentially loop or call other charts. The scripts here can interact with the Gateway, and provide greater control when each step needs to complete before the next one can begin in multi-step processes.



Related Topics ...

- Scripting Data Source
- Sequential Function Charts
- Notification Block
- Tag Event Scripts
- Client Event Scripts
- Gateway Event Scripts
- Perspective Component Methods
- Perspective Session Event Scripts
- Project Library

In This Section ...

Getting Started with Scripting in Ignition

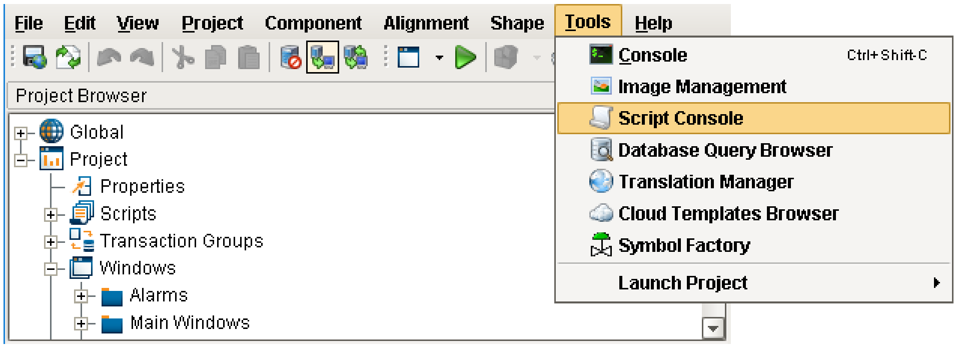
Overview

The best way to get started with scripting in Ignition is to write some very simple scripts and print them within Ignition. Let's take the phrase, "Hello World," and print it within Ignition. To show how flexibile Ignition is, there are actually a few ways to easily do this. One way is to use the Scripting Console, and another way is to use a Button component. Both examples are shown below.

Example - "Hello World" Using the Script Console

When testing or writing a new script, the Scripting Console is very useful since you can immediately get some feedback on the results of the script. When learning Python, it is a great place to start since you don't have to create a window or component before you begin writing your code.

1. In the **Designer**, open the Script Console from the Menu Bar: **Tools > Script Console.**

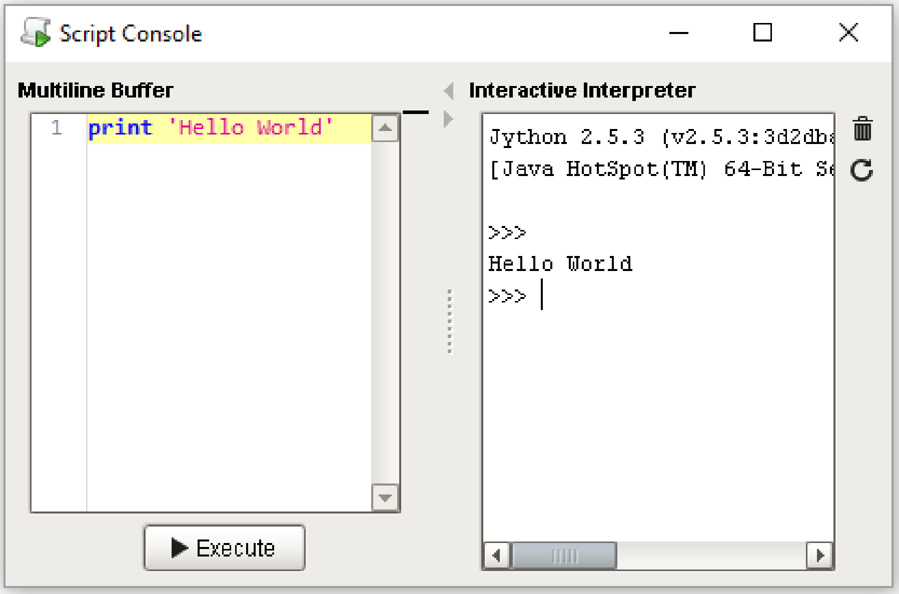


2. The Script Console will appear. Type the following code, or simply copy and paste it into the text area in the Multiline Buffer on the left side of the Script Console:

Python - Simple Print

```
print 'Hello World'
```

3. Click the **Execute** button at the bottom of the Script Console. You should see the message "Hello World" appear in the Interactive Interpreter on the right side of the Script Console.



```
print 'Hello World'
```

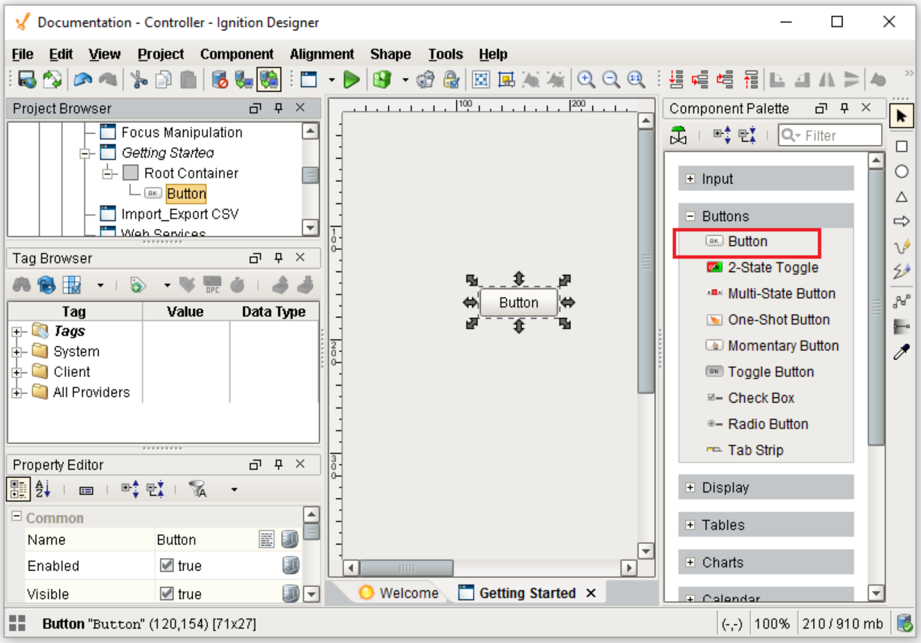
```
Jython 2.5.3 (v2.5.3:3d2db
[Java HotSpot(TM) 64-Bit Se

>>>
Hello World
>>> 
```

Example - "Hello World" on a Button Component

Scripts are commonly located on components in a window. In this example, let's add a script on a Button component, and print out "Hello World" when the Button is pressed.

1. In the **Designer**, create a new Main Window. Give it a meaningful name if you like. (We won't need to reference the name of the window in any of these examples).

2. Drag a standard **Button** component to your window.



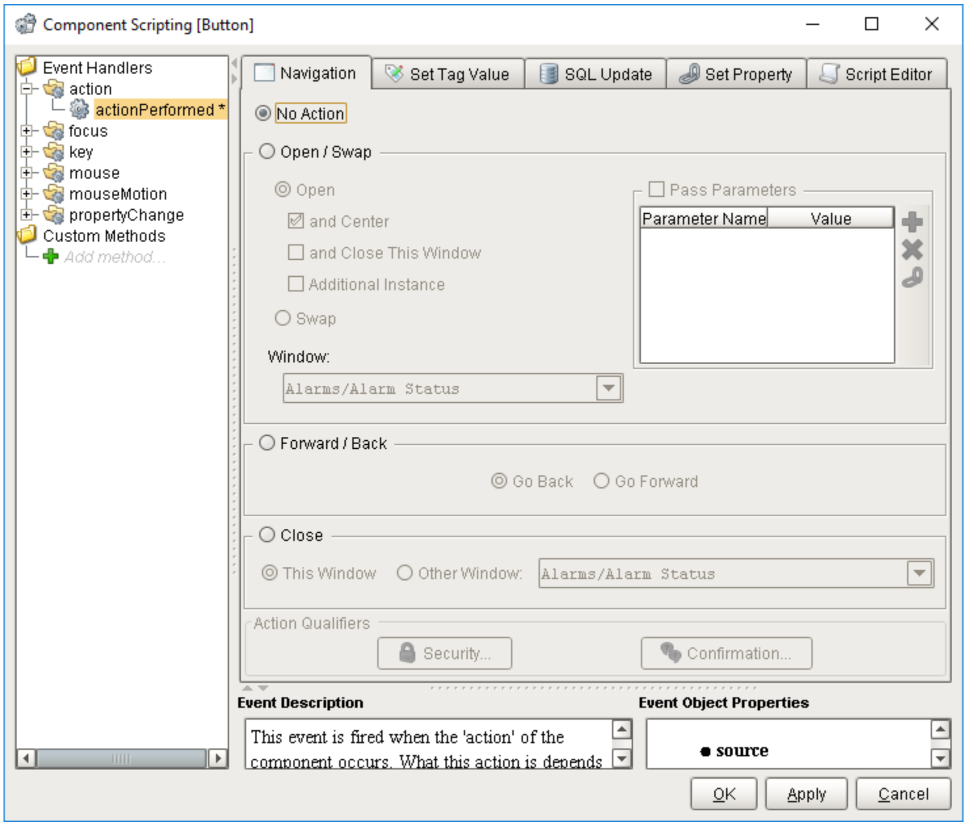
Name	Button
Enabled	☑ true
Visible	☑ true

Button "Button" (120,154) [71x27]

3. Let's add a script that triggers when the Button is pressed. With the Button component selected, right click on **Scripting** to open the **Component Scripting** window. Alternatively, you can double-click the Button to open the scripting window.

4. Under Event Handlers, select the **action > actionPerformed** option, and click the **Script Editor** tab. Make sure the **actionPerformed** event is highlighted. If an event is not selected, all of the remaining features in this window will be disabled, so we will not be able to

write a script. Additionally, if the wrong event is selected, then our script will not trigger when we expect it to.



● No Action

Event Description

This event is fired when the 'action' of the component occurs. What this action is depends

Event Object Properties

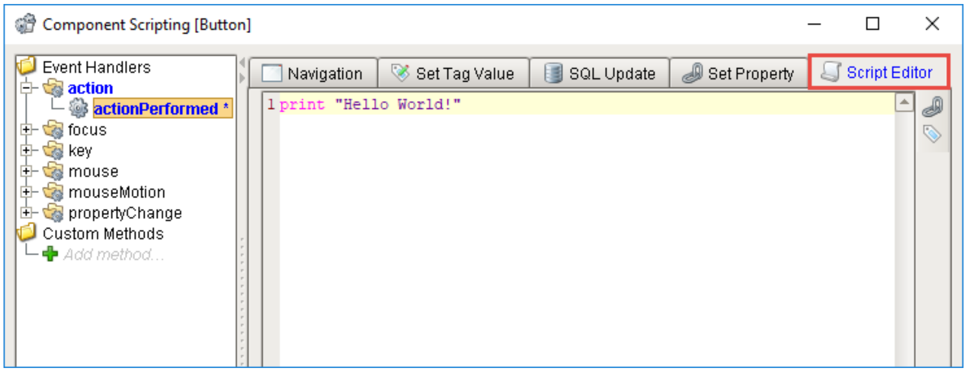
- source

5. In the Script Editor, you will see a large editable Text Area. This is where we will type our script.

6. Let's generate a message that shows "Hello World!" Use the following code to get started. Make sure the word "print" lines up exactly to the left edge. Indention in Python means something, so we need to avoid starting our lines with a space or tab unless we're denoting a block of code under something like an if-statement or function definition. Click **OK** to close the **Component Scripting** window.

Python - Simple Print

```
print "Hello World!"
```



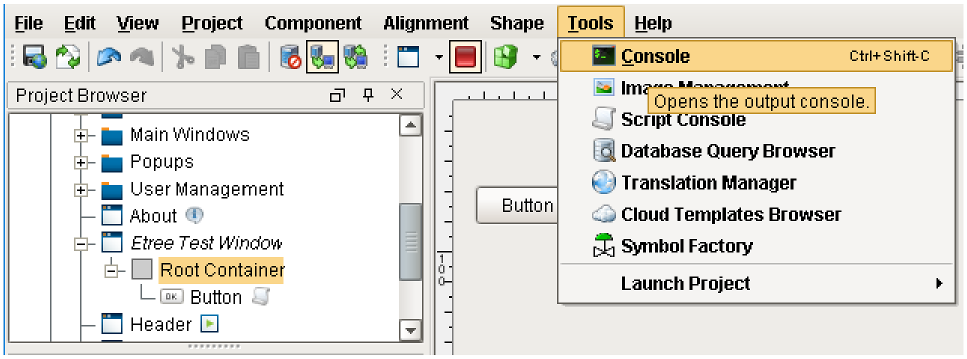
```
print "Hello World!"
```

Notice the **actionPerformed** event is blue and bold. This means there is a script on this event. This is useful to know in situations where a component has scripts on multiple events.

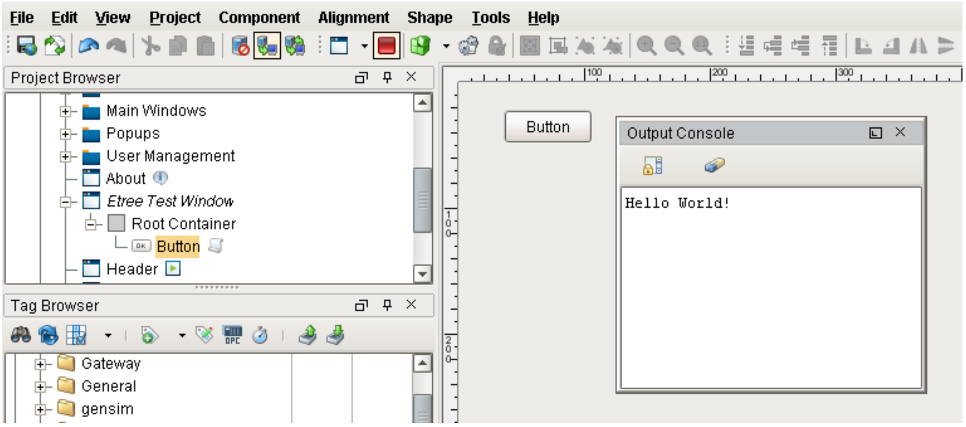
Additionally, an asterisk character (*) is next to the event. This means you have not applied/saved the changes to the script. The asterisk will disappear when you press either **OK** or **Apply**, and reappear whenever you make a change to the script. If you see this, then it means you may want to save any changes you made.

The **Script Editor** tab also has a blue color, denoting where the script is. An event will only ever have a script located on a single tab at any time. If a new tab is selected and configured, it will wipe out the work on the prior tab, for example, writing a script on **Script Editor** and then configuring the **Navigation** tab will erase the script on the **Script Editor** tab.

7. Now we can test the script. Place the **Designer** into Preview Mode, and press the Button. If everything is working as intended, then it should appear as if nothing happened. This is because we used `print` in our script, which always outputs to a console, as opposed to popping up at the user. This means we need to open the Designer's **Console** to see the results of our script. At the Designer's menu bar, select **Tools > Console**. This will make the console appear.



8. There may be a large amount of text in the Output Console. The Designer logs many different types of activities and events here, including polling events from components on other windows that are currently open in the Designer. However, the most recent events should be towards the bottom. As a quick tip, you can also click the eraser icon to clear out the console, and then press your **Button** again to generate a new entry like we did in this example.



```
Hello World!
```

We can now see where print statements go when called from a component, but this isn't too useful, as we don't want our users to open the console to see messages. Print statements are very useful when troubleshooting a problematic script, or even when testing a new script. Keep this in mind as you start to delve into scripting more.

Example - Using a Message Box

Now that we've seen how to print to the Output Console, lets make our message appear when we call it. This time, we will modify the script on the Button component to bring up a window that the user will see.

1. Open the **Component Scripting** window again, and find the script on the Button.

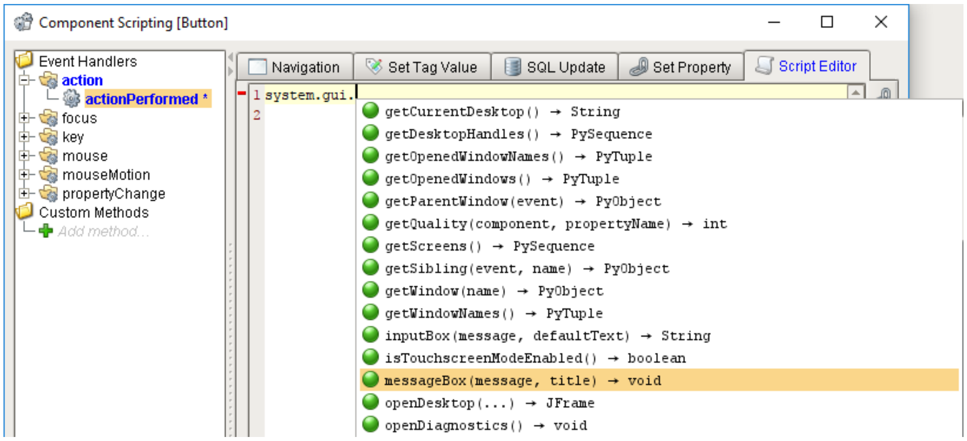
2. We will use one of Ignition's built-in functions, called system.gui.messageBox, to display the message. This will make a message box appear, which is a modal window that we can pass a string to. Remove your old code on the Button, as we will be replacing it. Start by typing the following:

Pseudocode - Built-in Functions Begin with `system.`

```
system.
```

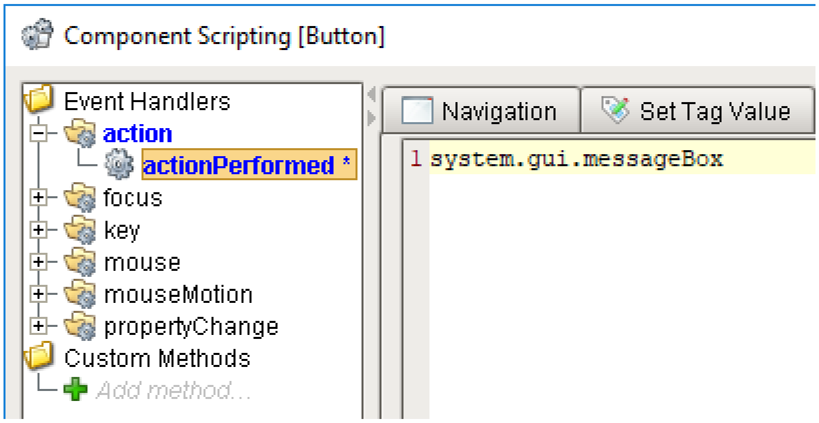
3. With the Text Cursor just to the right of the ".", hold The **Ctrl/Cmd** key and press the **Spacebar**. This will make the Autocompletion popup appear. This lists all of the available system functions. Start by clicking on **gui**, and then **messageBox**. Note, that you can still

type while the popup is open to filter the results in the list.


Component Scripting [Button]
Event Handlers
action
actionPerformed *
focus
key
mouse
mouseMotion
propertyChange
Custom Methods
Add method...
Navigation
Set Tag Value
SQL Update
Set Property
Script Editor
getCurrentDesktop() → String
getDesktopHandles() → PySequence
getOpenedWindowNames() → PyTuple
getOpenedWindows() → PyTuple
getParentWindow(event) → PyObject
getQuality(component, propertyName) → int
getScreens() → PySequence
getSibling(event, name) → PyObject
getWindow(name) → PyObject
getWindowNames() → PyTuple
inputBox(message, defaultText) → String
isTouchscreenModeEnabled() → boolean
messageBox(message, title) → void
openDesktop(...) → JFrame
openDiagnostics() → void

```
system.gui.
```

4. Once you've selected **messageBox**, click on it, and it should start some of the code for you automatically.



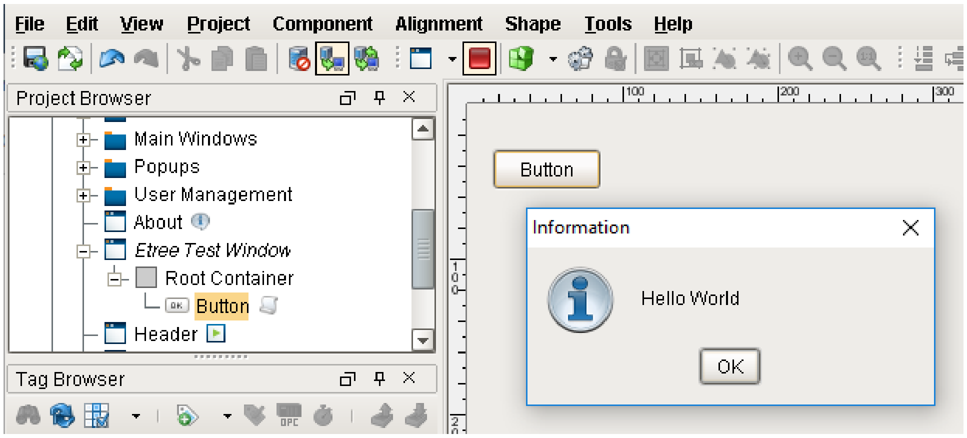
```
system.gui.messageBox
```

5. Complete the code by placing some parentheses and a message as a string. Alternatively, you can copy the example below.

Python - Using a System Function to Write to a Message Box

```
system.gui.messageBox('Hello World')
```

6. Place the **Designer** into **Preview Mode**, and click your Button. You will see a Message Box appear displaying the message we passed to the function.



Hello World

Related Topics ...

- Python Scripting
- Scripting Functions
- Component Events

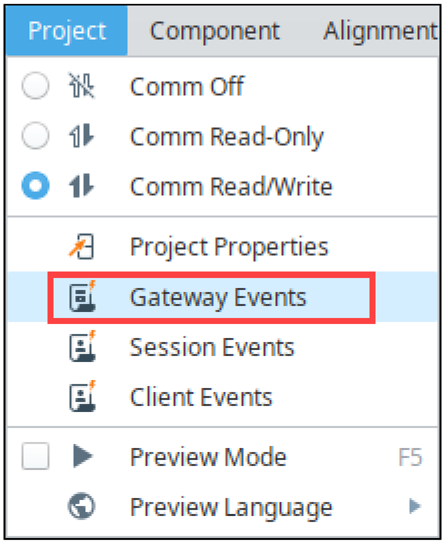
Gateway Event Scripts

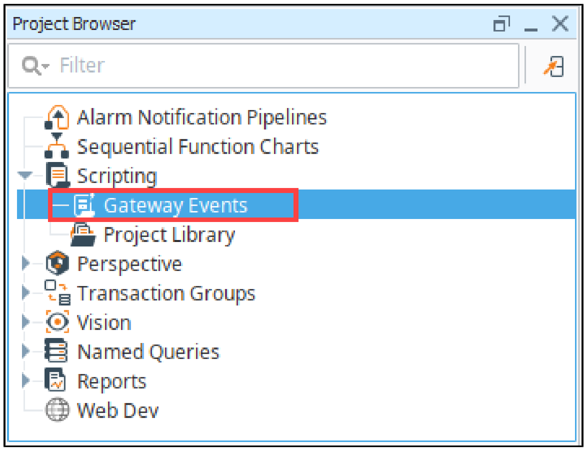
Gateway Event Scripts Overview

Gateway Event Scripts are scripts that run directly on the Gateway. They are useful because they always run, regardless if any sessions or clients are open. They also provide a guaranteed way to make sure only a single execution of a particular script occurs at a time, as opposed to placing a script in a window, as there could be multiple instances of the window open at a given point of time.

Note that even though Gateway Event Scripts run on the Gateway, they're still considered a project resource. Project backups will include any Gateway Event Scripts.

The Gateway Event scripting workspace is located in the Scripting menu of the Designer or in the Project Browser under **Scripting > Gateway Events**.





Other Event Scripts

The content on this page will focus primarily on Gateway Event Scripts. However, there is some overlap with Client Event Scripts, as they have similar events. More information can be found on the Client Event Scripts and Perspective Session Event Scripts pages.

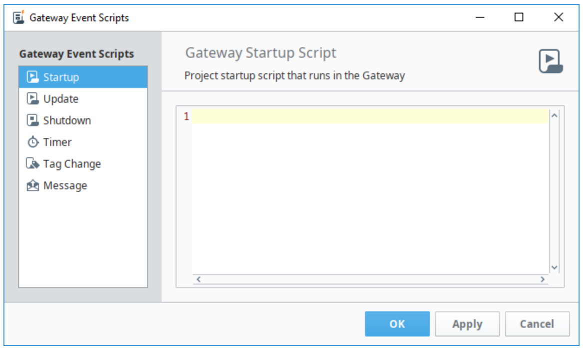
> ⚠ System functions are available for both Client Event Scripts and Gateway Event Scripts, but some system functions are specific to either one or the other. When you're writing event scripts, it's important to remember the scope of where you're writing the script: Client or Gateway. You can check Scripting Functions in the Appendix to see list of all system functions, their descriptions, and what scope they run in.

Gateway vs Client Event Scripts

Watch the Video

Startup Script

The Startup Script event runs at the startup of the Gateway. Additionally, if the project is restarted in someway, such as by making a change to a Gateway Event Script and saving, then the Startup Script will be called. This means that while editing scripts frequently in the Designer, the startup and shutdown events may happen frequently.



Startup Scripts

Watch the Video

Gateway Startup Behavior

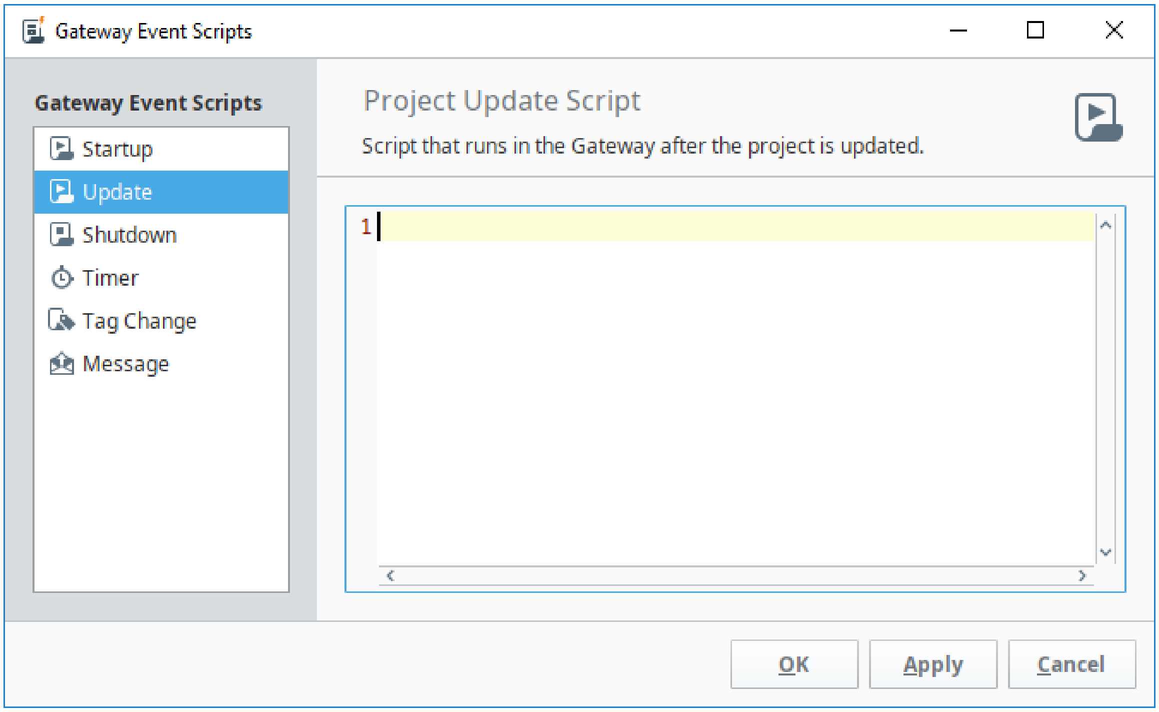
There is a specific order to when the various startup scripts are run. When troubleshooting your Gateway startup times, consider the following:

1. **Gateway starts** - The Gateway will start as an OS service, and start the context. No startup scripts can run before this is complete.
2. **Projects are started** - This includes all of the Gateway scoped items in the projects such as Transaction Groups, SFCs, etc. This does not refer to launching clients, and no clients can be automatically launched at this time. All **Gateway Startup Scripts** are run at this time for each project. Note: if you copied a project, always check for Gateway scoped events such as these. You generally don't want a Gateway Startup Script to run twice because it is in two projects.

The following feature is new in Ignition version **8.0.5**
Click here to check out the other new features

Update Script

The Update Script event runs after a project is saved or updated on the Gateway. This enables you to insert a script that will run every time a project is saved.



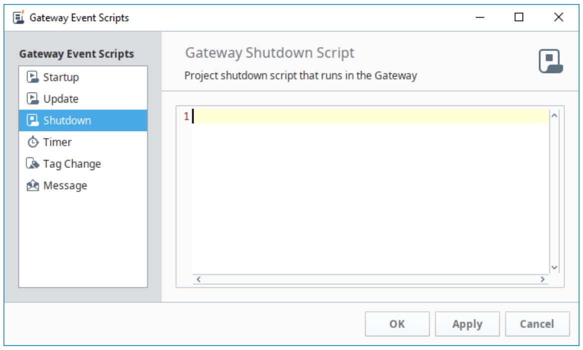
Project Update Script

Script that runs in the Gateway after the project is updated.

Shutdown Script

The Shutdown Script event runs at the shutdown of the project, which means it can be used as a way to trigger a script when the Gateway has to be restarted. It allows you to run a piece of code as the shutdown is occurring. After the script completes, the shutdown will finish.

Note that the Shutdown Script event only gets called if the Gateway is requested to shut down: if the computer power is lost abruptly (power outage, hard restart, etc.) this shutdown script will not run.



Gateway Shutdown Script

Project shutdown script that runs in the Gateway

Shutdown Scripts

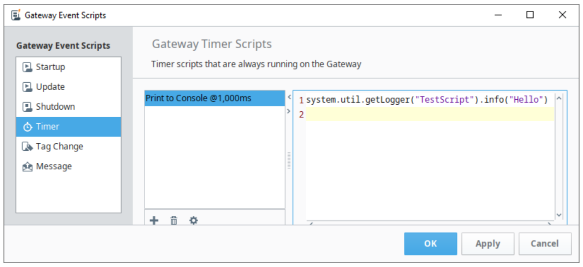
Watch the Video

Gateway Shutdown Behavior

Similar to how a Startup Script behaves, "Shutdown" in the context of this event means "project shutdown", so shutting down the Gateway would trigger this event, as well as disabling the project containing a script on this event. Additionally, making a change to a Gateway Event Script in the Designer, and then saving the project will cause the project to restart, which means this event can get called by simply making changes in the Designer and saving.

Timer Scripts

The Timer Scripts execute periodically on a timer at a fixed delay or rate. This allows you to set up a sort of heartbeat that can run on the Gateway. This is the ideal event to use if you need the Gateway to periodically perform some scripting task.



```
system.util.getLogger("TestScript").info("Hello")
```

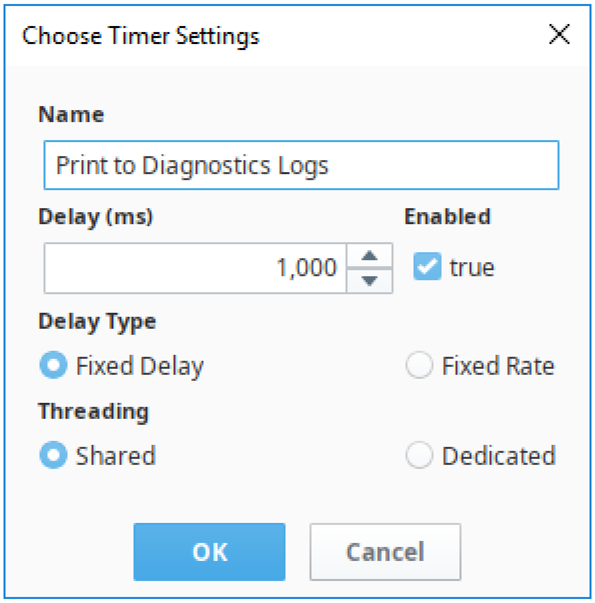
Timer Scripts

Watch the Video

Since multiple Timer Scripts can be added, there are separate buttons that allow you to manage each Timer Script.

- **Add Timer Script** - Adds a new Timer Script.
- **Remove Timer Script** - Will delete the selected Timer Script.
- **Modify Timer Script** - Will modify the settings for the selected Timer Script.

Timer Script Settings



Name: Print to Diagnostics Logs

Delay (ms): 1,000

Enabled: ☑ true

Delay Type: ● Fixed Delay ○ Fixed Rate

Threading: ● Shared ○ Dedicated

Below is an overview of the settings for a Timer Script.

- **Name**: The name of the Timer script. Names must be unique per project, so two timer scripts in the same project cannot have the same name.
- **Delay**: The delay period in milliseconds. The meaning of this setting is dependent on the **Delay Type** setting.
- **Enabled**: Allows you disable the Timer Script when set to false.
- **Delay Type**: Determines how the **Delay** setting is utilized.
  - A **Fixed Delay** timer script (the default) waits for the given **Delay** between each script invocation. This means that the script's rate will actually be the delay plus the amount of time it takes to execute the script. This is the safest option since it prevents a script from mistakenly running continuously because it takes longer to execute the script than the delay.
  - **Fixed Rate** scripts attempt to run the script at a fixed rate relative to the first execution. If the script takes too long, or there is too much background process, this may not be possible. See the documentation for java.util.Timer.scheduleAtFixedRate() for more details.
- **Threading**: Determines which thread this script should run in. In other words, this setting allows you to specify if you want this timer script to share execution resources or not. The rule of thumb here is that quick-running tasks should run in the shared thread, and long-running tasks should get their own dedicated thread.
  - The **Shared** setting means that all timer scripts will share a thread. This is usually desirable, as it prevents creating lots of unnecessary threads: threads have some overhead, so a small amount of resources are used per thread. However, if your script takes a long time to run, it will block other timer tasks on the shared thread.
  - The **Dedicated** setting will create a separate thread specifically for the timer script to use. This setting is desirable when your scripts executions must be as consistent as possible, as other timer scripts can't slowdown or otherwise impact the execution of a script in a separate thread.

Tag Change Scripts

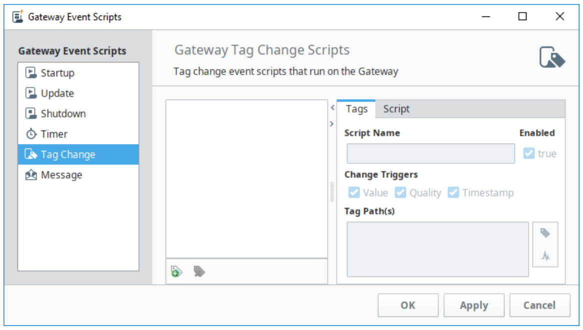
The Tag Change Script event allows you to specify any number of Tags, and trigger a script when one of them change. Since these execute based on a Tag changing value, Tag Change Scripts are ideal when you need a script to run based on some some signal from a PLC.

Having the Tag Change Scripts run in the Gateway Scope means that the scripts are active as long as the Gateway is running. Thus, you do not need a client or session to be open for a Gateway Tag Change Script to execute. When executing, each Tag Change Script runs in a separate thread. This prevents long running scripts from blocking the execution of other Tag Change Scripts.

Tag Change Scripts

Watch the Video

⚠ Due to their nature, Client Tags may not trigger Gateway Tag Change Scripts. However, there are other similar events, such as a Client Event Script, that enable you to trigger a script based on a Client Tag changing value.



Tag Change Script Interface

Tag List

Lists all of the available Tag Change scripts in the project. Two icons are available below the list:

- **Add Script** - Adds a new Tag Change Script to the list.
- **Remove Script** - Removes the currently selected script from the list.

Tags Tab

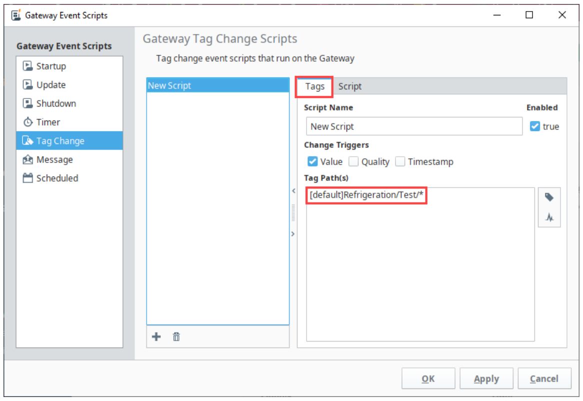
The **Tags** tab contains settings for the script. See the **Script Settings** description below

- **Script Name** - The name of the script. Script names must be unique per project.
- **Enabled** - Determines if the script is active or not. Set to false to disable the script.
- **Change Triggers** - When the Tag changes, the script can trigger based on the **Value**, **Quality**, and/or **Timestamp**. Note that regardless of how many triggers changed, the script only executes once per tag, so leaving all triggers enabled will not trigger three executions each time the Tag changes.
- **Tag Paths** - A list of Tag paths to monitor. When any of the Tags listed in this area change, the script will trigger. Note that the list is not comma separated: new paths are specified each line.
  - **Tag Browser** - Opens a Tag Browser window, allowing you to quickly lookup and add Tag paths to the Tag Change Script.
  - **Path Diagnostic** - Click this icon to verify the paths specified under the **Tag Paths** list. This is useful for checking for typos in the list.

The following feature is new in Ignition version **8.0.5**
Click here to check out the other new features

- Wildcards can be used at the end of configured tag paths. Using a wildcard will execute the script for all tags within the same folder. In the below example, the script "New Script" will run when the value of any tag in the "Test" folder changes.

> **Note:** Wildcards can only be used at the folder level and cannot be used at the tag level. For example, configuring a tag path like `[provider]folder/*` will execute a script on all tags within the folder, but `[provider]folder/ramp*` will not.



Script Name: New Script

Enabled: ☑ true

Change Triggers: ☑ Value ☐ Quality ☐ Timestamp

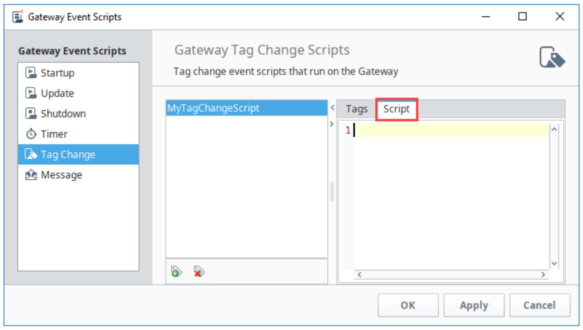
Tag Path(s): [default]Refrigeration/Test/*

Script Tab

The **Script** tab is where the Python script associated with this event will be placed. Two icons are available below the list:

Add Script - Adds a new Tag Change Script to the list.

Remove Script - Removes the currently selected script from the list.



Tag Change Objects

Tag Change Scripts contain several built-in objects that are useful for inspecting the event, such as seeing what value the Tag changed to. These objects are listed below:

The initialChange Value

A variable that is a flag (0 or 1) which indicates whether or not the event is due to the initial subscription or not. This is useful as you can filter out the event that is the inital subscription, preventing a script from running when the values haven't actually changed.

```
if not initialChange:

    # Do something useful here
```

The newValue Object

A QualifiedValue object that contains the following methods

Method	Description	Usage Example
`getValue()`	Returns the new value on the tag	`newValue.getValue()`
`getQuality()`	Returns the new quality on the tag	`newValue.getQuality()`
`getTimestamp()`	Returns the new timestamp value	`newValue.getTimestamp()`

The event Object

This object offers some additional utility, such as accessing the previous values on the tag.

Method	Description	Usage Example
`getCurrentValue()`	Returns a QualifiedValue object (similar to the newValue object), representing the new values on the tag.	`event.getCurrentValue().getValue()`
`getPreviousValue()`	Returns a QualifiedValue object, representing the values on the tag before the change.	`event.getPreviousValue().getValue()`
`getTagPath()`	Returns a TagPath object, that can be further examened for details about the path on the tag that changed. See the TagPath Object table below. Additionally, the TagPath object can easily be turned into a string, providing quick access to the path of the tag that changed value.	`event.getTagPath()`
`getValue()`	Returns the new value on the tag, similar to getCurrentValue(). This method is a convenient way to retrieve just the new value, without needing to interact with the QualifiedValue object returned by getCurrentValue().	`event.getValue()`

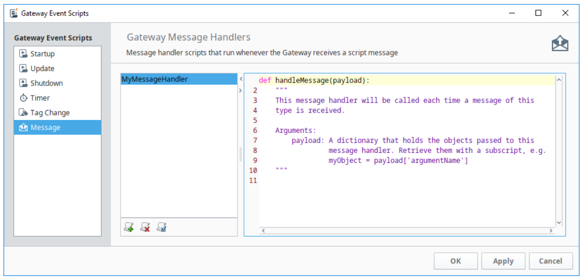
The TagPath Object

These methods are available on the TagPath object returned by `event.getTagPath()` calls

Method	Description	Usage Example
`getItemName()`	Returns the name of the item at the end of the path, which can be used to get the name of the tag that changed.	`event.getTagPath().getItemName()`
`getParentPath()`	Returns the path to the Tag's parent folder.	`event.getTagPath().getParentPath()`

Message Scripts

Message Handlers allow you to write a script that will run in the scope they are located in, but they can be invoked by making a call from other projects or even other Gateways. They can be called using three different scripting functions: system.util.sendMessage, system.util.sendRequest, and system.util.sendRequestAsync.



```
def handleMessage(payload):
    """
    This message handler will be called each time a message of this
    type is received.

    Arguments:
        payload: A dictionary that holds the objects passed to this
                 message handler. Retrieve them with a subscript, e.g.
                 myObject = payload['argumentName']
    """
```

Under the list of handlers, three small buttons allow you to add, remove and manage your handlers.

Add Message Handler - Will add a message handler.

Remove Message Handler - Will delete the highlighted message handler.

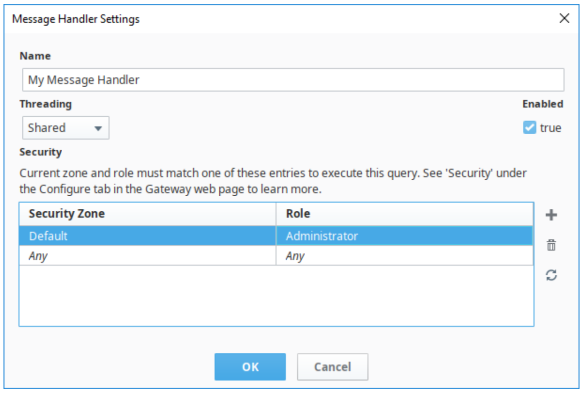
Modify Message Handler - Will modify the settings for the highlighted message handler.

Script Messaging

Watch the Video

Gateway Message Handler Settings

When adding or modifying a message handler, a Message Handler settings window will popup.



Name: My Message Handler

Threading: Shared

Enabled: ☑ true

Security

Current zone and role must match one of these entries to execute this query. See 'Security' under the Configure tab in the Gateway web page to learn more.

Security Zone	Role
Default	Administrator
Any	*Any*

The following settings are available:

Name - The name of the message handler. Each message handler must have a unique name per project.

Threading - Determines the threading for the message handler. Contains the following options:

Shared - The default way of running a message handler. Will execute the handler on a shared pool of threads in the order that they are invoked. If too many message handlers are called all at once and they take long periods of time to execute, there may be delays before each message handler gets to execute.

Dedicated - The message handler will run on its own dedicated thread. This is useful when a message handler will take a long time to execute, so that it does not hinder the execution of other message handlers. Threads have a bit of overhead, so this option uses more of the Gateway's resources, but is desirable if you want the message handler to not be impeded by the execution of other message handlers.

Security - Allows you to specify security zone and role combinations that are allow to request this message handler.

The Payload

Inside the message handler is your script. The script will have a single object available to it, the **pay load**. The payload is a dictionary containing the objects that were passed into it. In essence, the payload is the mechanism that allows you to pass the message handler values.

The payload is simply a python dictionary, so extracting values involves specifying the key:

Pseudocode - Payload Values

```
value1 = payload["MyFirstValue"] #"MyFirstValue" is the key that is
associated with a value. We are taking the value associated with
MyFirstValue, and assigning it to value1.
value2 = payload["MySecondValue"] #Similarly, we are taking the value
associated with MySecondValue and assigning it to value2.
```

Calling Message Handlers

Once you have your message handlers created, you can then call them from a script using one of three scripting functions: system.util.sendMessage, system.util.sendRequest, and system.util.sendRequestAsync. These functions allow you to call a message handler in any project, even if the project that the message handler resides on is different from the one you are calling it from. The message handler will then execute in the scope in which it was created, and will use any parameters that you pass in through the payload.

Pseudocode - Calling a Message Handler

```
project="test"
messageHandler="My Message Handler"
myDict = {'MyFirstValue': "Hello", 'MySecondValue': "World"}
results=system.util.sendMessage(project, messageHandler, myDict)
```

Troubleshooting Gateway Scripts

While they are technically project resources, remember that Gateway Event Scripts technically run on the Gateway. Thus the Status section of the Gateway is useful for diagnosing issues with Gateway Event Scripts.

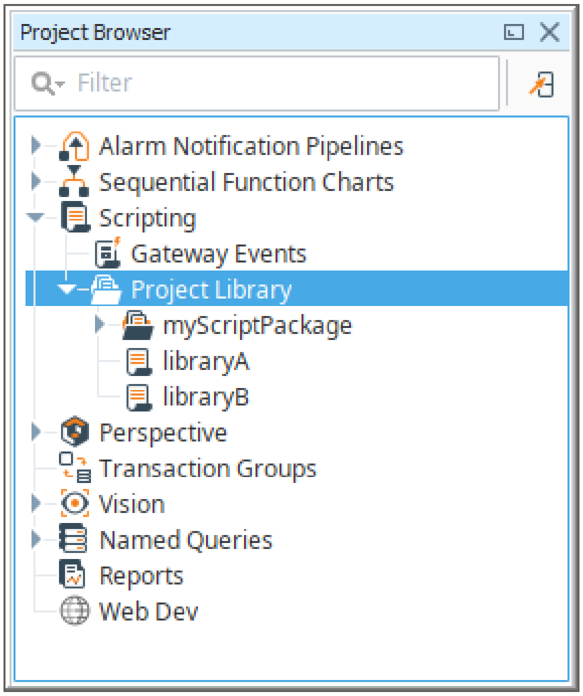
Related Topics ...

Client Event Scripts

Perspective Session Event Scripts

Project Library

Scripts under the Project Library (called "Project Scripts") are a project-based resource that allows user created Python scripts to be configured. Objects and functions created in a project script can be called from anywhere in the project. Project Scripts are accessible from the Project Browser, under the **Scripting** item.

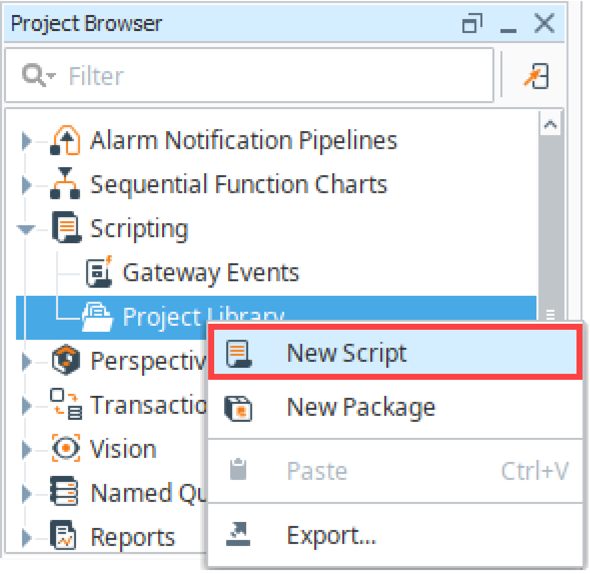


Additionally, a single project can be designated as the **Gateway Scripting Project**, meaning that scripts defined in the stated project can be called from the Gateway scope.

On this page ...

Add a Script

To add a a project script, simply right click the **Project Library** item and click the **New Script** option.



Scripts and Packages

There are two main types of resources under the Project Library.

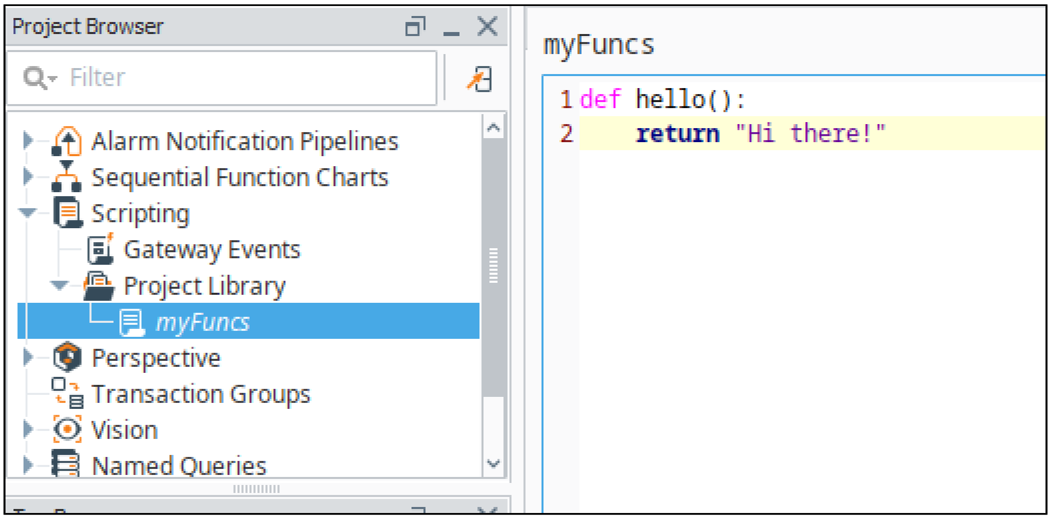
- **Scripts** - Each script resource can contain many functions and objects.
- **Packages** - Each Package effectively acts as a folder, allowing you to better organize each script resource.

Usage Example

For example, let's suppose you added the following script module named **`myFuncs`**, whose body is shown below.

Python

```
def hello():
        return "Hi there!"
```



```
def hello():
    return "Hi there!"
```

Once we **save** our project, we can now call this function from anywhere within the project using the following syntax

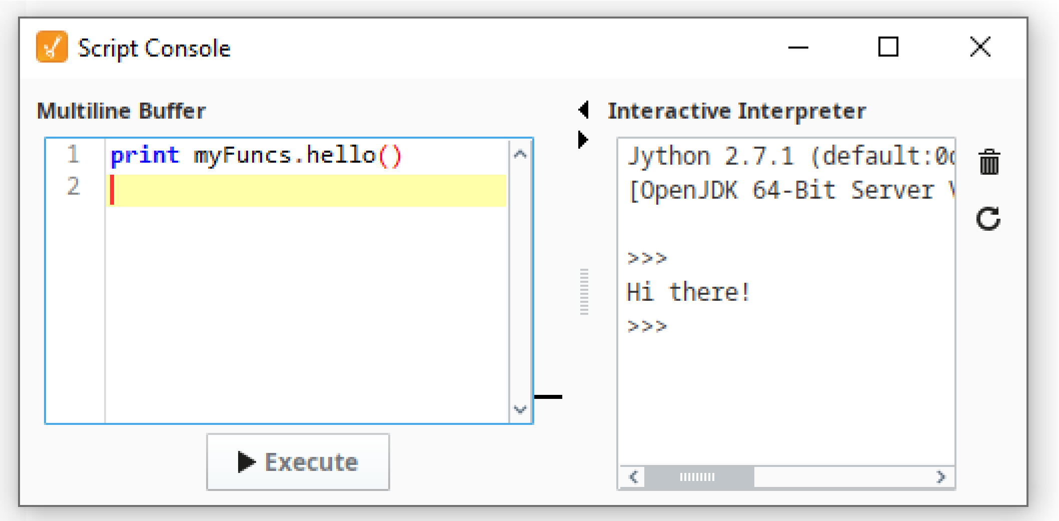
Python - Calling the Project Script

```
myFuncs.hello()
```

Don't Forget to Save

Project scripts are not accessible to the other resources until the project is saved.

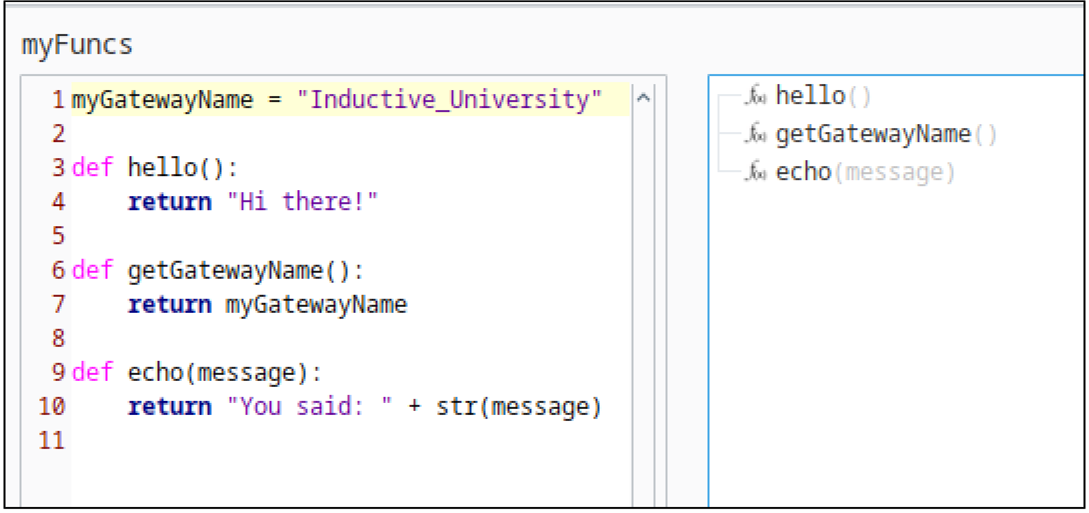
For example, we could open the Script Console (Tools menu > Script Console), write the following, and execute the script.



```
print myFuncs.hello()
```

```
Jython 2.7.1 (default:0c
[OpenJDK 64-Bit Server \

>>>
Hi there!
>>>
```

Each script resource can contain multiple functions and objects. Note that as you add new function definitions, the list on the right will populate, allowing you a quick way to navigate through long scripts.



```
myGatewayName = "Inductive_University"

def hello():
    return "Hi there!"

def getGatewayName():
    return myGatewayName

def echo(message):
    return "You said: " + str(message)

```

Project Libraries and Execution

Because Python is a dynamic language, any code inside of a project library must be run to build the function and class definitions. This is a common behavior across interpreters. Within Ignition, these project libraries will run under certain conditions. For example, such as when the Script Console in the Designer starts up, if changes are made to third-party libraries inside of the gateway's installation directory, when saving changes to these project libraries, and several other conditions.

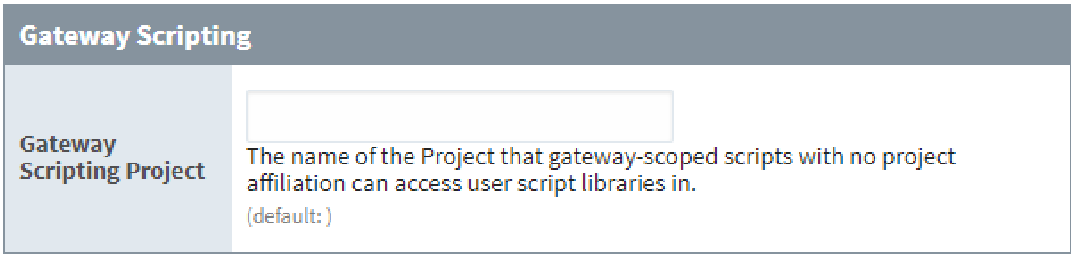
When this occurs, code inside of a project library is executed. Meaning classes and functions are defined, and any code that is not contained within either a class or function will execute.

Because of this process, it's generally recommended that all code within a project library is wrapped inside of a function or class definition.

Gateway Scripting Project

Project scripts are normally only accessible from the project they were defined in. Thus objects that exist in other scopes, such as Tags that exist in the Gateway scope, are unable to call project scripts. Attempting to do so will result in Gateway log errors stating that "global name 'yourScript' is not defined'.

The exception to this rule is the Gateway Scripting Project. This project is specified by the **Gateway Scripting Project** property, which is set in the Config section of the **Gateway** Webpage under Gateway Settings. Entering in the name of a project under this property allows the Gateway access to project scripts configured in the specified project.



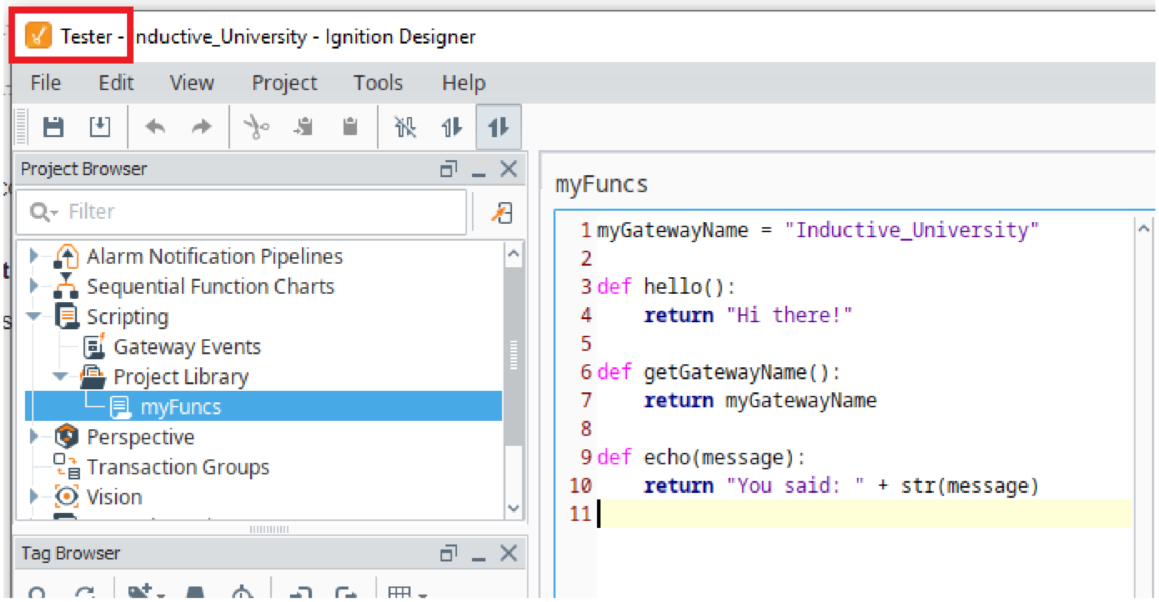
Gateway Scripting

Gateway Scripting Project

The name of the Project that gateway-scoped scripts with no project affiliation can access user script libraries in.

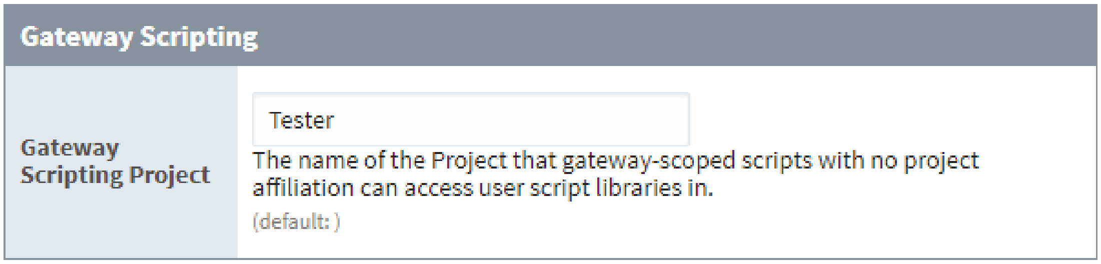
(default:)

Thus, if the **myFuncs** library in the prior section was configured in a project named "Tester"



```
myGatewayName = "Inductive_University"

def hello():
    return "Hi there!"

def getGatewayName():
    return myGatewayName

def echo(message):
    return "You said: " + str(message)

```

We could use that name in the Gateway Scripting Project.



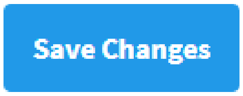
Gateway Scripting

Gateway Scripting Project

Tester

The name of the Project that gateway-scoped scripts with no project affiliation can access user script libraries in.

(default:)



After we save, Tags and other Gateway-scoped resources can then start calling any of the scripts from our project library in the "Tester" project.

Example

This example demonstrates how to call scripts in your Gateway Scripting Project from a Tag.

1. Set up your Gateway Scripting Project under **Config > Gateway Settings:**

2. Open your Gateway Scripting Project and add a library called **myFuncs**.
3. Define a new function in this library called **thankYou:**

   ```
   def thankYou(user):
           system.gui.messageBox("Thanks for noticing me, " + user)
   ```

4. Save your Gateway Scripting Project and exit.
5. Open a new project.
6. Choose a Tag and configure a Tag Event Script on the Alarm Acknowledged event.
7. You can call any function within the **myFuncs** library in your Gateway Scripting Project:

   ```
   myFuncs.thankYou(user = ackedBy)
   ```

   Note: You do not need to specify the name of your Gateway Scripting Project when calling any of its functions.

Related Topics ...

- User Defined Functions
- Scripting in Ignition

Web Services, SUDS, and REST

Web Services Overview

Web services are software solutions that allow for interacting with machines residing on a network. In short, web services are nothing more than web pages for machines. They provide a standard way for a third party to request and receive data from a piece of hardware on the network without having to know anything about how that machine works.

On this page

...

Protocols

There are two common approaches to Web Services in Ignition: making a HTTP (HyperText Transfer Protocol) method call, or making a SOAP (Simple Object Access Protocol) call.

- HTTP methods, such as GET and POST, are called using the built-in system functions, such as system.net.httpGet(). More information on HTTP calls can be found on the HTTP Methods page.
- SOAP is XML based, and typically requires a third party Python library. There are many third party Python libraries that utilize SOAP, however, documenting them goes outside the scope of this manual (as do all third party libraries).

The approach you choose typically depends on the server you're trying to make calls to: more specifically, the protocol(s) it supports.

What About the SUDS Library?

The SUDS library, a library that used to come included with the Python Standard Library, offered SOAP based functionality. However, SUDS development has been halted, and is no longer included in the standard library.

In the interest of posterity, the legacy SUDS documentation has been condensed and can be found on the SUDS - Library Overview page. Note that the legacy documentation should be considered deprecated.

Can Ignition Make RESTful Calls?

Yes it can! However, it is important to understand that REST is an architecture, **NOT** a protocol. Instead, REST utilizes and describes how a protocol should be used. Thus, a RESTful architecture could use both of the protocols mentioned on this page, although HTTP is far more common.

Common Web Services Workflow

While all Web Services follow the same standards, they all do different things. They wouldn't be worth anything if you didn't get the information you need, or if they contained a lot of excess data. If you are unfamiliar with a particular Web Service, there are a few things that you can do to figure out what data is available and how to get it.

1. Identify a Web Service that you will be using. Usually the Web Service has an API somewhere documenting how requests should be made.
2. Write a script to pull some information from the Web Service. If using HTTP, this could mean starting with a GET call, where as SOAP would involve retrieving the WSDL (Web Services Description Language). In both cases, you may need to find a way to authenticate against the server (usually with some user credentials or an auth token, the API for the service would have more details).
3. Once you have the results from the GET/WSDL, identify the information or functions you want to use.
4. Write a script to use that function and return your values.
5. Parse the results and use them. This can be for display, saving to a database, or anything else you need.

Note: Web Services sometimes take a lot of time to return results, especially the first time they are called. If you put your Web Services script in a button, the client will freeze until the call is complete (this is because the event handlers are run on the GUI thread). It's a good idea to use system.util.invokeAsynchronous() or add a waiting image to your screen to let the user know Ignition is working as expected.

Related Topics ...

- WebDev Module

In This Section ...

HTTP Methods

Overview

Web services calls typically require some protocol to make requests. HTTP is an incredibly common protocol, so this page will introduce how to incorporate these calls in a Python script. Note that all of the examples on this page can be easily called with the Script Console, but can be utilized through some other means (like the actionPerformed event on a Button).

On this page ...

Finding an Endpoint

Ignition doesn't natively expose an endpoint for web services calls, so we'll have to utilize a service of some sort for the examples on this page. Fortunately, there are many public services we can utilize, such as OpenWeather. From here, we can generate an endpoint to use. At the time of this writing, we're using the following endpoint URL template:

Endpoint URL Template

```
https://api.openweathermap.org/data/2.5/weather?lat={lat}&lon={lon}
&appid={API key}
```

The following table describes the various parameters in the endpoint URL:

Parameter Key	Description
lat, lon	Geographical coordinates (latitude, longitude).
appid	Your unique API key. **Note:** The code snippets on this page use "Your API key" as a placeholder for unique API keys. You will need to substitute the placeholder with your own API key for the code snippets to function as expected.

The following code snippets are using the geographical coordinates for the Inductive Automation corporate headquarters located in Folsom, CA.

Making the Call

Note: The code snippets on this page use system.net.httpClient() to fetch the results of the API call. Since the system.net.httpClient() system function was introduced in Ignition version 8.0.6, users running Ignition version 8.0.0 through Ignition version 8.0.5 can use system.net.httpGet() instead.

To retrieve the results of this information in Ignition, we can use system.net.httpClient() to fetch the results of this call. We can try the following script in the Scripting Console:

Python - Creates a variable to Store the Endpoint and Retrieves Results

```
#Set the endpoint URL
url = "https://api.openweathermap.org/data/2.5/weather"

#Declare a variable for system function we are using
myClient = system.net.httpClient()

#Declare a variable and set the parameters for the endpoint URL
#Instead of specifying the API call parameters for the endpoint URL in the beginning,
response = myClient.get(url, {"lat":38.652330, "lon":-121.189773, "appid":"Your API key"})

#Print the output
print response.getText()
```

Printing this results in the following Python string:

Python - Results

```
{"coord":{"lon":-121.1898,"lat":38.6523},"weather":[{"id":800,"main":"Clear","description":"clear sky","
icon":"01d"}],"base":"stations","main":{"temp":310.34,"feels_like":310.11,"temp_min":308.88,"temp_max":
312.74,"pressure":1013,"humidity":26},"visibility":10000,"wind":{"speed":4.12,"deg":290},"clouds":{"all":
0},"dt":1654893775,"sys":{"type":2,"id":2006213,"country":"US","sunrise":1654864794,"sunset":
1654918128},"timezone":-25200,"id":5349705,"name":"Folsom","cod":200}
```

Parsing the Results

If we wanted to extract a single value out of the results, we have a number of approaches. One useful approach would be to turn this JSON string into a Python Dictionary. This way we can single out a key instead of using regex or looking for substrings (both valid approaches in their own regard).

When presented with a JSON string, we can call system.util.jsonDecode() to turn a JSON string into a native Python Object. Thus, we can modify our code to the following:

Python - Parsing the Results Code

```
#Set the endpoint URL
url = "https://api.openweathermap.org/data/2.5/weather"

#Declare a variable for system function we are using
myClient = system.net.httpClient()

#Declare a variable and set the parameters for the endpoint URL
response = myClient.get(url, {"lat":38.652330, "lon":-121.189773, "appid":"Your API key"})

#Set a variable for the contents of the API call
results = response.getText()

# Convert the JSON string into a Python object. In this case, it results in a Dictionary.
decodedDict = system.util.jsonDecode(results)

# Now we can treat the results like a nested dictionary, thus we can specify the "weather" key,
# and then the nested "description" key to return a description of the current weather conditions.
# "[0]" is needed to specify the index of the list before specifying the "description" key.
print decodedDict.get("weather")[0].get("description")
```

Now we can easily retrieve a single value by specifying key names on the results. Printing this results in the following Python string (at the current time):

Python - Results

```
clear sky
```

Make the Results Human Readable

Now that we know how to extract the results, we should clean up the output of the GET call. The JSON string returned by the endpoint could potentially be long and cumbersome to read through for a human, but we can use Python's built-in **pprint** library to pretty print the results.

Python - Now with Pretty Print

```
# Import the pprint library
import pprint

# We'll instantiate an instance of PrettyPrinter, and store it in a variable named pp.
pp = pprint.PrettyPrinter(indent=4)

#Set the endpoint URL
url = "https://api.openweathermap.org/data/2.5/weather"

#Declare a variable for system function we are using
```

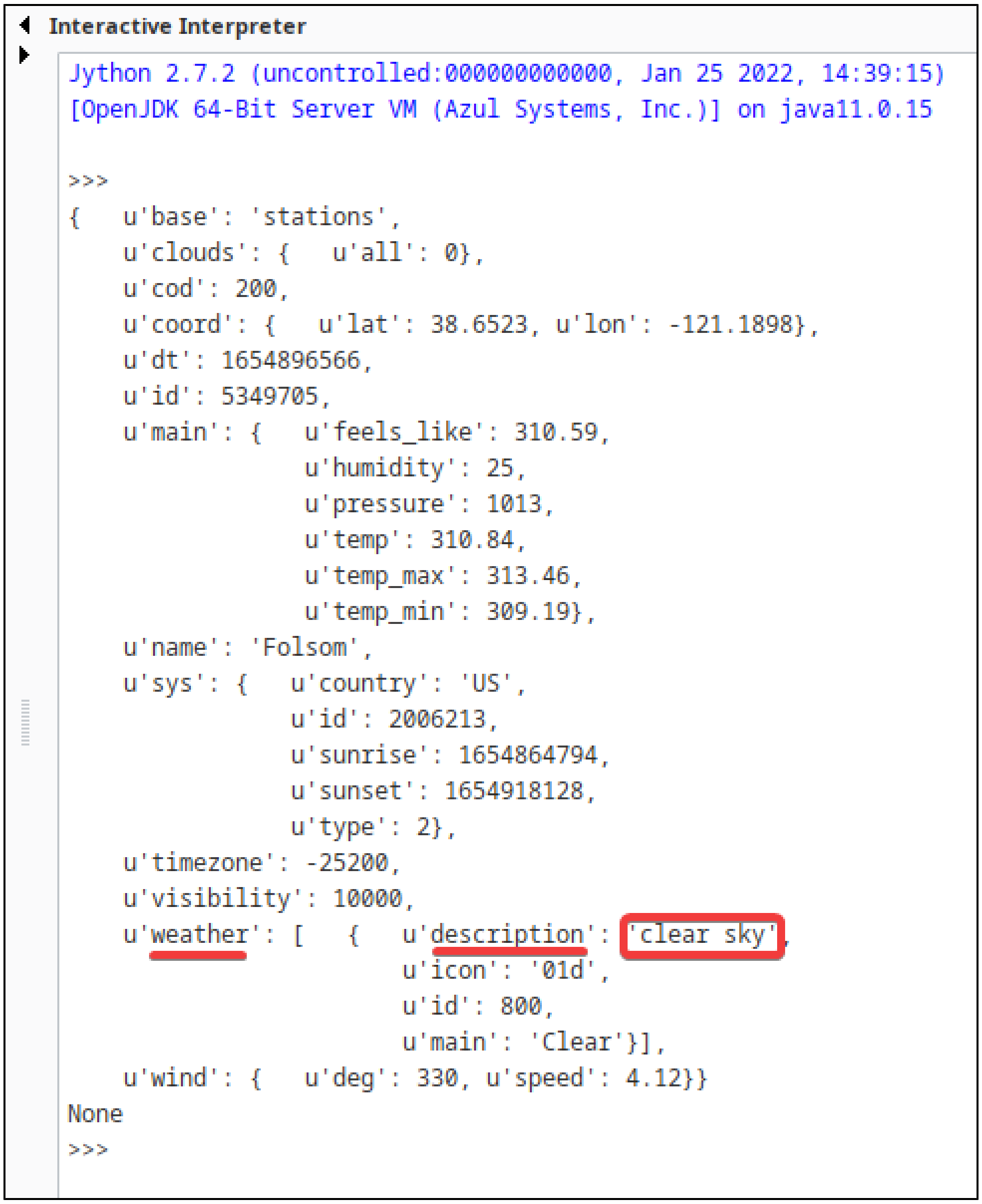
```
myClient = system.net.httpClient()

#Declare a variable and set the parameters for the endpoint URL
response = myClient.get(url, {"lat":38.652330, "lon":-121.189773, "appid":"Your API key"})

#Set a variable for the contents of the API call
results = response.getText()

# Convert the JSON string into a Python object. In this case, it results in a Dictionary.
decodedDict = system.util.jsonDecode(results)

# Print out the dictionary in an easy to read format.
print pp.pprint(decodedDict)
```

The resulting output, which is much easier to read, looks like the following:

Python - Results

```
{   u'base': 'stations',
    u'clouds': {   u'all': 0},
    u'cod': 200,
    u'coord': {   u'lat': 38.6523, u'lon': -121.1898},
    u'dt': 1654896566,
    u'id': 5349705,
    u'main': {   u'feels_like': 310.59,
                 u'humidity': 25,
                 u'pressure': 1013,
                 u'temp': 310.84,
                 u'temp_max': 313.46,
                 u'temp_min': 309.19},
    u'name': 'Folsom',
    u'sys': {   u'country': 'US',
                u'id': 2006213,
                u'sunrise': 1654864794,
                u'sunset': 1654918128,
                u'type': 2},
    u'timezone': -25200,
    u'visibility': 10000,
    u'weather': [   {   u'description': 'clear sky',
                        u'icon': '01d',
                        u'id': 800,
                        u'main': 'Clear'}],
    u'wind': {   u'deg': 330, u'speed': 4.12}}
None
```

From here we can see all of the keys that lead to our final value:



```
Jython 2.7.2 (uncontrolled:000000000000, Jan 25 2022, 14:39:15)
[OpenJDK 64-Bit Server VM (Azul Systems, Inc.)] on java11.0.15

>>>
{   u'base': 'stations',
    u'clouds': {   u'all': 0},
    u'cod': 200,
    u'coord': {   u'lat': 38.6523, u'lon': -121.1898},
    u'dt': 1654896566,
    u'id': 5349705,
    u'main': {   u'feels_like': 310.59,
                 u'humidity': 25,
                 u'pressure': 1013,
                 u'temp': 310.84,
                 u'temp_max': 313.46,
                 u'temp_min': 309.19},
    u'name': 'Folsom',
    u'sys': {   u'country': 'US',
                u'id': 2006213,
                u'sunrise': 1654864794,
                u'sunset': 1654918128,
                u'type': 2},
    u'timezone': -25200,
    u'visibility': 10000,
    u'weather': [   {   u'description': 'clear sky',
                        u'icon': '01d',
                        u'id': 800,
                        u'main': 'Clear'}],
    u'wind': {   u'deg': 330, u'speed': 4.12}}
None
>>>
```

To get to the value for the description key, we simply need to address each key along the way:

Python - To Get Value for the Sunset Key

```
# "[0]" is needed to specify the index of the list before specifying the "description" key.
print decodedDict.get("weather")[0].get("description")
```

Troubleshooting HTTP Methods

When making HTTP calls, it is helpful to be familiar with the status codes that are returned by errors. To demonstrate, we could modify an earlier example:

Python - Returns Error Status Codes

```
#Set the endpoint URL
url = "https://api.openweathermap.org/data/2.5/weather"

#Declare a variable for system function we are using
myClient = system.net.httpClient()

### Note that instead of replacing the "appid" parameter with a unique API key, we will leave it as the
```

```
"Your API key" placeholder.
#Declare a variable and set the parameters for the endpoint URL
response = myClient.get(url, {"lat":38.652330, "lon":-121.189773, "appid":"Your API key"})

#Print the output
print response.getText()
```

This will return an error, which looks like the following:

Python - IOError: Response Code 400

```
{"cod":401, "message": "Invalid API key. Please see http://openweathermap.org/faq#error401 for more
info."}
```

Note that the HTTP response code **401**, which means **unauthorized**, was referenced. This error code is correct because we intentionally used an incorrect API key!

HTTP Response Codes

The World Wide Web Consortium has a page dedicated to HTTP response codes, which details all possible error codes. However, several common codes are listed below:

Response Code	Description
400	**Bad Request** - The server could not understand the request due to malformed syntax.
401	**Unauthorized** - The request requires some sort of authentication. Potentially some user credentials or an auth token of some kind.
403	**Forbidden -** The server understood what you requested, but is intentionally refusing the request. In some cases, the error message may include a reason why the request was not fulfilled (but not always). Typically, if the server doesn't include a reason, they'll use a 404 error code instead
404	**Not Found** - Your syntax was correct, but the server could not find the resource you were asking for. This could mean a typo, or missing portion of the URL you are using. In this case, double check the address you're specifying. Depending on the configuration, this could also mean that the server does actually have the resource you requested, but doesn't want to confirm its existence (typically due to a security policy. See error code 403 above).

Related Topics ...

- system.net

SUDS - Library Overview

The SUDS Library

The SUDS library is a SOAP-based web services client developed for Python. It is extremely simple to use and practically eliminates the need for the user to understand or even view the WSDL of a web service.

> **Disclaimer**
>
> The SUDS library used to be included in the Python Standard Library. However it has since been removed, meaning you may not have access to it when performing a fresh install of Ignition. Additionally, development on the library has mostly ceased, so any copies you find online may be drastically outdated.
>
> The information on this page will be maintained for legacy users that need to be familiar with the old SUDS library. As a result, this page and its contents should be considered deprecated.

On this page

...

The SUDS library interprets the WSDL file for you and through a couple simple function calls allows you to get a list of the available methods provided to you by the web service. You can then invoke these methods in Ignition through event scripting to send and receive data in your projects. You will have to familiarize yourself with the SUDS library in order to make use of it.

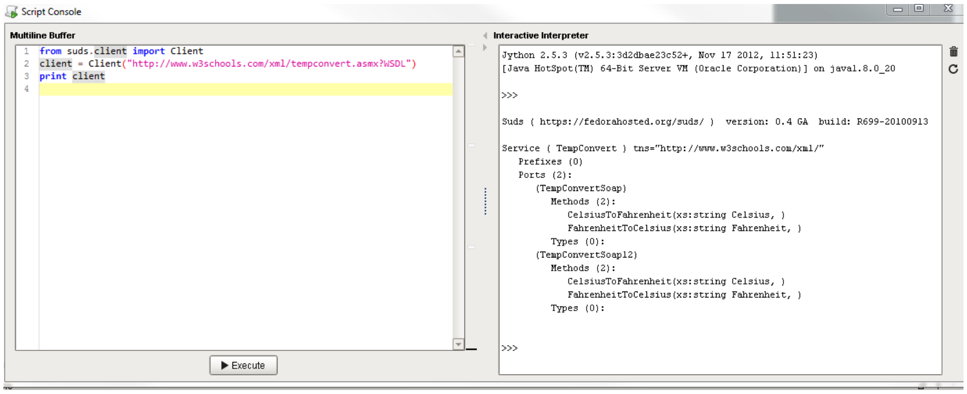
A Simple Example

If you read through the SUDS documentation, you'll see that the Client object is the primary interface for most users. It is extremely simple using this object and a few print statements to view a list of available methods provided by the web service you are connecting to. This example will illustrate how to make an initial connect to a web service, print out the list of available methods, and then call one of these methods and display the resulting value in a label on an Ignition window at the push of a button. The below example uses a public web service and is available for anyone to test against.

1. First, we can use the script playground to test out some scripting calls and see the output. The below example shows how to get a reference to a client object. By printing this client object to the console we get an output of all the available methods, types, and other information about the web service as defined in the WSDL file.

Python - W3Schools WSDL

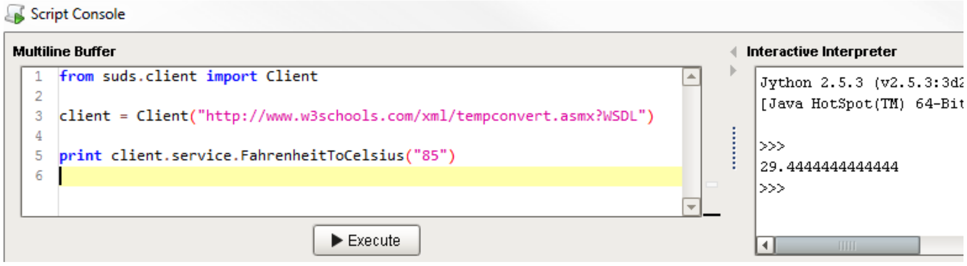
```
from suds.client import Client
client = Client("http://www.w3schools.com/xml/tempconvert.asmx?WSDL")
print client
```



```
from suds.client import Client
client = Client("http://www.w3schools.com/xml/tempconvert.asmx?WSDL")
print client
```

```
Jython 2.5.3 (v2.5.3:3d2dbae23c52+, Nov 17 2012, 11:51:23)
[Java HotSpot(TM) 64-Bit Server VM (Oracle Corporation)] on java1.8.0_20

>>>

Suds ( https://fedorahosted.org/suds/ )  version: 0.4 GA  build: R699-20100913

Service ( TempConvert ) tns="http://www.w3schools.com/xml/"
   Prefixes (0)
   Ports (2):
      (TempConvertSoap)
         Methods (2):
            CelsiusToFahrenheit(xs:string Celsius, )
            FahrenheitToCelsius(xs:string Fahrenheit, )
         Types (0):
      (TempConvertSoap12)
         Methods (2):
            CelsiusToFahrenheit(xs:string Celsius, )
            FahrenheitToCelsius(xs:string Fahrenheit, )
         Types (0):


>>>
```

This WSDL defines two functions: CelsiusToFahrenheit(string tempCelsius) and FahrenheitToCelsius(string tempFahrenheit). These are the functions that this web service makes available to you. Don't worry about the fact that the methods are listed twice. This is just because the WSDL has two definitions of the functions that are formatted for different SOAP version standards. To call these functions in Ignition scripting, you have to make use of the "service" member of the client object. You can see printing the returned results shows the conversion.



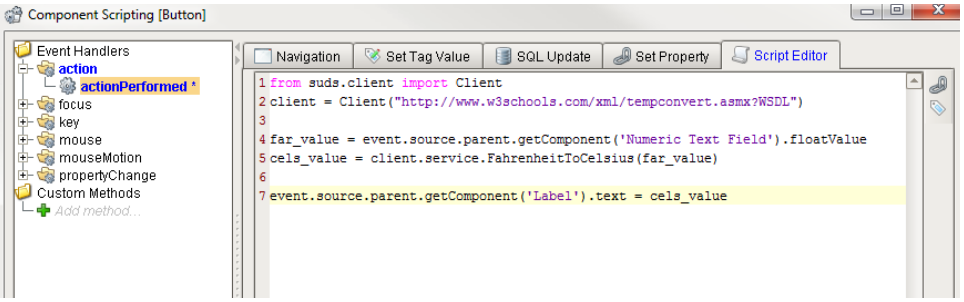
```
from suds.client import Client

client = Client("http://www.w3schools.com/xml/tempconvert.asmx?WSDL")

print client.service.FahrenheitToCelsius("85")
```

```
Jython 2.5.3 (v2.5.3:3d2
[Java HotSpot(TM) 64-Bit

>>>
29.4444444444444
>>>
```

2. To make a simple conversion window in an Ignition project you can add a button, a numeric textbox, and a label to a window. Then on the button to calculate a Fahrenheit to Celsius calculation, you would place something like the following:

Python - Fahrenheit to Celsius

```
from suds.client import Client
client = Client("http://www.w3schools.com/xml/tempconvert.asmx?WSDL")

far_value = event.source.parent.getComponent('Numeric Text Field').floatValue
cels_value = client.service.FahrenheitToCelsius(far_value)

event.source.parent.getComponent('Label').text = cels_value
```

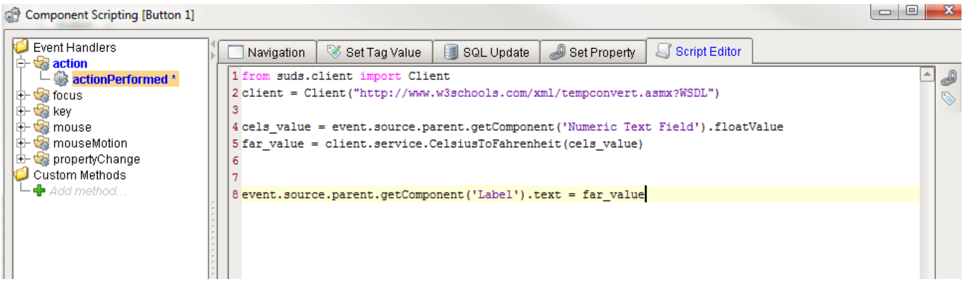


```
from suds.client import Client
client = Client("http://www.w3schools.com/xml/tempconvert.asmx?WSDL")

far_value = event.source.parent.getComponent('Numeric Text Field').floatValue
cels_value = client.service.FahrenheitToCelsius(far_value)

event.source.parent.getComponent('Label').text = cels_value
```

3. Then you can make a second button to do the opposite: calculate Celsius to Fahrenheit.

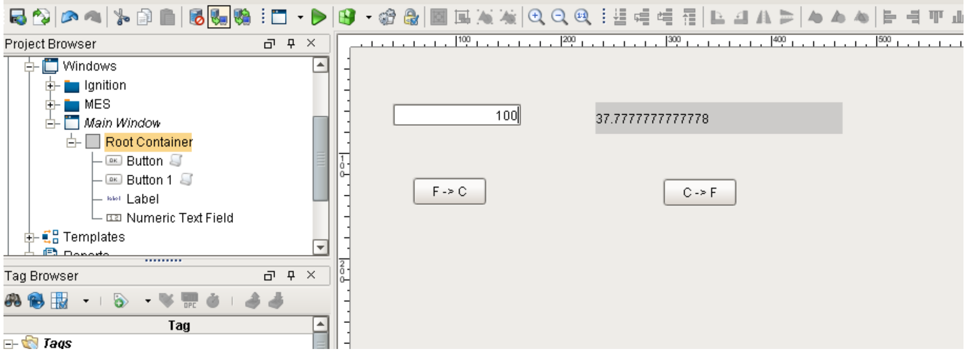
Python - Celsius to Fahrenheit

```
from suds.client import Client
client = Client("http://www.w3schools.com/xml/tempconvert.asmx?WSDL")

cels_value = event.source.parent.getComponent('Numeric Text Field').floatValue
far_value = client.service.CelsiusToFahrenheit(cels_value)

event.source.parent.getComponent('Label').text = far_value
```



```
from suds.client import Client
client = Client("http://www.w3schools.com/xml/tempconvert.asmx?WSDL")

cels_value = event.source.parent.getComponent('Numeric Text Field').floatValue
far_value = client.service.CelsiusToFahrenheit(cels_value)


event.source.parent.getComponent('Label').text = far_value
```

4. With your scripts in place your window should now function as a simple temperature conversion tool!



Beyond the Example

While the example is relatively simple, it can easily be expanded upon. However, always keep the general workflow in mind when using the SUDS library:

Pseudocode - WSDL Workflow

```
#Import the SUDS Client object
from suds.client import Client

#Instantiate a new Client Object
client = Client("url_to_your_wsdl")

#Call the desired method using the service instance variable
client.service.MyMethod(myArgument)
```

Complex Arguments

In the overview, the methods provided by the web service were very simple and took simple argument types. Sometimes however, the web service will describe complex types and allow you create instances of these types that can then be added to the system/machine that the web service is providing an interface for.

A simple, hypothetical example of this would be a system that stores contact information of clients and can be used as an address book of sorts by other systems on the network. It may provide not only a way to pull contact information for a certain individual out, but also a way to insert new contacts. We'll keep the example simple and say that contacts have only a name and a phone number.

⚠ This example is completely hypothetical. It is intended to give insight into complex types. It does not make use of an an actual functional web service.

For example, say we create and print the client object we associated with our web service in the following manner:

Pseudocode - Client Object

```
from suds.client import Client
url = 'http://localhost:7575/webservices/hypothetical_webservice?wsdl'
client = Client(url)
print client
```

And the resulting output is the following:

Python - Results

```
Suds ( https://fedorahosted.org/suds/ )  version: 0.4 GA  build: R699-20100913

Service (hypothetical_webservice)
   Prefixes (0):
   Ports (1):
     (Soap)
       Methods:
         addContact(Contact contact, )
         getContactList(xs:string str, xs:int length, )
         getContactByName(Name name, )
   Types (3):
     Contact
     Name
     Phone
```

Here you can see that, while not too complicated, the web service defines more than just methods that take simple type arguments and return the same. Under the Types section, you can see there are three "complex" types. These are basically just objects like one creates in an object oriented programming language like java. The SUDS Client object has an instance variable called "factory" that allows you to create these complex types so you can use them to invoke methods defined by your web service that take complex arguments.

If we wanted to add a contact using the addContact() method, we have to create a contact object first:

Pseudocode - Using a Method

```
contact = client.factory.create('Contact')
print contact
```

The create function creates a new contact object that knows its own structure. We can view this structure by calling print on this new object and see that it prints the following:

Python - Structure

```
(Contact)=
  {
    phone = []
    age = NONE
    name(Name) =
        {
            last = NONE
            first = NONE
        }
   }
```

By examining the Contact type object, we can see its structure and know what we need to create in order to have a valid Contact to add to the address book. We could then do the following to supply the necessary information for the Contact object and then call our addContact function.

Pseudocode - Adding a Contact

```
contact = client.factory.create('Contact')

phone= client.factory.create('Phone')
phone.areacode = '916'
phone.number = '5557777'

name = client.factory.create('Name')
name.first = 'John'
name.last = 'Doe'

contact.name = name
contact.phone = phone
contact.age = 30

client.service.addContact(contact)
```

After execution a new contact will have been added via the web service!

Steps to remember when using complex types:

Pseudocode - Complex Type Reminders

```
#Create a new type object using the factory instance variable of the Client object
my_type = client.factory.create('MyType')

#If you don't know the structure of the newly created object then print it to the console
print my_type
```

Related Topics ...

- Web Services, SUDS, and REST

JSON Format

About JSON

JavaScript Object Notation (JSON) is a language-independent data format. It's a lightweight format for storing and transporting data (i.e.,when data is sent from a server to a web page). JSON is easy for users to read and write and for machines to parse and generate.

On this page ...

JSON Rules

JSON has a few simple rules:

- Data is in name/value pairs.
- Data is separated by commas.
- Curly braces hold objects.
- Square brackets hold arrays.

How JSON Works

JSON Data - Name and a Value

JSON data is written as name/value pairs. A name/value pair consists of a field name in double quotes, followed by a colon, followed by a value.

```
{
        "companyName":"Inductive Automation"
}
```

JSON Objects

JSON objects are written inside curly braces.

> ⚠ The properties of a JSON object have no defined order. If you need a defined order, use a dataset or an array instead.

```
{
        "firstName":"Sally",
        "lastName":"Smith"
}
```

JSON Arrays

JSON Arrays are written inside square brackets. An array can contain objects. In the following example, the object "companies" is an array and contains three objects.

```
{
   "companies":[
      {
         "companyName":"Inductive Automation",
         "cityName":"Folsom",
         "stateName":"CA"
      },
      {
         "companyName":"Hewlett Packard",
         "cityName":"Palo Alto",
         "stateName":"CA"
      },
      {
         "companyName":"Apple",
         "cityName":"Cupertino",
         "stateName":"CA"
      }
   ]
}
```

Where JSON is Used in Ignition

Ignition uses the JSON format to store much of its data internally, including Tags and component properties.

Component Properties

Components have properties (props), which are simply named values. These properties are arranged in a tree structure following the structure and data model of the common JSON document format. Component properties are defined as a JSON structure, and are variable according to the type of component that the config object represents. All components registered in the module have a set of default properties included. These defaults are provided to the instantiated component at runtime, and so default props are not saved. Instead, only those which have a value that differs from the default are stored, serialized and sent to the client during loading.

Example - Sample Data from Table Component

```
[ { "city": "Helsinki", "country": "Finland", "population": 635591 }, { "city": "Jakarta", "country":
"Indonesia", "population": 10187595 }, { "city": "Madrid", "country": "Spain", "population": 3233527 },
{ "city": "Prague", "country": "Czech Republic", "population": 1241664 }, { "city": "San Diego",
"country": "United States", "population": 1406630 }, { "city": "Tunis", "country": "Tunisia",
"population": 1056247 } ]
```

Tags

Ignition exports and imports Tag configurations to and from JSON. Tags are defined as JSON objects, which consist of properties, arrays, and sub-objects. The system.tag.configure function can take either a String document definition, or a JSON object that defines one or more Tags. Overrides for UDTs are created by simple redefinition of properties, and complex structures like Event Scripts and Alarm configurations will be merged with inherited definitions.

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

As of version 8.0.8, you can copy the JSON or one or or more Tags in the Tag Browser. This copies them into the system clipboard. In addition, pasting the JSON into a different provider/designer will create or overwrite tags. For more information, see Understanding Tags.

Example 1 - Tag Export

```
{
  "name": "Tank Instance",
  "typeId": "Tank UDT",
  "tagType": "UdtInstance",
  "tags": [
    {
      "value": "80",
      "name": "Tank Level",
      "tagType": "AtomicTag"
    },
    {
      "value": 80,
      "name": "sliderValue",
      "tagType": "AtomicTag"
    }
  ]
}
```

Example 2 - Tag Export

```
{
  "tags": [
    {
      "valueSource": "memory",
      "dataType": "Boolean",
      "alarms": [
        {
          "setpointA": 1,
          "name": "Above Normal"
        }
      ],
      "name": "Boolean Tag",
      "value": false,
      "tagType": "AtomicTag"
    },
    {
      "valueSource": "memory",
      "dataType": "Boolean",
      "name": "One Shot Trigger",
      "tagGroup": "Driven One Shot",
      "value": true,
      "tagType": "AtomicTag",
      "enabled": true
    },
    {
      "valueSource": "opc",
      "opcItemPath": "ns\u003d1;s\u003d[Generic]_Meta:Random/RandomDouble1",
      "dataType": "Float8",
      "name": "Pressure3",
      "tagGroup": "Driven One Shot",
      "tagType": "AtomicTag",
      "enabled": true,
      "opcServer": "Ignition OPC UA Server"
    },
    {
      "valueSource": "opc",
      "opcItemPath": "ns\u003d1;s\u003d[Generic]_Meta:Random/RandomDouble2",
      "dataType": "Float8",
      "name": "Thickness3",
      "tagGroup": "Driven One Shot",
      "tagType": "AtomicTag",
      "opcServer": "Ignition OPC UA Server"
    }
  ]
}
```

The following feature is new in Ignition version **8.0.10**
Click here to check out the other new features

JSON in Tag UDTs

You can also set JSON strings as properties in an Ignition Tag. Any properties of a Tag can be set to a string that represents a JSON object.

In a UDT, the { } braces are used to denote a property reference. Make sure you don't have any properties with names that look like JSON objects so there is no overlap. Strings will appear as black text, parameter references will appear as grey and italicized.

Looping through JSON Objects with Scripting

Traversing a JSON object in scripting is simple as long as the structure is known. If there are objects within objects, you can use multiple loops to get through it all.

Let's use the JSON Array object above for a simple example. You can loop through the list to get the repeating items by name.

```
# loop through the JSON data
# fetch the data, this will change depending on where the script is in relation to the table
json = self.items
# access the companies object (which is a list)
companies = json["companies"]

# loop through the companies list
for company in companies :
        # get each item out of the row object
        name = company["companyName"]
        city = company["cityName"]
        state= company["stateName"]
        # now do something with the data
```

Related Topics ...

- JSON Functions

Basic Python Troubleshooting

When learning how to code in Python, most Ignition users tend to place most of their learning efforts on memorizing syntax or other aspects of the language. While being comfortable with the language is useful, there are plenty of references available: countless books and websites that describe syntax and usage already exist.

In truth, the best thing you can do to make yourself a better programmer is to learn some basic troubleshooting behaviors. While syntax examples are all over the Internet, examples detailing exactly what you want your code to do, to the extent you want it to, will be difficult if not impossible to find.

This section details how to troubleshoot a script in Ignition. You won't walk away from this section having exact answers to specific problems, but rather examples and concepts that you can apply to your own scripts.

On this page ...

Coding Best Practices

The following are some general and helpful best practices that can help minimize the amount of time you spend troubleshooting, as they can help better direct you to a problem.

Look for Errors

When code fails mid-execution, it always generates an error message. Where the message appears depends on where the script executed:

Scope	Print Command	Print Output Location
Gateway *Note: anything that isn't in the Client or Session scopes listed below uses this scope*	system.util.getLogger	Scripts on Gateway-scoped resources (Tags, Alarm Pipelines, SFCs, etc.,) will appear on the Logs page of the Status section on the Gateway. Additionally, the wrapper.log file in Ignition's installation directory will have these messages. Here are the default wrapper.log file paths for each operating system: - **Windows:** Program Files/Inductive Automation/Ignition/logs - **Lunix:** /var/log/ignition - **Mac OS X:** /Users/UserName/Documents/Ignition-osx-x.x.x/logs
Vision	Python's `print` command, or system.util.getLogger	The Client Console will contain any errors generated in the client: press **Ctrl + Shift + F7** to open the console, or using the menubar in the client to go to **Help > Diagnostics** then click on the **Console Tab**. Additionally, a red Error Box should appear with details on the error if a Component Event Handler threw the exception: Extension Functions do not generate the Error Box. If you don't see the error box then it might be minimized, or open in the background (behind the Client).
Perspective	system.perspective.print	Web browsers generally offer a way to inspect a page, which usually contains a console of some sort.
Designer	Depends. See the output location.	The Designer reports errors in a similar manner to Clients: errors appear in the Designer's Console (**Ctrl + Shift + c**), Component Event Handlers will generate a red Error Message. Note that events some events, such as Client Startup Scripts, will not trigger in the Designer, so get in the habit of launching a Client when testing non-component scripting events. While in the Designer, and working on Perspective resources, calls to system.perspective.print will appear in the Designer's Console as well.

What happens if I don't see any errors?

If there truly isn't an error message somewhere, but your code isn't doing what you expected, then ask yourself the following questions:

1. What is your script supposed to do?
2. What is it actually doing?

Question #2 is harder to answer: if you knew what it was doing, you wouldn't be stuck! The best way to answer this question is by adding **`print`** statements to your code.

When a script doesn't perform to expectations, it can suggest a problem with the script's workflow. Some of the pages in this section can offer some suggestions on what to do.

If You Find an Error, Read It!

It is common for users that are new to scripting to see an error message, immediately close it without much thought, and then stare at their code as if the problem will politely make it self known. While looking over your code line-by-line will eventually lead you to the problem, the error messages can provide you with a shortcut to the issue. Check out some of the pages in this section for more information on reading an error message.

Use Print Statements

When testing your scripts, the print command can help you verify that your code is behaving the way it should. This allows you to reconstruct what your code did when it executed. Don't be afraid to add helpful print statements to your code.

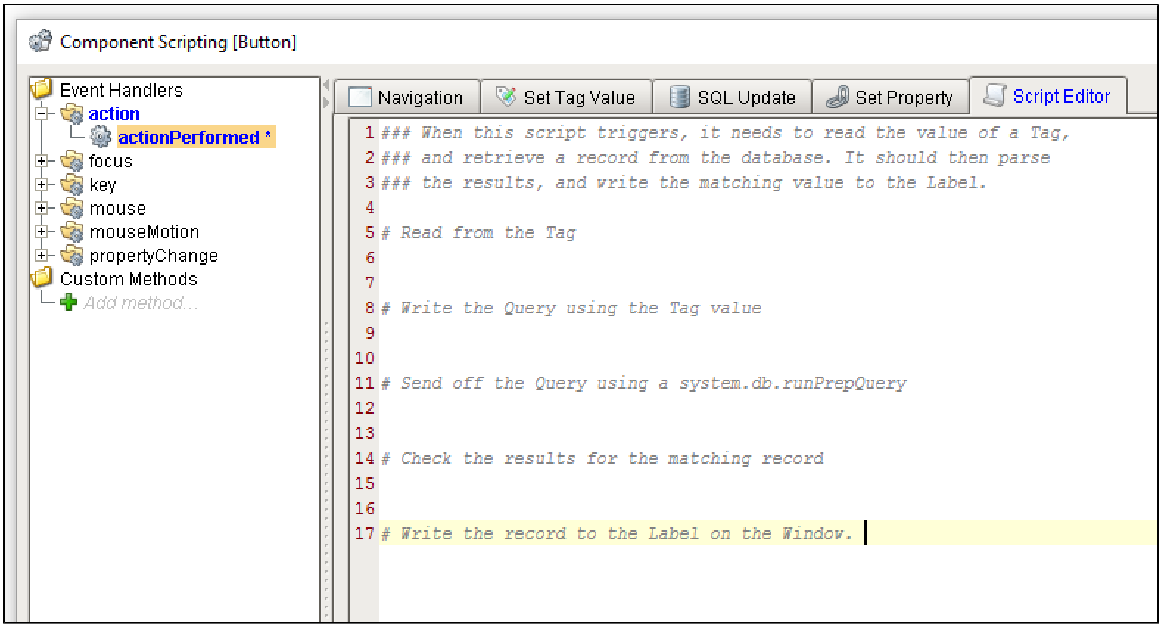
Once your code works as expected, remember to either remove or comment out the print statements, so they don't flood the console during normal use as this makes troubleshooting other issue more difficult. Just be mindful of the scope of the script, as that determines how you generate print messages, as well as where the console is.

Python - Using print to Troubleshoot

```
myVar = system.tag.read("folder/tag")

# printing out variables you are going to use in an if-statement later allows you to confirm that the values are what you expect them to be.
print "myVar is set to: " + str(myVar)

# Sometimes viewing the data type of the variable can prove helpful.
print "myVar is a: " + type(myVar)

# Should your code have multiple if-statements, adding a print statement before and after can show you where the flow of the script went.
print "Starting if-statement"

if myVar > 100:

        # If you don't see this print statement, then your if-statement evaluated to False.
        print "Inside if-statement"
        doWork()

# Printing the end of your script doesn't give you any useful troubleshooting information, but if you need to trigger the script multiple times, it helps delineate each execution.
print "Script Ended"
```

Add Comments

While comments are useful to remind you how your code works, you can also use them to plan your script before you write any code. Break down what you want the code to do into several smaller steps, and then leave comments describing those steps in order. This provides you a chance to review the script's workflow before worrying about syntax. It also provides natural points to stop and test your code, to make sure it is doing what you think it should do.



```
### When this script triggers, it needs to read the value of a Tag,
### and retrieve a record from the database. It should then parse
### the results, and write the matching value to the Label.

# Read from the Tag


# Write the Query using the Tag value


# Send off the Query using a system.db.runPrepQuery


# Check the results for the matching record


# Write the record to the Label on the Window.
```

Test Early, Test Often

Unless the script is very simple, avoid writing the entire script and then testing at the end. You may have missed an important line early on, so now you have to adjust all of the code below that line to make the script run. As mentioned above, add print statements, and run your script to make sure it is doing what you think it should.

Additionally, stopping to test your code provides an excellent opportunity to **save your project**. Get into the habit of saving before you execute any new code.

Avoid Hard-Coding Arguments: Use Variables Instead

Instead of doing this:

Python - Hard Coded Message Box

```
system.gui.messageBox("Hello you. Glad to see you")
```

Try to get into the habit of doing this:

Python - Message Box Variable

```
message = "Hello you. Glad to see you"
system.gui.messageBox(message)
```

Simple examples like the above don't make for the best use case, but when you have a large script that references a value multiple times, it is easier to declare the value once in a variable, and then just reference the variable throughout your code. If you need to change the value later, then you can simply change it once where you initialize the variable, and you don't have to search every line of your code looking for the value.

Decide on a Naming Convention

When creating variables, try to adopt a naming convention that comes natural to you, and stick to it. If you consistently write variables using the same conventions, you will be less likely to end up with a typo when referencing that variable later in your code. Remember that code is case-sensitive, so something as simple as forgetting to capitalize a letter will cause an error.

This is especially important if you are working in a group on the same project. It's better to get together with your colleges and agree upon some naming conventions before you write any code.

Ignition's **system.* functions** use the camelCase naming convention. That is, the first letter of each word is capitalized except for the very first letter. We recommend that you use it for variable names because It is easy to remember to use camelCase for both functions and variable names instead of for just one.

The Simplest Approach Really Is the Best Approach

When learning how to code, it's not uncommon to run into multiple issues that require you to find a workaround. However, these workarounds can cascade into other issues and make your script more complicated. Consider the following:

> "The goal of my script is to access A and then output B...
> Shoot, B requires C, so I'll add C...
> Wait, C requires that D exists, so I'll create that...
> Oh, D needs interfaces E, F, and G, so let's add those in...
> Hmm. E needs H and I, F needs J and K, while G needs Y, Z and...A **again**?!?"

Take a step back and ask yourself "what is this script doing"? If you can't do that in a sentence or two, you may want to rethink the script. If the scope of the script is too large, then it considerably increases the complexity of the code, which in turn could add a plethora of problems later.

If you keep adding workarounds, but you're code is not getting any closer to achieving its end goal, there may be an easier way to accomplish what you're trying to do with a different approach.

Related Topics ...

- Python Scripting

In This Section ...

Reading Error Messages

When an error occurs in the execution of a script, an Error Message Box will pop up. The popup box appears in front of any open Designer windows, and will remain in view until you close it or click on something behind it.

The following feature is new in Ignition version **8.0.6**
Click here to check out the other new features

There are two modes for the Error box selectable by the Send to Front and Send to Back icons. Send to Front means additional error messages will cause the popup to reappear on top. Send to Back will cause the errors to remain hidden below the Designer.

On this page

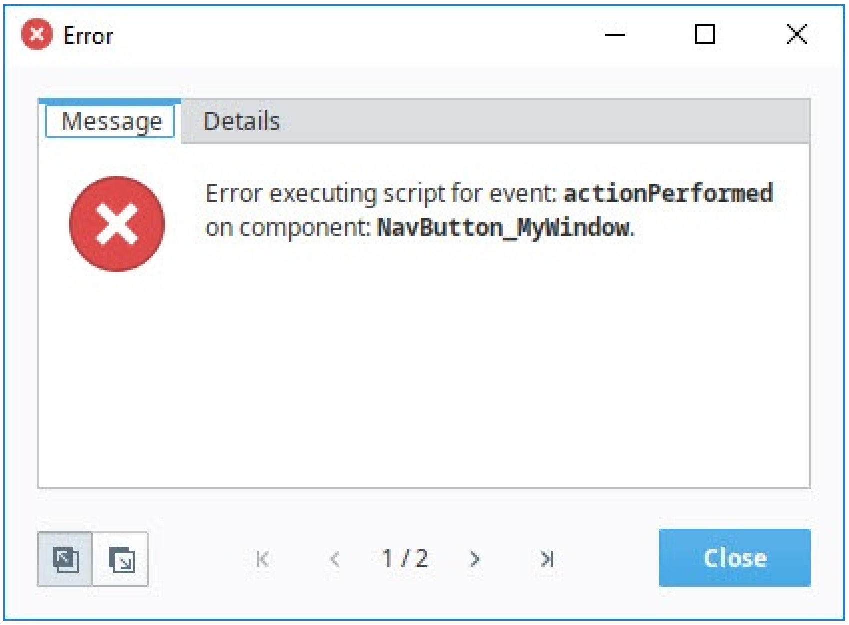
...

Error Message Box Overview

Exceptions usually include a line number. Take note of the number in the Details tab, and start your search for the problem there. Be aware, however, that the line reported may not be the cause of the issue. The actual problem may be higher up in the code due to a faulty initialization, or some other issue. When troubleshooting, always start looking at the line reported, and work your way back towards the top.

When testing a script, you may eventually come across an Error Message Box like the one shown below.

Your first inclination will be to close the error without closer examination: resist it!

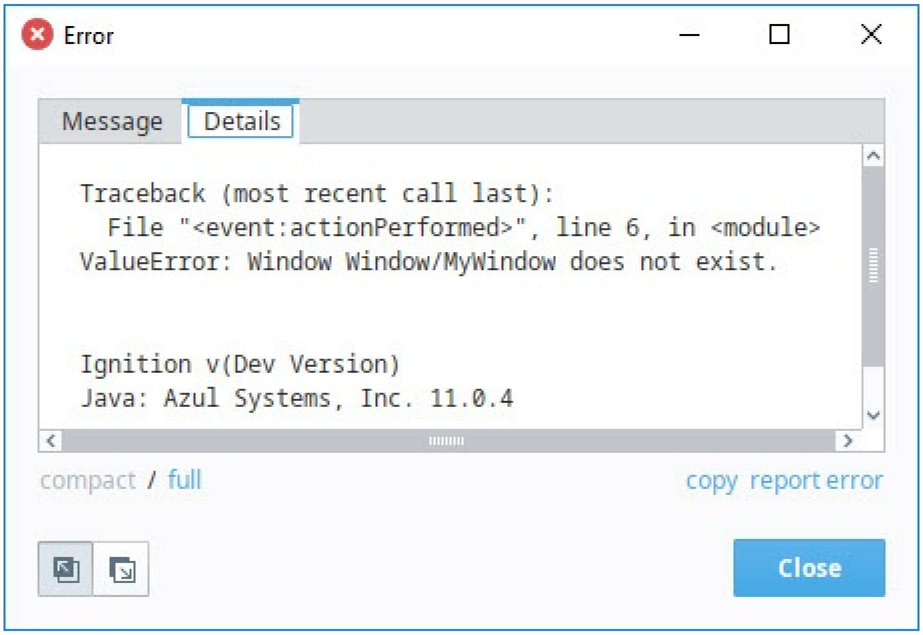


Error executing script for event: **actionPerformed** on component: **NavButton_MyWindow**.

The error messages generated from failed script executions are incredibly helpful. The **Message** tab on the error describes the Event Handler that encountered the exception; in this case, the actionPerformed event handler. This is important in cases where multiple Event Handlers on the same component have scripts. Without knowing which Event Handler generated the script, we could waste time trying to troubleshoot the wrong script.

Additionally, it shows the name of the component, which is "**NavButton_MyWindow**". Again, this is helpful if multiple scripts from multiple components are triggering in quick succession. The **Message tab** will clearly point you towards the source of the exception. This is another reason to give meaningful names to your components: doing so makes the process of tracking an error much easier.

The **Details** tab has even more information, specifically the **line number** that the exception occurred on, as well as the **error message**. The message here states that the window at path "MyWindow" does not exist, so we can check the Project Browser to see if we simply mistyped the name of the window.



```
Traceback (most recent call last):
  File "<event:actionPerformed>", line 6, in <module>
ValueError: Window Window/MyWindow does not exist.


Ignition v(Dev Version)
Java: Azul Systems, Inc. 11.0.4
```

Troubleshooting Errors Using the Error Message Box

When we don't get our expected results from our script, as in the following examples, always read both the Message and Details tabs in the Error Message Box. They are pretty good about pointing you to the root cause of your error. From there, you can easily find and fix any errors in your script.

The following examples show how to troubleshoot some of these error messages from the Error Message Box. Keep in mind that every script is unique, but at least you'll become familiar with what some of the error messages mean, and gain a little insight of the troubleshooting process.

Broken Example - Incorrect Attribute

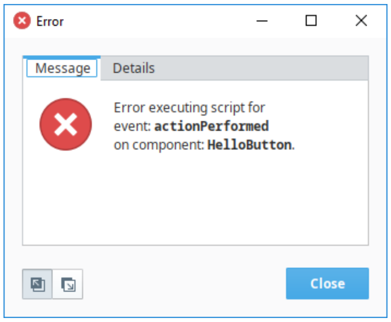
Here is a script that when a Button is pressed, it's supposed to open a Message Box on a window and display "Hello World".

Python - Broken: Incorrect Attribute

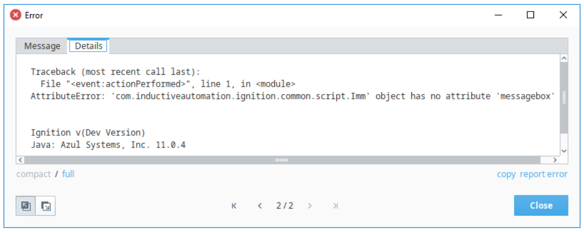
```
system.gui.messagebox("Hello World")
```

From the Error Message Box, you know two important things right away: the script is using the actionPerformed event on the Hello Button, and line 1 displays the description of the error. Check to be sure you used the correct attribute. Then check the spelling, case sensitive letters, spacing, and operators in your script. If you are familiar with Ignition's Built-in Scripting Functions, you can probably spot the error immediately. If not, you might want to use the autocompletion popup feature to retype your 'messageBox' scripting function. This will automatically fix any syntax errors.

The error in this example is in the attribute name. It uses case sensitive letters (i.e., **messageBox**).



Error executing script for event: **actionPerformed** on component: **HelloButton**.



```
Traceback (most recent call last):
  File "<event:actionPerformed>", line 1, in <module>
AttributeError: 'com.inductiveautomation.ignition.common.script.Imm' object has no attribute 'messagebox'


Ignition v(Dev Version)
Java: Azul Systems, Inc. 11.0.4
```

In this case, "messagebox" should be spelled with a capital "B" since we're trying to use Ignition's system.gui.messageBox function. We corrected the script by changing our code to the following:

Python - Corrected: Incorrect Attribute

```
system.gui.messageBox("Hello World")
```

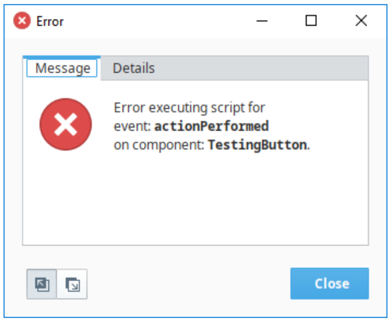
Broken Example - Undefined

Here is a similar script that when a Button is pressed, it's supposed to open a Message Box on a window and display "Testing".

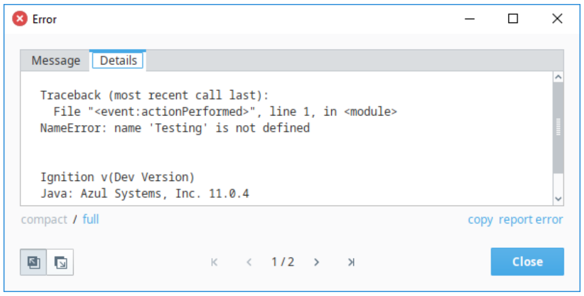
Python - Broken: Undefined

```
system.gui.messageBox(Testing)
```

The Error Message Box displays an error on the actionPerformed script on the button component in line 1 with an **undefined** name. The Details tab is simply telling us that something is not defined in the script. You can either define a variable or create a string by putting "Testing" in quotes to correct the error.



Error executing script for event: **actionPerformed** on component: **TestingButton**.



```
Traceback (most recent call last):
  File "<event:actionPerformed>", line 1, in <module>
NameError: name 'Testing' is not defined


Ignition v(Dev Version)
Java: Azul Systems, Inc. 11.0.4
```

For this example, we corrected the code by turning the argument passed to system.gui.messageBox into a string literal with quotation marks:

Python - Corrected: Undefined

```
system.gui.messageBox("Testing")
```

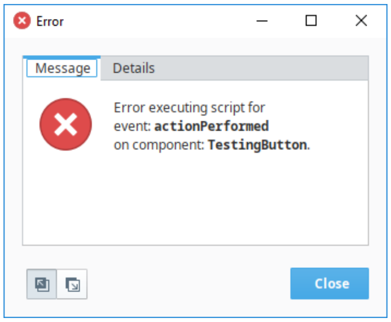
Broken Example - Type Error

Similar to the other scripts, this script opens a Message Box and is supposed to display "100" when a Button is pressed.

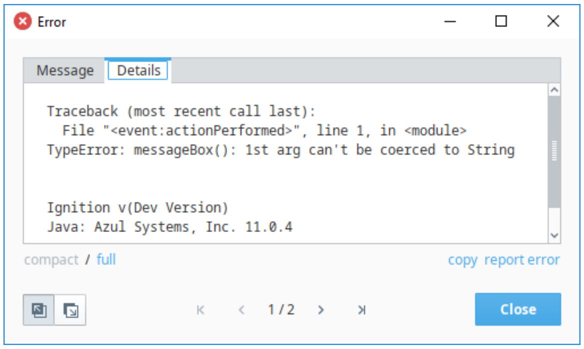
Python - Broken: TypeError

```
system.gui.messageBox(100)
```

Since this script failed to execute successfully, the Error Message Box popped up and displayed an error on the actionPerformed script on the button component in line 1 with a **TypeError**. This error tells us that the 1st parameter is expecting a string, not an integer.



Error executing script for event: **actionPerformed** on component: **TestingButton**.



```
Traceback (most recent call last):
  File "<event:actionPerformed>", line 1, in <module>
TypeError: messageBox(): 1st arg can't be coerced to String


Ignition v(Dev Version)
Java: Azul Systems, Inc. 11.0.4
```

For this example, we corrected the code by changing the argument passed to system.gui.messageBox into a string literal with quotation marks.

Python - Corrected: TypeError

```
system.gui.messageBox("100")
```

You can see from the examples above, that the Error Message Box provides quite a bit of information that points to the root cause of an error on an Event Handler script. As you learn from the Error Message Boxes about what these messages mean, you'll be able to quickly spot them in your script and quickly correct any errors you encounter.

Related Topics ...

- Basic Python Troubleshooting
- Scripting in Ignition
- Getting Started with Scripting in Ignition

Troubleshooting - Nothing Happened

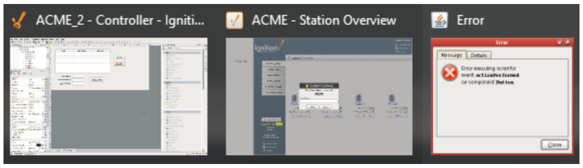
Did It Work?

When testing a script, you may find yourself in a situation where the script appears to be running, but doesn't seem to work. Furthermore, you may not see any error messages stating there is a problem.

When a script doesn't perform to expectations and doesn't throw an exception, it can suggest a problem with the script's workflow. This page describes a couple of common scenarios.

Before You Begin...

Make **absolutely** certain that there isn't an Error Message Box hiding somewhere, otherwise you'll waste time applying the troubleshooting tips outlined below, when the real the error message was hiding in the background behind another window the whole time.



Error executing script for event: actionPerformed on component: Button.

If a button component on a window is running a script, the output will be displayed on the Designer /Client Output Console. You can check for any error messages by navigating to the Console. From the **Client** menu bar, go to **Help > Diagnostics** and select the **Console tab**. From the **Designer** menu bar, select **Tools > Console**.

Common Scenarios When Nothing Happens

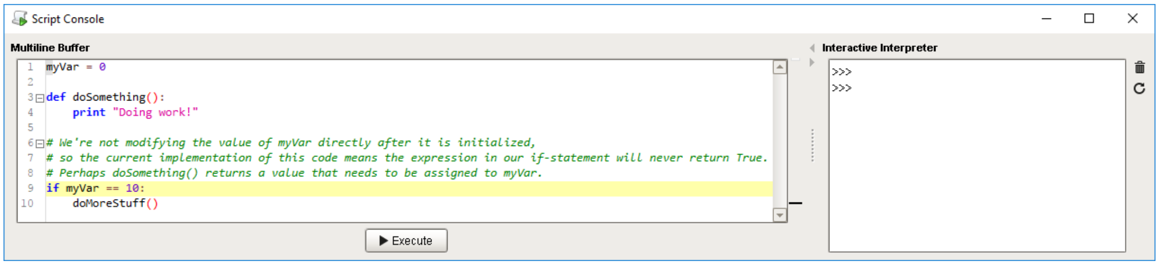
An Important Line Is Missing

Some critical part of your code is missing or commented out. It could be something simple, like your code is supposed to increment a value, but you forgot to write the line that increments the value.

Pseudocode - Missing Code

```
myVar = 0

def doSomething():
        print "Doing work!"

# We're not modifying the value of myVar directly after it is initialized,
# so the current implementation of this code means the expression in our if-statement will never return
True.
# Perhaps doSomething() returns a value that needs to be assigned to myVar.
if myVar == 10:
        doMoreStuff()
```

A quick way to test your code is by running your script in the Script Console before attaching it to a scripting event or specific component. You can see that running the code above in the Script Console that nothing happened, thus, suggesting an error.



```
myVar = 0

def doSomething():
    print "Doing work!"

# We're not modifying the value of myVar directly after it is initialized,
# so the current implementation of this code means the expression in our if-statement will never return True.
# Perhaps doSomething() returns a value that needs to be assigned to myVar.
if myVar == 10:
    doMoreStuff()
```

```
>>>
>>>
```

The Script Is Not Being Called

The script doesn't appear to be working because the mechanism that is supposed to be triggered hasn't been called. This can be caused by using the wrong event: (i.e., perhaps you placed the code on a Button's **propertyChange** event, when you meant to place it on **actionPerformed**).

Alternatively, perhaps you defined a function in your script, but you forgot to call the function.

Additionally, if you're testing the script in the Designer, make sure it is in **Preview Mode**. Event based scripts will not run in the Designer unless it is in Preview Mode.

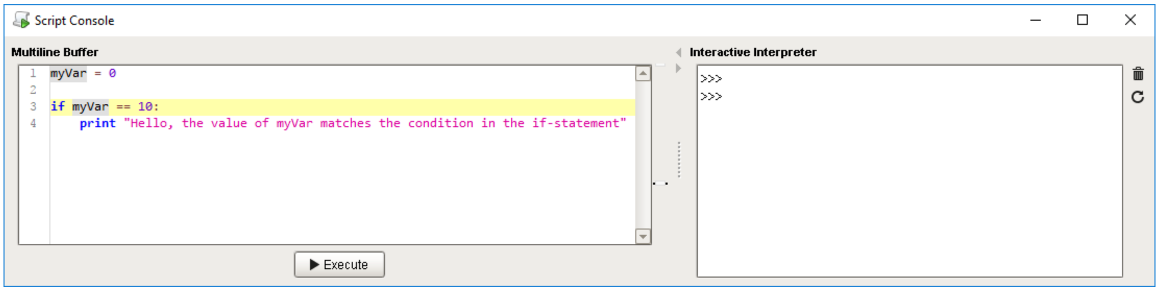
Important Lines Are Being Skipped

This can be caused by incorrect indentation, or a misconfigured condition in an if-statement's expression. For example, you used "**==**" when you meant to use "**!=**". Verify that you are using the correct operators in if-statements.

Pseudocode - Skipping Code

```
myVar = 0

if myVar == 10:
        print "Hello, the value of myVar matches the condition in the if-statement"
```

Test your code for any errors by running your script in the Script Console. You can see nothing happened by running the code above in the Script Editor, once again suggesting an error.



```
myVar = 0

if myVar == 10:
    print "Hello, the value of myVar matches the condition in the if-statement"
```

```
>>>
>>>
```

Determining the Cause

The easier step to take when troubleshooting your script is to start adding print statements to your code. From here, you can start piecing together what the code is doing:

- **Print** the value of variables to make sure they are coming in the way you expect.
- Place **print** statements at the start and end of your code. This allows you to determine when your script is being called, and when it finishes.
- Adding a **print** statement before and after an if-statement can show you the flow of your script went as expected, or was filtered out by the if-statement.

Related Topics ...

- Script Console

- Basic Python Troubleshooting
- Reading Error Messages

Troubleshooting Workflow

Troubleshooting a script is an iterative process. Since the script stops executing at the first error, we won't see if there are any other errors on later lines unless, of course, we spot them ourselves while writing the code. As a result, we will have to keep trying our script until it executes successfully.

It is important to understand that scripts always execute from top to bottom, and each line must complete before the next line may move on. In the event, our code returns an exception, we can assume that our troubleshooting process should always start at the line reported in the error, and then work up until we find the problem.

On this page, we will take an indepth look at a script with multiple problems, and work through each one. Note, that this is not a comprehensive list of all possible types of exceptions that could occur when writing a script, but instead, this page attempts to demonstrate the troubleshooting process.

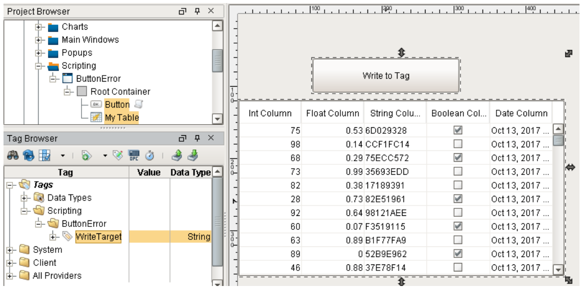
To learn more about troubleshooting specific script errors, refer to the following sections:

- Reading Error Messages
- Troubleshooting - Nothing Happened

On this page ...

Scenario Overview

Here we have a window with two components: a **Button** and **Power Table**. The purpose of the button is to find which cell in the Power Table the user selected, and write the value to a Tag.



Int Column	Float Column	String Colu...	Boolean Col...	Date Column
75	0.53	6D029328	☑	Oct 13, 2017 ...
98	0.14	CCF1FC14	□	Oct 13, 2017 ...
68	0.29	75ECC572	☑	Oct 13, 2017 ...
73	0.99	35693EDD	□	Oct 13, 2017 ...
82	0.38	17189391	□	Oct 13, 2017 ...
28	0.73	82E51961	☑	Oct 13, 2017 ...
92	0.64	98121AEE	□	Oct 13, 2017 ...
60	0.07	F3519115	☑	Oct 13, 2017 ...
63	0.89	B1F77FA9	□	Oct 13, 2017 ...
89	0	52B9E962	☑	Oct 13, 2017 ...
46	0.88	37E78F14	□	Oct 13, 2017 ...

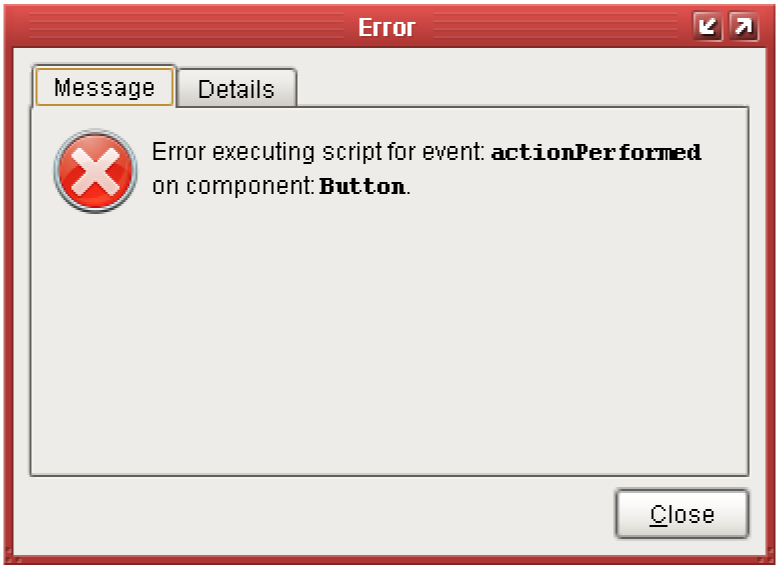
The code on the button is listed below.

Python - Sample Code Block

```python
# Create a variable that references the Power Table
table = event.source.parent.getComponent('Power Table')

# Find the cell the user currently has selected, and store the value
in a variable
userSelectedValue = table.data.getValueAt(table.selectedrow, table.
selectedColumn)

# Write the User Selected Value to a tag
system.tag.write("Scripting/ButtonError/WriteTarget",
userSelectedValue)
```

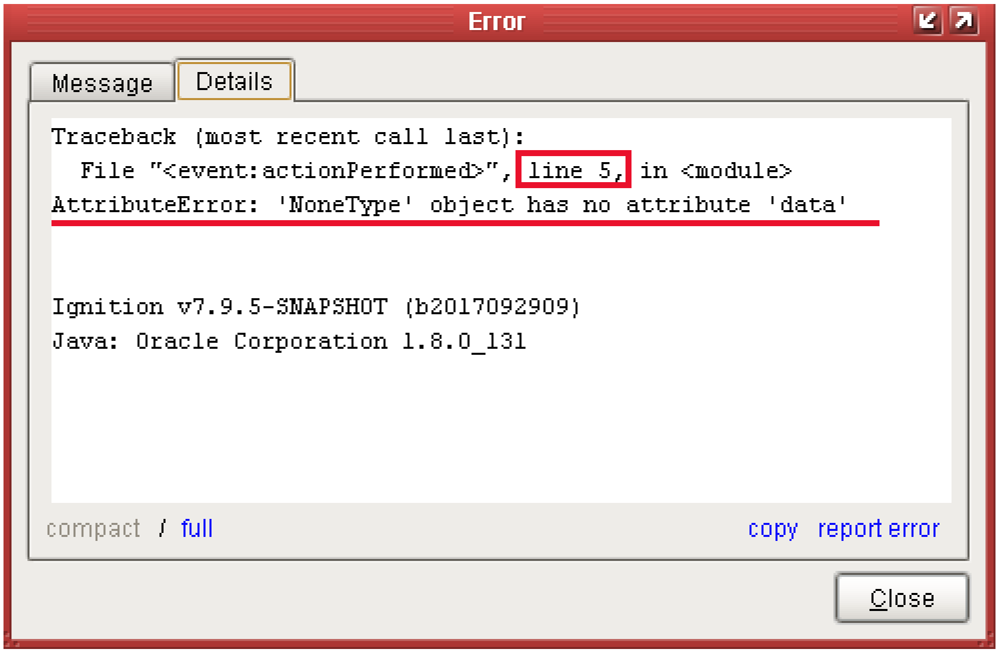
First Error - NoneType Object

When the button is pressed, we are presented with an error. The **Message** tab in the Error Box describes the Event Handler that encountered the exception.



Error executing script for event: **actionPerformed** on component: **Button**.

The **Details** tab tells us where to start looking, specifically the line number where the exception occurred as well as the error message. Line 5 is referenced, so the troubleshooting process should start there.



```
Traceback (most recent call last):
  File "<event:actionPerformed>", line 5, in <module>
AttributeError: 'NoneType' object has no attribute 'data'


Ignition v7.9.5-SNAPSHOT (b2017092909)
Java: Oracle Corporation 1.8.0_131
```

Exception Error Explained

When an error message refers to a **NoneType** object, that simply means a null, or None in Python. The word '**attribute**' is used to reference a property on an object. In this case, this means the script was trying to access the **'data'** property on nothing. The message is telling us that it couldn't find a property named **'data'** on a nothing, which is correct since NoneType objects don't have any properties named 'data'.

If we were to reword this message to something a bit more straightforward, it would say the following: "I tried to access the 'data' property on 'null', but it doesn't have a property by that name."

What Should We Look For?

If your exception is referring to a NoneType object, then some line of code probably tried to reference something else (i.e., Tag value, property, another variable, etc.), but couldn't find anything at the location you specified. It is also helpful to note what attribute was mentioned in the error, which was **data**. If we look at our code, we see the following on line 5.

Python - Line 5

```
userSelectedValue  = table.data.getValueAt(table.selectedrow, table.selectedColumn)
```

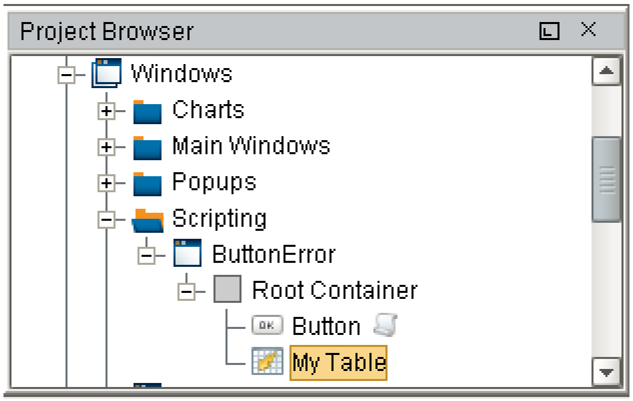
Based on this information, we should take a closer look at the table variable since our code is specifying **'data'** as an attribute of the table. The error states that it could not find the data attribute on a NoneType object, but our code only references data in regards to the table object, so maybe there is something wrong with how that table object was referenced or initialized. If we look further up in our code, we see that `table` was initialized on line 2:

Python - Line 2

```
table = event.source.parent.getComponent('Power Table')
```

Since this script was placed on the Button component's **actionPerformed** event, we can trace back to the source of the issue:

event = actionPerformed event
source = Button this script is placed on
parent = Container that the Button was placed in. Based on the Button's position in the window, this appears to be the Root Container.
getComponent('**Power Table**') = Should return a reference to a component named '**Power Table**' directly inside of the Root Container, assuming one exists. If we take a look at the Project Browser, we see the following:



There is no component named '**Power Table**' in the Root Container. However, there is a Power Table component named '**My Table**', so it appears someone renamed the component, which caused our script to initialize the variable **`table`** as a '**NoneType**' instead of a reference to a component.

Solution

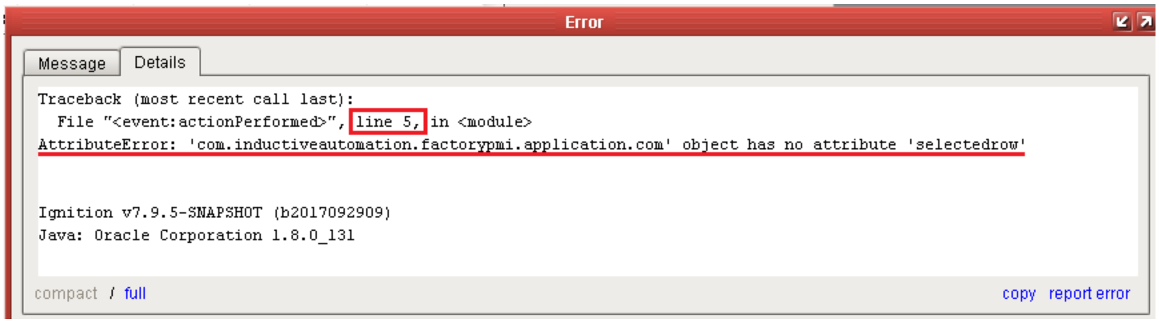
To fix this issue, we can simply update our code to use the new name of the Power Table, which is '**My Table**'. We can type this in manually, but since it is case sensitive, it must match exactly. It may be easier to find the **Name** property on the component, copy the name to the system clipboard (**Ctrl-C**), and paste it (**Ctrl-V**) into the script. Our code now looks like the following:

Python - Updated Code Block

```
# Create a variable that references the Power Table
table = event.source.parent.getComponent('My Table')

# Find the cell the user currently has selected, and store the value in a variable
userSelectedValue = table.data.getValueAt(table.selectedrow, table.selectedColumn)

# Write the User Selected Value to a tag
system.tag.write("Scripting/ButtonError/WriteTarget", userSelectedValue)
```

Second Error - Checking Attributes

If we try our button again, we get the following exception error.



```
Traceback (most recent call last):
  File "<event:actionPerformed>", line 5, in <module>
AttributeError: 'com.inductiveautomation.factorypmi.application.com' object has no attribute 'selectedrow'


Ignition v7.9.5-SNAPSHOT (b2017092909)
Java: Oracle Corporation 1.8.0_131
```

Exception Error Explained

Like the last exception, the problem has to do with our code trying to access an attribute (property) on an object, but the given attribute doesn't exist on the object.

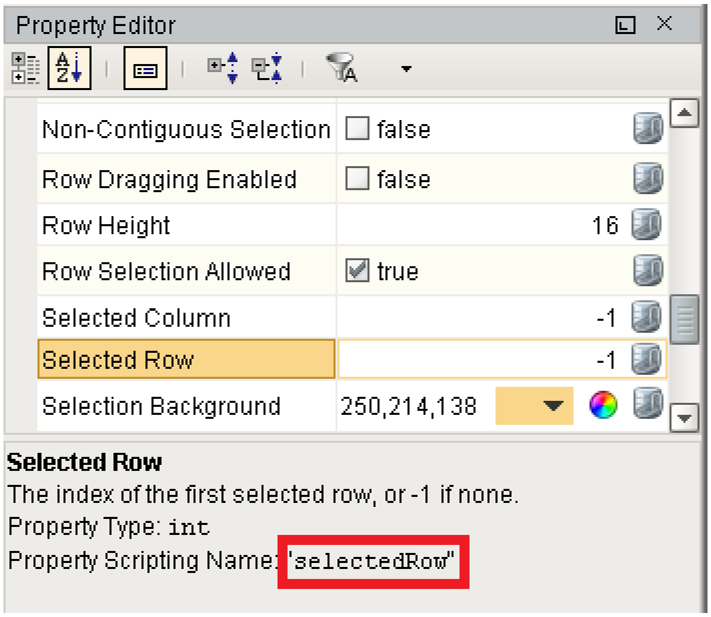
What Should We Look For?

Again, the exception refers to line 5, but note that the error mentions a different object and attribute this time. Without knowing what a **'com.inductiveautomation.factorypmi.application.com'** object is, we can tell that our script is trying to reference an attribute named **'selectedrow'**. This gives us a starting point for troubleshooting the error. If we check line 5, we see the following:

Python - Line 5

```
userSelectedValue = table.data.getValueAt(table.selectedrow, table.selectedColumn)
```

We see **table.selectedrow** as the first parameter being passed to the getValueAt() function. We checked into the `table` variable earlier, so this doesn't necessarily mean that the variable is the problem, although there is no harm in double checking, especially if you have not personally verified the variable. Assuming that the table variable is correct, let's check the attribute. We can check the Property Editor in the Designer to find a matching property, or head over to the Power Table component page in the manual and check the property reference. We can see that there is a **Selected Row** property on the Power Table, but it should be spelled **"selectedRow"** with a capital **"R"**.



Non-Contiguous Selection	□ false
Row Dragging Enabled	□ false
Row Height	16
Row Selection Allowed	☑ true
Selected Column	-1
Selected Row	-1
Selection Background	250,214,138

Selected Row
The index of the first selected row, or -1 if none.
Property Type: int
Property Scripting Name: 'selectedRow'

We also know that the getValueAt() function takes a row index as the first parameter, so it would make sense that our code has a typo.

Solution

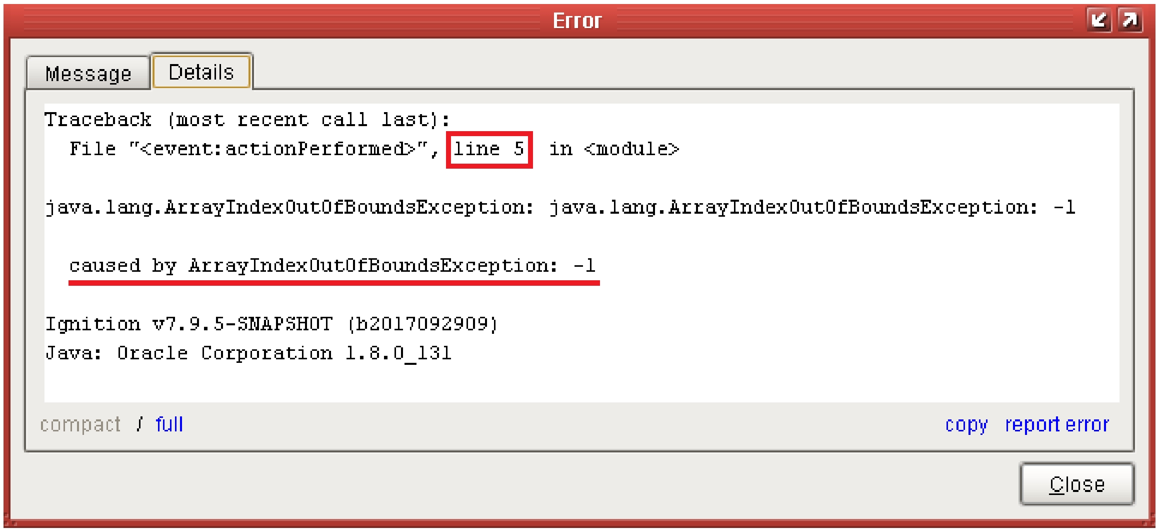
We should fix the typo in the attribute name.

Python - Line 5 Updated

```
userSelectedValue = table.data.getValueAt(table.selectedRow, table.selectedColumn)
```

Third Error - ArrayIndexOutOfBounds

After making our most recent change, the button works great! Users are able to select a cell, and write to the Tag. The one exception, is if the user does not have a cell selected before clicking the button. In this case, the following error occurs:



```
Traceback (most recent call last):
  File "<event:actionPerformed>", line 5  in <module>

java.lang.ArrayIndexOutOfBoundsException: java.lang.ArrayIndexOutOfBoundsException: -1

  caused by ArrayIndexOutOfBoundsException: -1

Ignition v7.9.5-SNAPSHOT (b2017092909)
Java: Oracle Corporation 1.8.0_131
```

Exception Error Explained

Our code attempted to look up something in a collection of some sort (i.e., sequence, dataset, etc.,) by index, but was told to check index -1, which is out of bounds. Indexes in Ignition are typically zero-based, meaning they start at 0, and increment from there, so an index of -1 doesn't exist. Commonly, when a property in Ignition refers to index -1, that means nothing is selected, so our code failed because nothing was selected.

What Should We Look For?

Fortunately, the exception told us what the issue is: the user did not have anything selected before pressing the button, so we don't have to search further.

Solution

The solution to this issue is more open ended, as there are many ways to suggest to our user that they need to select a cell in the table first. The **Selected Row** and **Selected Column** properties on the Power Table will have -1 values if none of the cells are selected, so we could use an **'if'** statement checking the values of one of those properties. Our new code could look like the following:

Python - Code Block Update #3

```
# Create a variable that references the Power Table
table = event.source.parent.getComponent('My Table')

# Make sure a cell in the Power Table is selected first
if table.selectedRow != -1:
```

```
# Find the cell the user currently has selected, and store the value in a variable
        userSelectedValue = table.data.getValueAt(table.selectedRow, table.selectedColumn)

        # Write the User Selected Value to a tag
        system.tag.write("Scripting/ButtonError/WriteTarget", userSelectedValue)

# If a cell isn't selected, then let the user know
else:
        system.gui.messageBox("Please select a cell in the table first!")
```

Related Topics ...

- Reading Error Messages
- Troubleshooting - Nothing Happened

Scripting Vs. SQL Vs. Expressions

On this page ...

Expression Language & SQL Queries vs Scripting

There are three major languages in Ignition, the Expression language, the SQL Queries, and Python Scripting. It is important to understand the differences between the three and to know where each is used. Scripting is used in the event handlers that are available all over Ignition, but Expressions and SQL are in the Property Binding locations shown here.

Spot the Difference - Comments

When starting out with Ignition, it can be difficult to know which syntax you should be using for a particular area of text. One little trick that may help is to be familiar with how each language handles comments, and then utilize the **Ctrl + /** command which automatically adds the characters that comments out a line of code in the language are you are typing in. Once you remember to use **Ctrl + /**, you simply need to be familiar with the characters that each language uses.

Expression Language - Comment

```
// The Expression Language uses double forward slashes.
```

SQL - Comment

```
-- Areas in Ignition that accept SQL syntax use double dashes.
```

Python - Comment

```
# Python uses the pound/number/hash character.
```

In addition to comments, interfaces for each of the languages usually contain other signs or reminders about the language. These will be covered in their respective sections on this page.

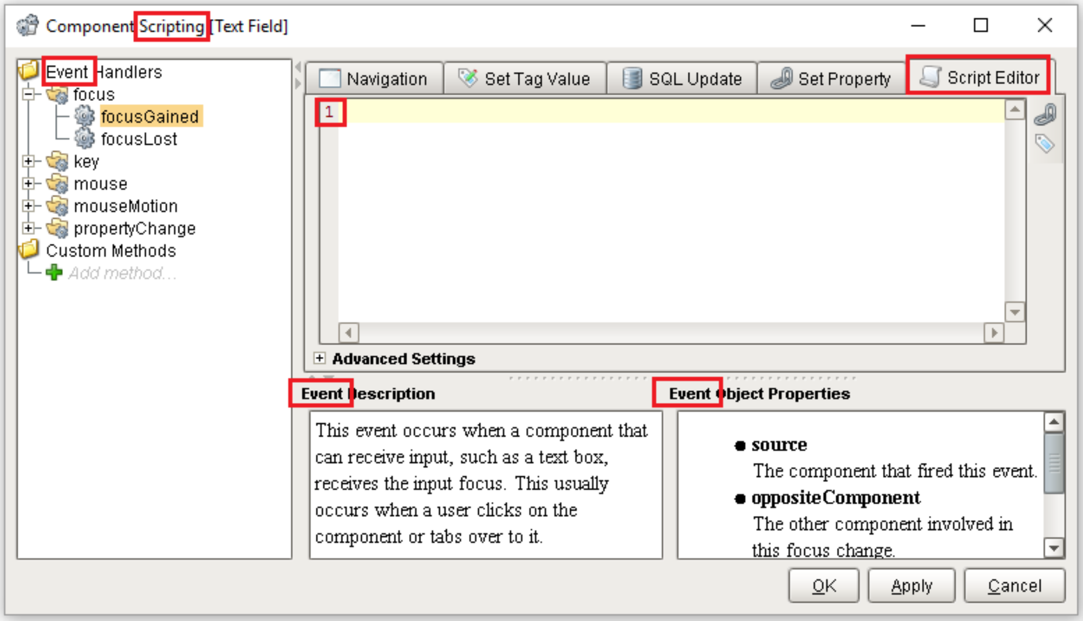
Python Scripts

Overview

Python is featured prominently throughout Ignition and many different resources can contain a Python Script.

How Can I Tell If I'm Writing a Python Script?

Python Scripts typically use the words **Script** or **Event** in the interface. Additionally, Python requires a particular Event to be selected so if you see event Handlers on the left, you know you are looking at a Python script.



This event occurs when a component that can receive input, such as a text box, receives the input focus. This usually occurs when a user clicks on the component or tabs over to it.

- **source**
  The component that fired this event.
- **oppositeComponent**
  The other component involved in this focus change.

Where Are Python Scripts Used?

Python Scripts are used all throughout Ignition. Some resources, such as components, even have multiple places to write scripts! Below are some common locations for scripts:

- Scripting in Vision
- Extension Functions
- Tag Event Scripts
- Client Event Scripts
- Gateway Event Scripts

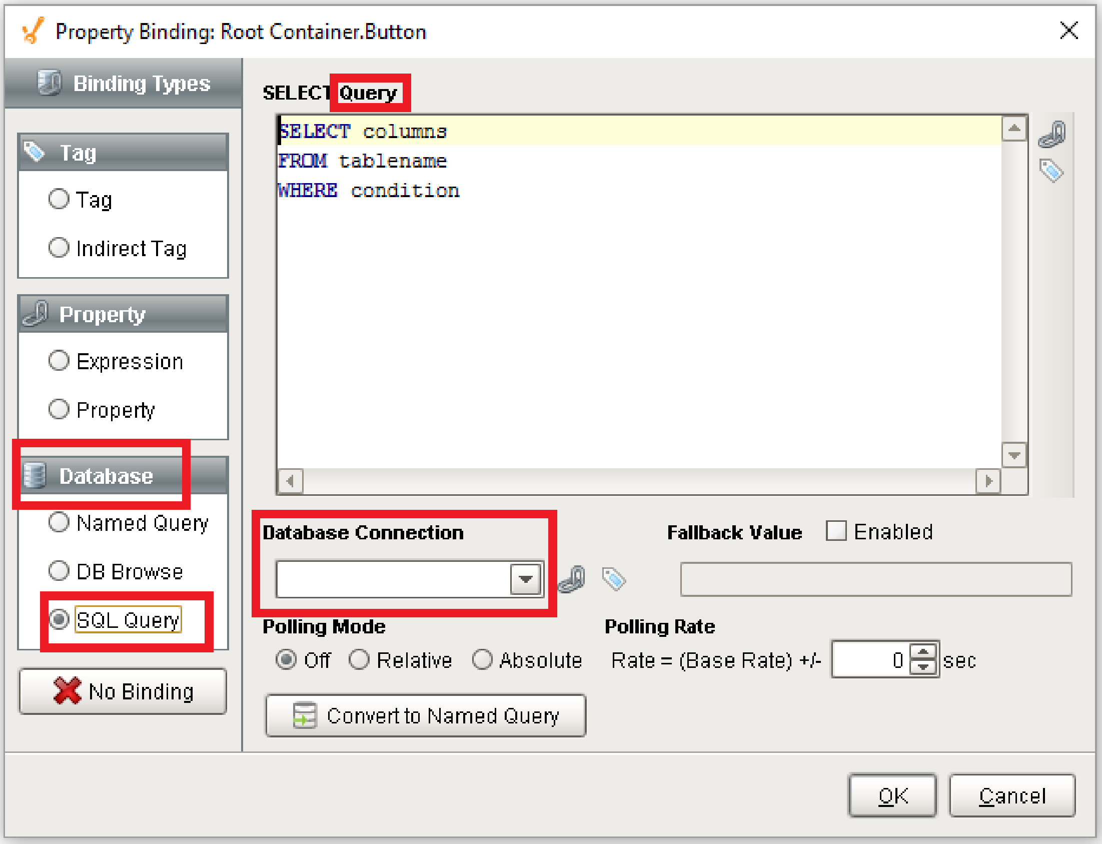
SQL Queries

Overview

The SQL language is used for selecting information from a database. It can be used in a variety of ways, but most of them will either make modifications to the database or return a set of values (with a few notable exceptions like Stored Procedures). The majority of users will be returning large chunks of data into a Table or Report in Ignition. This means a complete dataset will be returned based on a user selection, a time range, or any combination of factors.

How Can I Tell If I Should Be Using SQL Syntax?

Typically, the words **Query** and **Database** appear in the interface somewhere. Additionally, there is usually a way to specify a **Database Connection**.



```
SELECT columns
FROM tablename
WHERE condition
```

Where Is SQL Used in Ignition?

Below is a reference of the most common areas in Ignition where SQL queries may be used.

- SQL Query Bindings
- Named Queries
- Database Query Browser
- Reporting Data Source: SQL Query Data Source and Basic SQL Query
- Query Tags
- Python Scripts - See the SQL in Python header below for more details

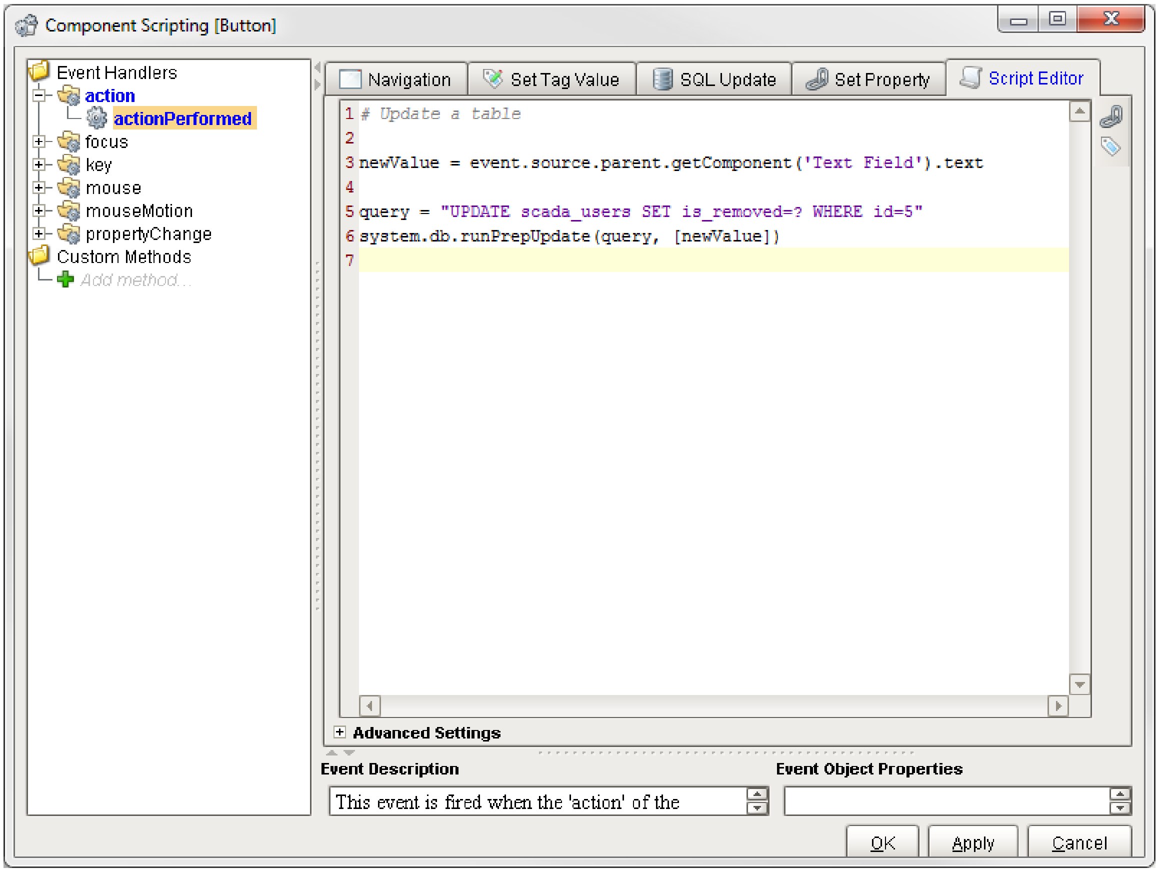
SQL in Python

SQL queries can be called from a **Python** script. There are several system functions in Ignition that allow a script to run a query against the database, such as system.db.runPrepQuery. This is a more advanced technique, as you need to adhere to both language's syntax. Furthermore, when typing a SQL query in a Python script interface, the syntax highlighting can not help with the SQL portions. The syntax highlighting in a Scripting Window is only looking for Python syntax, not SQL.

In cases where you plan on calling a SQL query from a Python script, it is highly recommended to write the query in the **Database Query Browser** first (substituting parameters with static values for testing purposes), and then move the query over to the script once the query executes successfully on its own. This approach can save you some time troubleshooting, as there will be less ambiguity when an error occurs since you know the query runs.

Below we see an example of calling a SQL query in a script. Line 5 creates a variable called "query", and assigns it a string consisting of a prepared statement (using SQL). The query is then executed with the system.db.runPrepUpdate function.

For more examples of using a query in a Python Script, check out the system.db.* functions.



```
# Update a table

newValue = event.source.parent.getComponent('Text Field').text

query = "UPDATE scada_users SET is_removed=? WHERE id=5"
system.db.runPrepUpdate(query, [newValue])

```

Event Description: This event is fired when the 'action' of the

Event Object Properties:

Expression Language

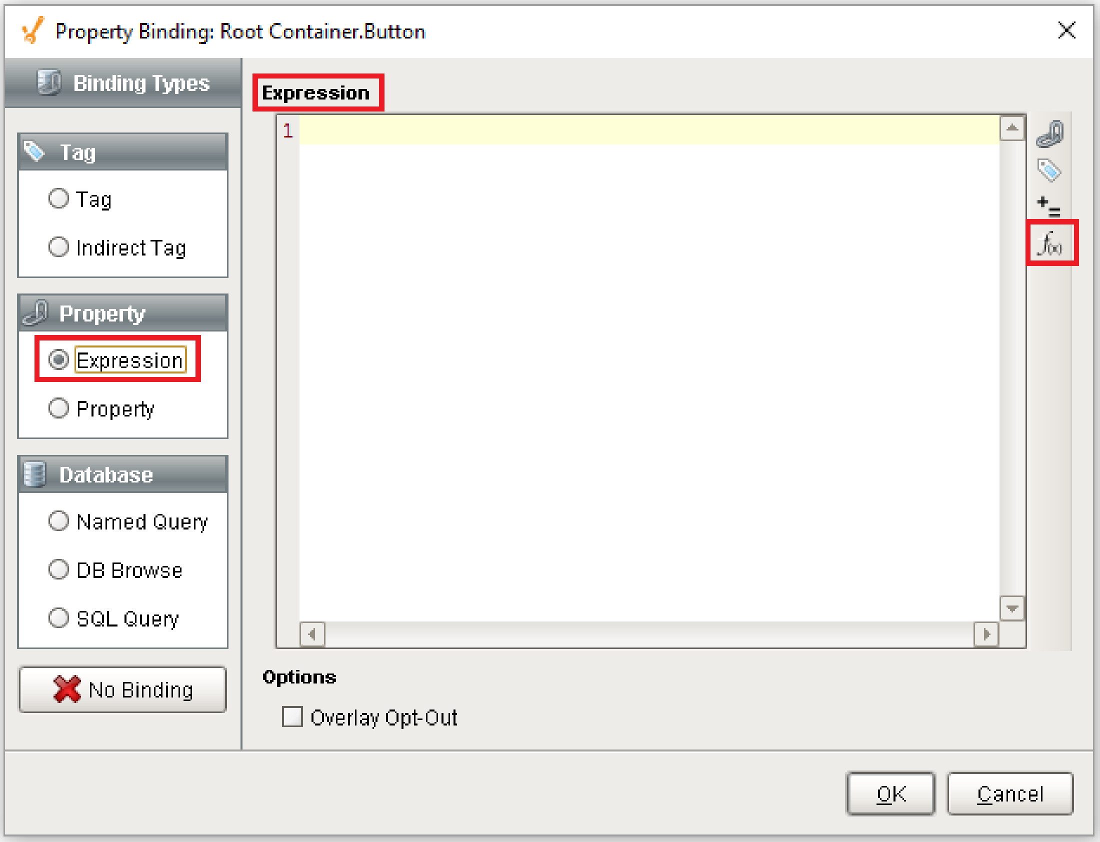
Overview

The Expression Language is a simple programming language that we invented (very similar to many other existing expression languages), and is different from the Python scripting you will find in Ignition. The expression language is a very simple kind of language where everything is an expression - which is a single piece of code that returns a value. This means that there are no statements, and no variables, just operators, literals, and functions.

The most common expression language that most people are familiar with is the one found in Microsoft Excel. You can have Excel calculated a cell's value dynamically by typing an expression like `=SUM(C5:C10)`. Our expression language has similar functionality, but different syntax. It is mainly used to define dynamic values for **Tags** and **component properties**. In Ignition's Expression Language, you will use component properties and functions like if({Root Container.type}="Type C",True,False).

How Can I Tell If I'm Writing an Expression?

Typically interfaces that expect the Expression Language use the word **Expression**. Additionally, you'll commonly find the **Expression Function** button f(x) towards the right side of the text area.



Where Is The Expression Language Used?

Below is a list of resources that commonly utilize the Expression Language

- Expression Bindings on Property Bindings and Alarm Bindings
- Expression Tags and Derived Tags
- Several Alarm Pipeline Blocks, including the Expression Block, Switch Block, and Notification Blocks.
- Expression Items in Transaction Groups
- Parameters on Reports
- SFC Transitions

Related Topics ...

- Expression Overview and Syntax
- SQL in Ignition
- Scripting
- Python Scripting

Scripting Examples

This section contains examples for items we've identified as "common" tasks: undertakings that many users are looking to utilize when first starting out with a specific module or feature in Ignition. Additionally, this section aims to demystify some of the more complex or abstract tasks that our users may encounter.

The examples in this section are self-contained explanations that may touch upon many other areas of Ignition. While they are typically focused on a single goal or end result, they can easily be expanded or modified after the fact. In essence, they serve as a great starting point for users new to Ignition, as well as experienced users that need to get acquainted with a new or unfamiliar feature.

Below is a list of common tasks related to this section of the manual.

On this page ...

Reading and Writing to Tags

There are simple interfaces in Ignition that allow you to easily write to a Tag on some Event, and Reading can be as simple as creating a Tag binding. Sometimes, however, the built-in approaches can be too simplistic or limiting. The Reading and Writing to Tags page details how to better interact with Tags from scripting.

Importing and Exporting a CSV

CSV files are used by many software programs to export data so that other systems may utilize the information contained within. The Importing and Exporting a CSV page demonstrates how to both import a CSV into Ignition, as well as export data from Ignition into a CSV.

Reading a Cell from a Table

Once data is populated into a Table component, it's useful to know how to read and extract a data from a cell in a Table, particularly if users can select a row in a Table. The Read a Cell from a Table page has some good examples for retrieving data from a single cell and multiple cells in a Table.

Export Tag Historian to CSV

The Tag Historian Data can be great, but it's sometimes difficult to view it outside of Ignition. The Export Tag Historian to CSV page details how to pull out a subset of Tag history data and export it to a CSV file.

Related Topics ...

- More Scripting Tasks
- Basic Troubleshooting

In This Section ...

Location Based Vision Startup Scripts

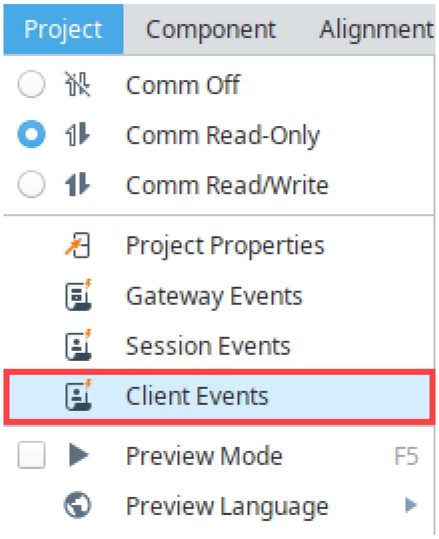
Client Startup Scripts

Security can be created on a Vision Client using a Startup Script that checks certain user information and then customizes the level of access within the project based on that information. We can pull user information from places like System Tags for Vision Clients or Session Properties for Perspective Sessions. Using that information, the script can then customize which windows or views the user sees, provide some additional information to the user, or even prevent the user from accessing the project completely. An additional example is located here.

Location Based Vision Client Restrictions

Each Client has access to distinct hostname and IP addresses. These can be used in Client startup scripts that evaluate the Client's information and compares it to a list of acceptable host names or IP addresses. This information can come from a database or a set of Tags. In this example, we will prevent access to the client if the user is logging in from an incorrect location.

1. On the Project tab, choose Client Events.



2. Select the Startup icon.

3. Add the following script:

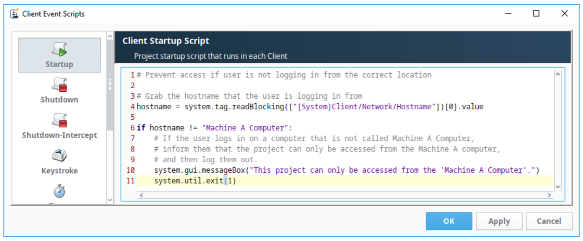
```
# Prevent access if user is not logging in from the correct
location

# Grab the hostname that the user is logging in from
hostname = system.tag.readBlocking(["[System]Client/Network
/Hostname"])[0].value

if hostname != "Machine A Computer":
        # If the user logs in on a computer that is not called
Machine A Computer,
        # inform them that the project can only be accessed
from the Machine A computer,
        # and then log them out.
        system.gui.messageBox("This project can only be
accessed from the 'Machine A Computer'.")
        system.util.exit(1)
```

On this page

...

Custom Security

Watch the Video

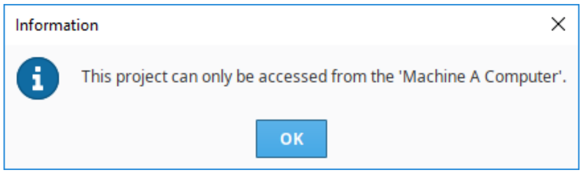


Client Startup Script

Project startup script that runs in each Client

```
# Prevent access if user is not logging in from the correct location

# Grab the hostname that the user is logging in from
hostname = system.tag.readBlocking(["[System]Client/Network/Hostname"])[0].value

if hostname != "Machine A Computer":
    # If the user logs in on a computer that is not called Machine A Computer,
    # inform them that the project can only be accessed from the Machine A computer,
    # and then log them out.
    system.gui.messageBox("This project can only be accessed from the 'Machine A Computer'.")
    system.util.exit(1)
```

4. Click OK to save the script.
5. Save your project and launch the client. You will get a popup that informs you that the project must be launched using the Machine A Computer.

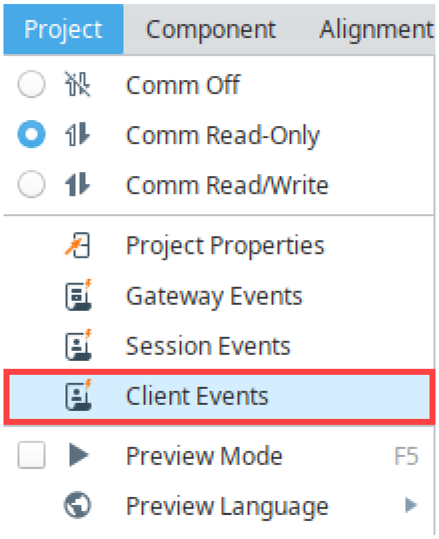


This project can only be accessed from the 'Machine A Computer'.

Location Based Startup Display

Another common use for startup scripts on Vision Clients is to open a specific window depending on the location of the log in. In this example, if the user logs in on Machine A Computer, then the Machine A Details window will be displayed. If they log in on a different computer, the Overview window is displayed.

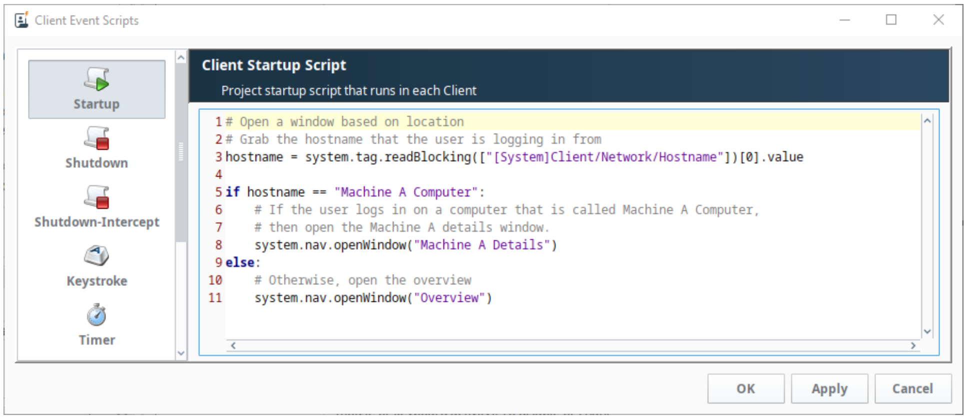
1. On the Project tab, choose Client Events.



2. Select the Startup icon.

3. Add the following script:

```
# Open a window based on location

# Grab the hostname that the user is logging in from
hostname = system.tag.readBlocking(["[System]Client/Network/Hostname"])[0].value

if hostname == "Machine A Computer":
        # If the user logs in on a computer that is called Machine A Computer,
        # then open the Machine A details window.
        system.nav.openWindow("Machine A Details")
else:
        # Otherwise, open the overview
        system.nav.openWindow("Overview")
```



Client Startup Script

Project startup script that runs in each Client

```
# Open a window based on location
# Grab the hostname that the user is logging in from
hostname = system.tag.readBlocking(["[System]Client/Network/Hostname"])[0].value

if hostname == "Machine A Computer":
    # If the user logs in on a computer that is called Machine A Computer,
    # then open the Machine A details window.
    system.nav.openWindow("Machine A Details")
else:
    # Otherwise, open the overview
    system.nav.openWindow("Overview")
```

4. Click OK to save the script.

> ⚠ Note that any windows that are set to Open on Startup will still open in addition to the window specified in the startup script. You should disable any main windows from opening on startup if using this method.

5. Save your project.

Related Topics ...

- Client Tags for Indirection

Reading and Writing to Tags

In many cases, the binding system is the most appropriate way to display a Tag value on the screen. However, you may wish to access a Tag's value in a script. Using the system functions, you can read from a Tag and write to a Tag in Ignition.

On this page ...

Script Builder

If you simply need to write to value to a Tag from a script, you can use the Set Tag Value tab of the Script Editor. To learn more, refer to the sections on Script Builders for more information. However, if you need to do more than just send a single write, see the section below.

Know Your Scope

Whenever a script from a Shared resource (such an Alarm Pipeline) attempts to interact with a Tag, the script must specify the Tag Provider, otherwise, the script could return an exception. Tag Providers are always included at the start of the Tag Path and look like the following: `[tagProvider]`

When interacting with a Tag from the Project scope (such as a script on a component) you may optionally include the Tag provider. If omitted, the Project default Tag provider will be used.

Pseudocode - Tag Read Scope

```
# Reading a Tag without the Tag Provider. You would not want to use
this from the Shared scope.
system.tag.readBlocking(["My/Tag/Path"])

# Reading from the same Tag, but specifying the Tag Provider. This
format may be used safely in either scope.
system.tag.readBlocking(["[default]My/Tag/Path"])
```

Reading and Writing Tags

Watch the Video

Manual Tag Reads

Reading from a Single Tag

Reading a Tag from a script is accomplished with the `system.tag.readBlocking()`function, which requires the Tag path you wish to read from. This function returns a 'Qualified Value'; this object is more than just the value. A Qualified Value is the Tag value that has three attributes; Value, Quality, and TimeStamp.

The function normally expects a list of Tag paths, but can be used with a single Tag: simply provide a list of only a single Tag path.

When handling the results of the read, the results will be returned in a list of qualified values, so you'll need to specify which qualified value you're interested in via Python slicing, even if there is only a single qualified value in the list.

Pseudocode - Reading Tag Attributes

```
# get the Tag value
value = system.tag.readBlocking(["tagPath"])[0].value

# get all three attributes
tag = system.tag.readBlocking(["tagPath"])[0]
value = tag.value
quality = tag.quality
timestamp= tag.timestamp
```

Reading from Multiple Tags

The `system.tag.readBlocking()` function can easily be used to read multiple Tags in a single call:

Python - Reading Multiple Tags and Printing the Values

```
# Create a List of Tag Paths to read
paths = ["Scripting/Tags/Alarm_1", "Scripting/Tags/Alarm_2", "Scripting/Tags/Alarm_3"]

# Read the Tags, and store the complex results in a variable
values = system.tag.readBlocking(paths)

# For each Tag Path, iterate through our results...
for index in range(len(paths)):

        # ...and do something with the individual values
        print values[index].value
```

Relative Tag Paths

Like elsewhere in Ignition, relative paths may be used from within a script. This is especially useful when writing a Tag Event script inside of a UDT, as you can specify relative members in the same UDT with "[.]".

Pseudocode - Reading with Relative Tag Paths

```
# Assuming a UDT with two sub-members, a script from one can read from one member to the other using the following code:
system.tag.readBlocking(["[.]otherMember"]).value
```

Manual Tag Writes

Writing to a Single Tag

Much like reading, there is a function you can use to write to Tags: `system.tag.writeBlocking` . It requires a list of Tag paths, as well as a list of values to write to those Tags.

Pseudocode - Tag Write

```
system.tag.writeBlocking(["tagPath"],["Hello World"])
```

Writing to Multiple Tags

The `system.tag.writeBlocking` function can also write to multiple Tags, again by providing multiple paths and values.

Python - Multiple Tag Writes

```
# Create a List of Tag Paths to write to
paths = ["Scripting/Tags/Alarm_Setpoint_1", "Scripting/Tags/Alarm_Setpoint_2"]

# Create a List of values to write
values = [72, 72]

# Send of the write requests
system.tag.writeBlocking(paths, values)
```

Exporting and Importing a CSV

A CSV file, (comma separated values) is one of the most simple structured formats used for exporting and importing datasets. It is a convenient and flexibe way to edit and share data across applications. Ignition has a built-in function to convert a dataset to CSV data called system.dataset.toCSV. You can even convert the contents of a CSV to a script and move it to an Ignition component, such as a Power Table.

This section contains examples for exporting and importing data to a CSV as well as converting the contents of a CSV to a script.

On this page ...

INDUCTIVE UNIVERSITY

Exporting Data to CSV

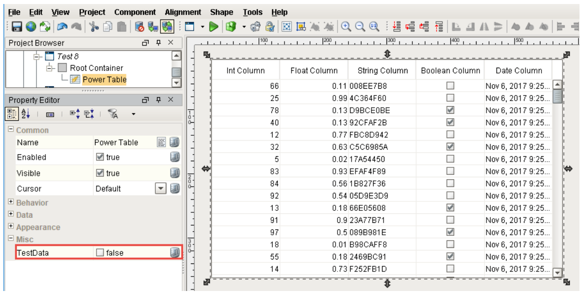
Watch the Video

> The examples on this page make use of Vision components, and run in the Vision Client scope.

Exporting Data to a CSV

You can export a dataset from a query or table to a CSV file.

1. Identify the dataset that you want to export to a CSV file. In this case, we can generate some data on a Power Table and export that data. Drag a **Power Table** component on to your window and toggle its **TestData** property to generate some data.



Name	Power Table
Enabled	☑ true
Visible	☑ true
Cursor	Default
TestData	□ false

Int Column	Float Column	String Column	Boolean Column	Date Column
66	0.11	008EE7B8	□	Nov 6, 2017 9:25...
25	0.99	4C364F60	□	Nov 6, 2017 9:25...
78	0.13	D9BCE0BE	☑	Nov 6, 2017 9:25...
40	0.13	92CFAF2B	☑	Nov 6, 2017 9:25...
12	0.77	FBC8D942	□	Nov 6, 2017 9:25...
32	0.63	C5C6985A	☑	Nov 6, 2017 9:25...
5	0.02	17A54450	□	Nov 6, 2017 9:25...
83	0.93	EFAF4F89	□	Nov 6, 2017 9:25...
84	0.56	1B827F36	□	Nov 6, 2017 9:25...
92	0.54	05D9E3D9	□	Nov 6, 2017 9:25...
13	0.18	66E05608	☑	Nov 6, 2017 9:25...
91	0.9	23A77B71	□	Nov 6, 2017 9:25...
97	0.5	089B981E	☑	Nov 6, 2017 9:25...
18	0.01	B98CAFF8	□	Nov 6, 2017 9:25...
55	0.18	2469BC91	☑	Nov 6, 2017 9:25...
14	0.73	F252FB1D	□	Nov 6, 2017 9:25...

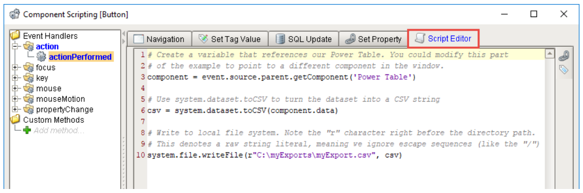
2. Drag a **Button** component on the window, and double click on the Button to open the **Component Scripting** window.

3. Next, let's add our script on the Button component's actionPerformed event. Select the **actionPerformed** event, and click on the **Script Editor** tab.

4. Copy the contents from one of the examples below, and paste the contents to the **Script Editor**. Notice the system.dataset.exportCSV scripting function is used in the first example to export the dataset to a CSV file:

Python - Hard Coded Filepath

```
# Create a variable that references our Power Table. You could
modify this part
# of the example to point to a different component in the
window.
component = event.source.parent.getComponent('Power Table')

# Use system.dataset.toCSV to turn the dataset into a CSV
string
csv = system.dataset.toCSV(component.data)

# Write to local file system. Note the "r" character right
before the directory path.
# This denotes a raw string literal, meaning we ignore escape
sequences (like the "/")
system.file.writeFile(r"C:\myExports\myExport.csv", csv)
```



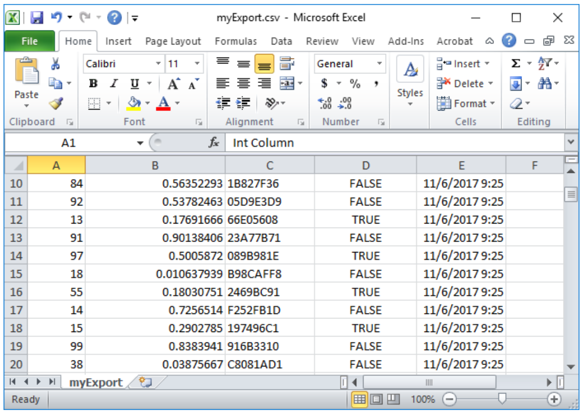
```
# Create a variable that references our Power Table. You could modify this part
# of the example to point to a different component in the window.
component = event.source.parent.getComponent('Power Table')

# Use system.dataset.toCSV to turn the dataset into a CSV string
csv = system.dataset.toCSV(component.data)

# Write to local file system. Note the "r" character right before the directory path.
# This denotes a raw string literal, meaning we ignore escape sequences (like the "/")
system.file.writeFile(r"C:\myExports\myExport.csv", csv)
```

5. Instead of hardcoding the path as we did in the above example, we could ask the user to select a directory on the local system with system.file.saveFile:

Python - User Selected Directory

```
# Create a variable that references our Power Table. You could
modify this part
# of the example to point to a different component in the
window.
component = event.source.parent.getComponent('Power Table')

# Use system.dataset.toCSV to turn the dataset into a CSV
string.
csv = system.dataset.toCSV(component.data)

# Use system.file.saveFile to have the user find a directory
to write to.
filePath = system.file.saveFile("myExport.csv", "csv", "Comma
Separated Values")

# We can check the value of filePath to make sure the user
picked a path before
# attempting to write.
if filePath:
        system.file.writeFile(filePath, csv)
```

6. To test your scripts, put the Designer in **Preview Mode**, and press the **Button**. Open your **myExport.csv** file and check your data.



A1 | Int Column

	A	B	C	D	E	F
10	84	0.56352293	1B827F36	FALSE	11/6/2017 9:25	
11	92	0.53782463	05D9E3D9	FALSE	11/6/2017 9:25	
12	13	0.17691666	66E05608	TRUE	11/6/2017 9:25	
13	91	0.90138406	23A77B71	FALSE	11/6/2017 9:25	
14	97	0.5005872	089B981E	TRUE	11/6/2017 9:25	
15	18	0.010637939	B98CAFF8	FALSE	11/6/2017 9:25	
16	55	0.18030751	2469BC91	TRUE	11/6/2017 9:25	
17	14	0.7256514	F252FB1D	FALSE	11/6/2017 9:25	
18	15	0.2902785	197496C1	TRUE	11/6/2017 9:25	
19	99	0.8383941	916B3310	FALSE	11/6/2017 9:25	
20	38	0.03875667	C8081AD1	FALSE	11/6/2017 9:25	

Importing Data from a CSV

There are several ways to import data from a CSV file. First, we could use system.file.readFileAsString to read the entire file as a string. Note, that this will read the file as is, meaning `"\n"` can be used to denote new lines.

Python - Using system.file.readFileAsString()

```
# Ask the user to find the CSV in the local file system.
path = system.file.openFile("csv")

# Use readFileAsString to read the contents of the file as a string.
# This string will be the parameter we pass to fromCSV below
stringData = system.file.readFileAsString(path)

# Split stringData into a List of strings, delimited by the new line character
```

```
stringData = stringData.split("\n")

# Iterate through the list, and do something with each line
for i in range(len(stringData)):

        # We're printing the row here, but you could do something more useful with the data.
        print stringData[i]
```

Alternatively, Python's CSV Library could be use to read in the contents of a CSV. In some cases, this is the easier approach, as the reader object is ready to be iterated over. Note, that this approach does read in each row as a List of strings:

Python - Using csv.reader()

```
# Import Python's built-in csv library
import csv

# Ask the user to find the CSV in the local file system.
path = system.file.openFile("csv")

# Create a reader object that will iterate over the lines of a CSV.
# We're using Python's built-in open() function to open the file.
csvData = csv.reader(open(path))

# Iterate through the reader object, and do something with each line
for row in csvData:

        # We're printing the row here, but you could do something more useful with the data.
        print row
```

Converting the Data into a Dataset

Once you've read in the contents of a CSV into a script, you may wish to move it elsewhere. It is not uncommon to move the data to a **Power Table**, or other components on the screen. The main difficulty with this is converting the CSV data into a dataset so that it fits into the **Power Table** component's **Data** property. We have a couple of approaches listed below.

Calling the system.dataset.fromCSV Function

The system.dataset.fromCSV function can take a string and convert it to a dataset. Note, that the function expects a very specific format:

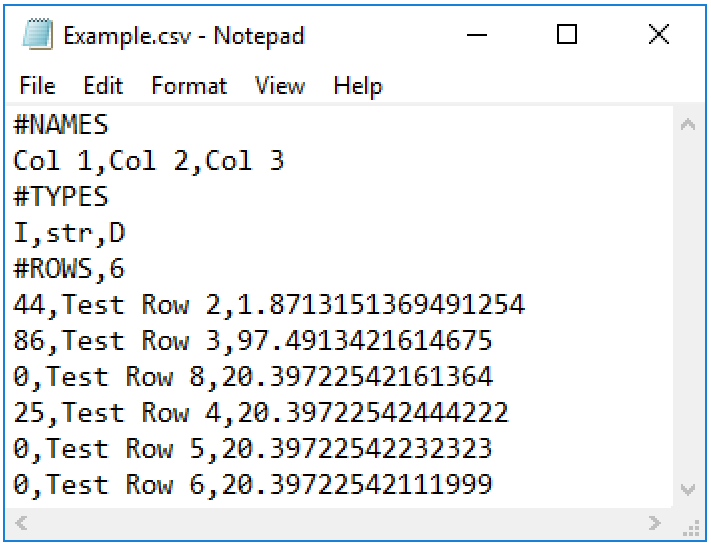
CSV File Content

```
#NAMES
Col 1,Col 2,Col 3
#TYPES
I,str,D
#ROWS,6
44,Test Row 2,1.8713151369491254
86,Test Row 3,97.4913421614675
0,Test Row 8,20.39722542161364
25,Test Row 4,20.39722542444222
33,Test Row 5,20.39722542232323
62,Test Row 6,20.39722542111999
```

Example

1. Create a Text file on your local system named "**example.csv**."

2. Open the Text file, copy the contents of the "**CSV file Content**" box above, and paste the contents to the file. Save the changes to the **example.csv** file.

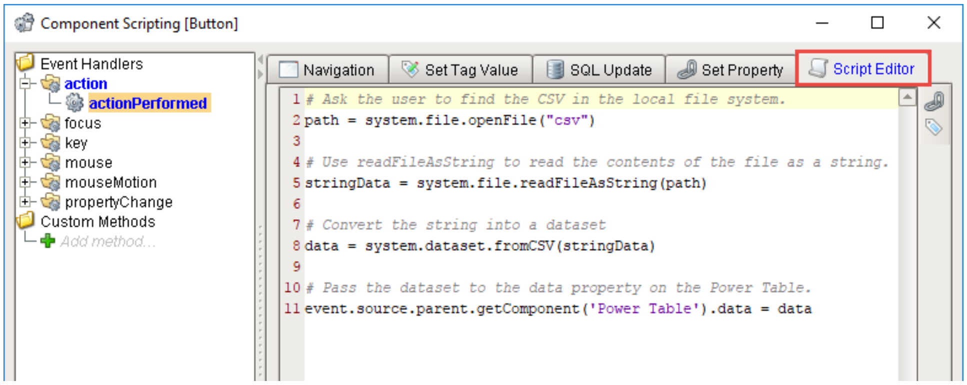


```
#NAMES
Col 1,Col 2,Col 3
#TYPES
I,str,D
#ROWS,6
44,Test Row 2,1.8713151369491254
86,Test Row 3,97.4913421614675
0,Test Row 8,20.39722542161364
25,Test Row 4,20.39722542444222
0,Test Row 5,20.39722542232323
0,Test Row 6,20.39722542111999
```

3. Let's move the contents of a CSV file to a Power Table. Add a **Power Table** and a **Button** component to your window. Double click on the **Button** component, and paste the following code into the **Script Editor** of the **actionPerformed** event:

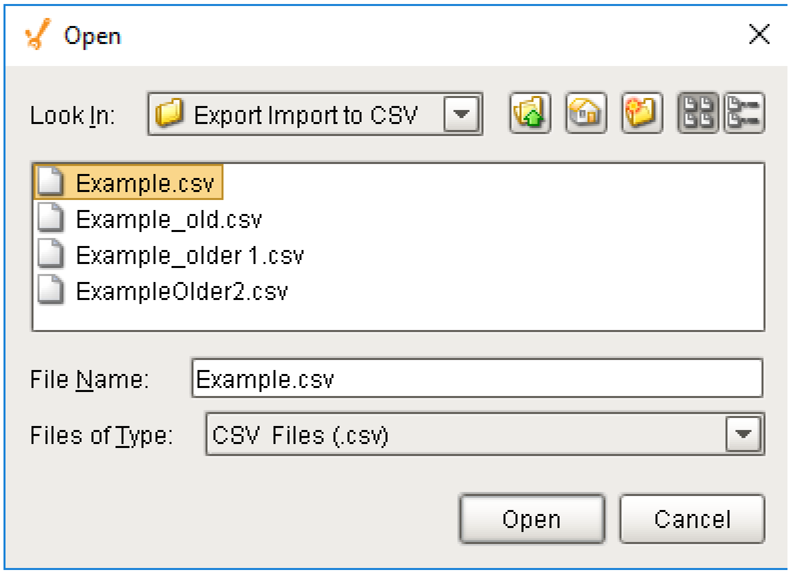
Python - Using system.dataset.fromCSV()

```
# Ask the user to find the CSV in the local file system.
path = system.file.openFile("csv")

# Use readFileAsString to read the contents of the file as a string.
stringData = system.file.readFileAsString(path)

# Convert the string into a dataset
data = system.dataset.fromCSV(stringData)

# Pass the dataset to the data property on the Power Table.
event.source.parent.getComponent('Power Table').data = data
```



```
# Ask the user to find the CSV in the local file system.
path = system.file.openFile("csv")

# Use readFileAsString to read the contents of the file as a string.
stringData = system.file.readFileAsString(path)

# Convert the string into a dataset
data = system.dataset.fromCSV(stringData)

# Pass the dataset to the data property on the Power Table.
event.source.parent.getComponent('Power Table').data = data
```

4. To execute your script, put the Designer into **Preview Mode**, and press the **Button**. A window will open for you for you to navigate and choose your CSV file, then click Open.



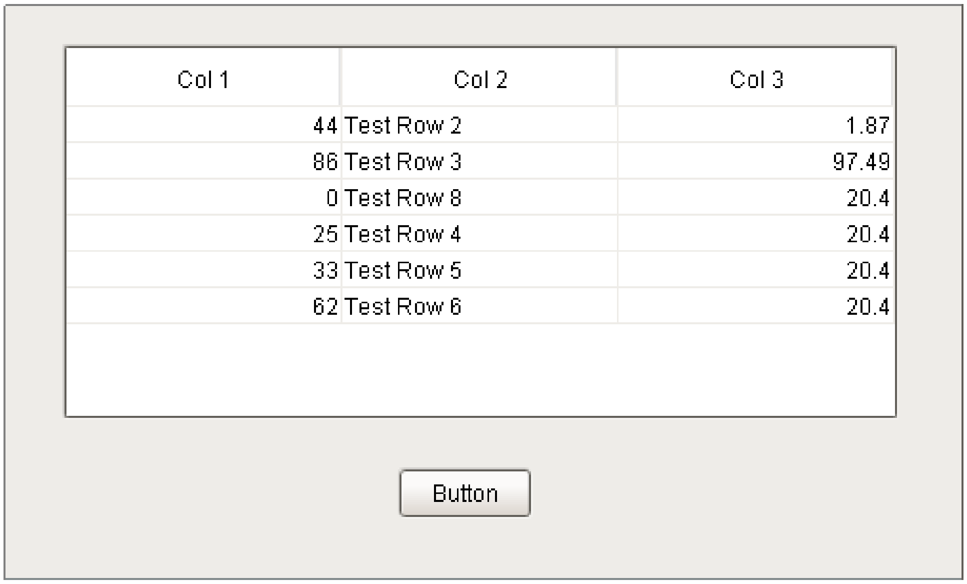
Look In: Export Import to CSV

Example.csv
Example_old.csv
Example_older 1.csv
ExampleOlder2.csv

File Name: Example.csv

Files of Type: CSV Files (.csv)

5. Your data will be displayed in the Power Table as shown below.



Col 1	Col 2	Col 3
44	Test Row 2	1.87
86	Test Row 3	97.49
0	Test Row 8	20.4
25	Test Row 4	20.4
33	Test Row 5	20.4
62	Test Row 6	20.4

Calling the csv.reader Function

As mentioned, system.dataset.fromCSV() requires a specific format, which may not match the format of your file. In this case, we can use Python's CSV Library to parse the file and convert it to a dataset.

CSV File Content

```
Col 1,Col 2,Col 3
44,Test Row 2,1.8713151369491254
86,Test Row 3,97.4913421614675
0,Test Row 8,20.39722542161364
25,Test Row 4,20.39722542444222
33,Test Row 5,20.39722542232323
62,Test Row 6,20.39722542111999
```

Here is the code to import the above CSV data.

Python - Using Python's csv Library

```
# Import Python's built-in csv library.
import csv
```

```
# Ask the user to find the CSV in the local file system.
path = system.file.openFile("csv")

# Create a reader object that will iterate over the lines of a CSV.
# We're using Python's built-in open() function to open the file.
csvData = csv.reader(open(path))

# Create a List of strings to use as a header for the dataset. Note that the number
# of headers must match the number of columns in the CSV, otherwise an error will occur.
# The simplest approach would be to use next() to read the first line in the file, and
# store that at the header.
header = csvData.next()

# Create a dataset with the header and the rest of our CSV.
dataset = system.dataset.toDataSet(header ,list(csvData))

# Store it into the table.
event.source.parent.getComponent('Power Table').data = dataset
```

Related Topics ...

- system.dataset.exportCSV

Adding a Delay to a Script

On this page ...

Overview

In some cases, having a script execute after a delay is preferable. A common use case is waiting for some event elsewhere in the system to finish: a Tag change script executes that needs to wait for a new value from a separate Tag. One approach to this is to trigger our script, and then hold or wait until the other event occurs. On this page, we'll take a look at a couple of different approaches to this problem.

It is important to note that pausing a script can cause your client to lock. It is often preferred to look for another event to trigger the script you need. For the example above where we are waiting on a Tag to change, you might be able to use the Tag change to fire your script instead of waiting in the original script. It is up to you to determine the best trigger based on what exactly your script does.

Using the system.util.invokeLater Function

The system.util.invokeLater function is a great way to add a delay mid-way through the script. Simply create a function that represents all of the work that should occur after the delay, and pass the function to invokeLater, along with a delay period.

The example below calls two Message Boxes: once initially when pressed, and the other after a three second delay.

Python - Two Message Boxes

```
message = "All Done!"

# Create a function that will be called by invokeLater.
def runThisLater():
        system.gui.messageBox(message)

# Call invokeLater with a 3000ms delay. Note that our function will
not run immediately
# because invokeLater always executes once the rest of this script is
complete.
system.util.invokeLater(runThisLater, 3000)

# Bring up another Message Box. This will appear before the "All
Done!" message, because of invokeLater.
system.gui.messageBox("Waiting...")
```

One of the main limitations with invokeLater is that you can not pass parameters to the function that will be called. Parameters need to be initialized and determined elsewhere in your script, usually in the function definition.

Using a Timer from Python's Threading Library

The Threading Library has a Timer function that works in a very similar fashion to invokeLater. The main difference is that you can pass parameters to the function parameter when calling the Timer. Take a look at Python's official documentation for more details on the Timer object.

The example below will again call two Message Boxes with a three second delay between them. However, the text that is defined in the second Message Box is specified when starting the Timer.

Python - Python Threading Timer

```
from threading import Timer

# Create a function that will be called by the Timer.
def runThisLater(param):
        system.gui.messageBox(str(param))

# Constructs a Timer object that runs a function after a specified interval of time has passed.
# Note the start() at the end: this is required to start the Timer. Don't forget this part!
Timer(3.0, runThisLater, ["Stop Waiting, I'm done"]).start()
```

```
# Bring up another Message Box. This will appear before the "All Done!" message, because of the Timer
system.gui.messageBox("Waiting")
```

Another benefit to using the Timer, is you can cancel the execution of the Timer's action using the cancel() function, but it only works if the Timer is still in its waiting stage.

Calling time.sleep from Python's Time Library

The simplest approach to pausing a script is to use the sleep() function in Python's Time Library:

Python - Sleeping the Code

```
from time import sleep

# This will pause execution of the script for 3 seconds. After that time, the script will continue.
sleep(3)

print "I'm awake!"
```

Using a While Loop

The While Loop is another simple approach to adding a delay: simply keep looping until the other event occurs. As always, you will want to take steps to ensure that an infinite loop never occurs: easiest by initializing a counter variable before iterating in the loop, and breaking out if the counter reaches a certain value.

Python - While Loop Safeguard

```
counter = 0

while not otherEvent:

        checkOtherEvent()
        counter += 1

        # Use this to break out if the event takes too long. You'll need the rest of your script to
account for this possibility.
        if  counter >= 10000:
                print "Took too long. Leaving the While Loop"
                break
```

Approaches to Avoid - Locking the Client

While the sleep() and while loop functions are simple to use, they can cause problems by locking up the client and because of this, we generally recommend avoiding them if possible. Typically, system.util.invokeLater on a delay can accomplish the same task.

Note, that there are use cases for both approaches outlined below especially so in regard to the While Loop. However, neither function should be used to force a delay in a Client.

Using either of these methods to pause or delay a script can lock up the entire client, which may be very dangerous.

Reasons to Avoid time.sleep and While Loops

If used on a component, these approaches could lock up the Client or Designer. Scripts called from a component run in the same thread that the Client and Designer use to refresh the screen. Whenever a script on a component triggers, the screen is unable to refresh until the script finishes. In most cases, this is so fast that no one notices. If your script is intentionally calling sleep() or using a long running While Loop, then the Client will lock up for the duration.

Additionally, these approaches run an extended period of time, and block other component based scripts from executing. They do not yield to other scripts while waiting.

Because of these two reasons, the sleep() and While Loop functions can cause your clients to appear unresponsive, when really they are just running a script that prevents the screen from refreshing.

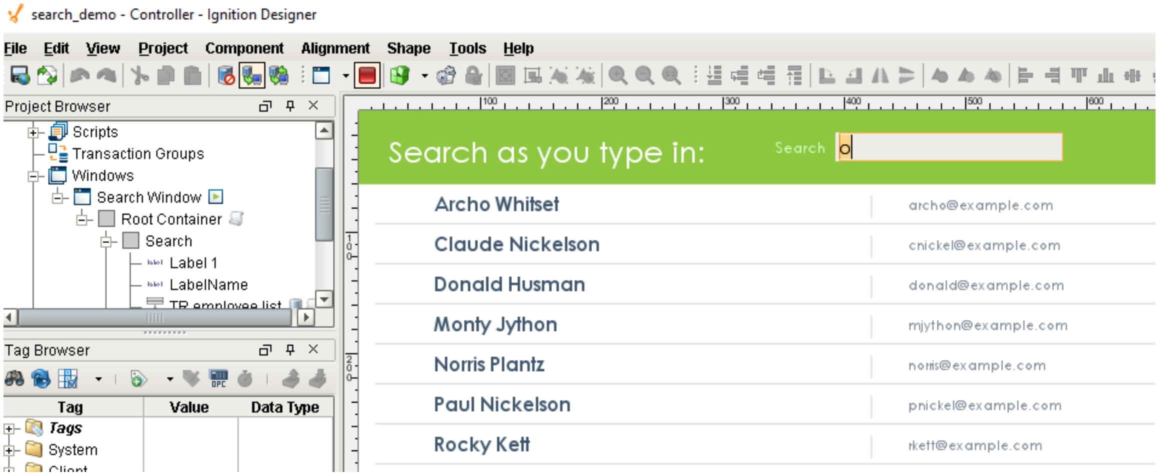
Recommended Alternative

As mentioned earlier, the system.util.invokeLater function (also mentioned on this page) can provide the same functionality. If a sleep() or While Loop function must be used, then they should be called from system.util.invokeAsynchronous to prevent blocking any other scripts.

Demonstration - Executing a Delay Between User Keyboard Input

In many cases, you may need to show your users a large number of entries on a Table or Template Repeater. As more entries are added to the system, adding a search field that can filter results becomes more appealing. Furthermore, being able to filter as the user types (as opposed to forcing them to hit enter every time) adds some polish to the window. However, if the entries are backed by the database, you will not want to run a query every time a user presses a key. Instead, it would be preferable to wait for a delay in input, and run one query to limit strain on the database.

The following is not a traditional example that you would find here in the User Manual, and assumes you are comfortable with many concepts in Ignition.



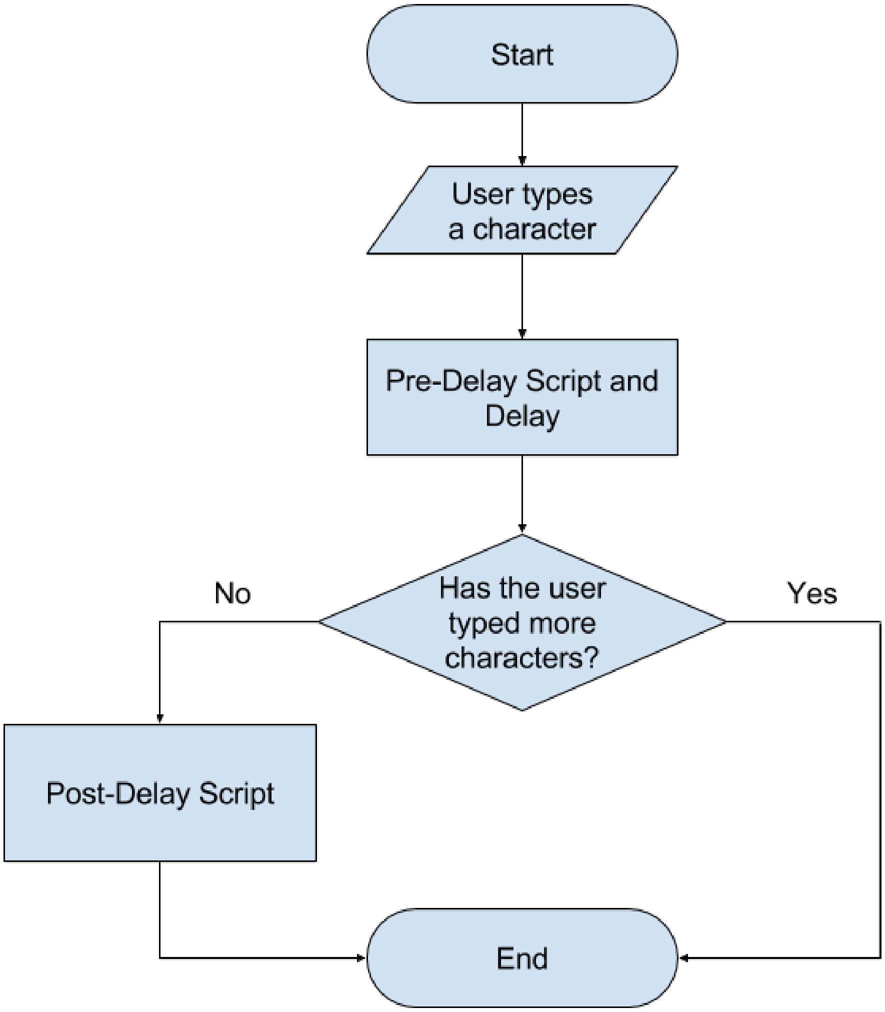
o

Archo Whitset	archo@example.com
Claude Nickelson	cnickel@example.com
Donald Husman	donald@example.com
Monty Jython	mjython@example.com
Norris Plantz	norris@example.com
Paul Nickelson	pnickel@example.com
Rocky Kett	rkett@example.com

Workflow for a Delay Using Two Scripts

One approach to adding a delay involves two scripts:

- **Pre-Delay Script**: This script fires before the delay. In this case, we will use a script on the Text Field's keyReleased event. This will call the script everytime a new key is entered into the Text Field, and calls the Post-Delay Script on a delay. It also passes the text that is currently in the Text Field, and increments an edit count (could be configured as a custom property on the same component).
- **Post-Delay Script**: This script runs after the delay, so it needs to check if there were any new keystrokes that occurred during the delay. If there were no new keystrokes, then it can run the query. In this demonstration, we will use a Custom Method not the Text Field component.


Start
User types a character
Pre-Delay Script and Delay
Has the user typed more characters?
No
Yes
Post-Delay Script
End

Additionally, we will need some criteria that our script can use to determine if it is safe to run our query. You can have multiple criteria here, but for the sake of simplicity we will use the **Text Property on the Text Field**. This way we will know what the user typed before the delay, and then can compare it back to the Text property after the delay. If the Text on the component after the delay is different, then we don't want to run the query as the user may still be typing.

The keyReleased script is shown below. It is using a .05 second delay, but could of course can be modified. We're creating a new Timer object every time a key is released, so there will be many of these scripts firing as the user types. However, they are fairly lightweight, and don't directly interact with the database, so having many executions of this script isn't taxing on the system.

Python - Pre-Delay Script

```
# We're using the Timer object for this because we want to pass parameters to the function that will run
after the delay.
from threading import Timer

# Ignoring arrow keys.
if (event.keyCode < 37 or event.keyCode > 40):

        # Grab the text in the component currently.
        currentText = event.source.text

        # Calls the function after 0.5 seconds, and pass the currentText.
        Timer (0.5, event.source.sendingQuery,[currentText]).start()
```

Below is the sendingQuery script. As mentioned, this script will determine if the user ceased keyboard activity before running a query against the database. This example is using a Custom Method on the Text Field, so it uses the `self` argument to reference the source component.

Python - Post-Delay Script

```
def sendingQuery (self, oldText):

    # Read the value of the text property, since it may have changed during the delay.
    currentSearchText = self.text

    # If the text before the delay is the same as the after the delay, there has been a pause in
keyboard activity, so we should run the query.
    if (currentSearchText == oldText):

        # If the text field isn't empty, then we need to filter the results with a WHERE clause
and our criteria.
        if(currentSearchText != ""):
            newQuery = "SELECT * FROM employees WHERE CONCAT(firstname, ' ', lastname) like
'%" + currentSearchText + "%'"
        # If the Text Field Is blank, then we should run the query again, but this time without
a filter, so all results appear.
        else:
            newQuery = "SELECT * FROM employees"
```

We can now run our query in between user keyboard activity.

Related Topics ...

- system.util.invokeLater
- Timer Scripts

Export Tag Historian to CSV

On this page

...

Obtaining Historian Data

Sometimes, it may be useful to export the values from the Tag Historian. There are many ways that this can be done, but the easiest way is to use the built in system.tag.queryTagHistory function. This allows you to specify a list of Tag paths to query history for.

Python - Grabbing History Data

```
# First we start by creating a start and end time that we will be query history for.
endTime = system.date.now()
startTime = system.date.addMinutes(endTime, -30)

# Next we call our queryTagHistory function, using the start and end dates as well as a Tag path.
# The other parameters listed for this function can be altered to fit your need.
data = system.tag.queryTagHistory(paths=["[default]myTag"],
startDate=startTime, endDate=endTime, returnSize=10, aggregationMode="Average", returnFormat='Wide')
```

This simple script will query Tag History for a Tag called myTag. It will query the last 30 mins of data, return 10 records, each record will be an average of the time slice, and the return format will be wide. For more information on the function parameters and the different options available, see the queryTagHistory function page. The data returned is then being stored in a variable **data**, and we can use it later in the script in anyway we want.

Exporting to CSV

Now that we have our data, it is really easy to export it to a CSV file. Ignition has a built-in function to convert a dataset to CSV data called system.dataset.toCSV. This CSV data can then be written to an actual file using system.file.writeFile.

Python - Export History to CSV

```
# First we start by creating a start and end time that we will be query history for.
endTime = system.date.now()
startTime = system.date.addMinutes(endTime, -30)

# Next we call our queryTagHistory function, using the start and end dates as well as a Tag path.
# The other parameters listed for this function can be altered to fit your need.
data = system.tag.queryTagHistory(paths=["[default]myTag"],
startDate=startTime, endDate=endTime, returnSize=10, aggregationMode="Average", returnFormat='Wide')

# Turn that history data into a CSV.
csv = system.dataset.toCSV(data)

# Export that CSV to a specific file path. The r forces it to use the raw path and not require double backslashes.
system.file.writeFile(r"C:\myExports\myExport.csv", csv)
```

History Tag Search and Export

If you have a lot of Tags storing history, it can be difficult to list them all out in the function above. This example makes things easier by specifying a folder of Tags instead of individual Tags. The script first grabs all of the historical Tags from a particular folder and gets the Tag paths using the system.tag.browseHistoricalTags function, runs those through the system.tag.queryTagHistory function, and then exports the results to a CSV file. This script is great, because it can be run from the scripting console, or from a button on the client. However, everything in the script is hardcoded to work with a specific system, though that can be easily changed to fit any system. Below are the list of parts that can be changed with what each of them is used for.

Part of Script	Description

`path='histprov:myDB:/drv:myIgnitionGateway:myTagProvider:/tag:myFolderOfTags'`	The path to the parent folder that we are searching through for history Tags. Each part of the string corresponds to a different part of your system. - histprov:myDB:/ - This is the name of your Historical Tag Provider that you are searching in, simply replace **myDB** with the name of your historical Tag Provider. - drv:myIgnitionGateway:myTagProvider:/ - The first thing is the name of your Ignition Gateway. Simply replace **myIgnitionGateway** with the name of your Ignition Gateway, which can be found by going to the Gateway Configure Webpage and navigating to **System > Gateway Settings**, and it is the **System Name**. The second thing is the name of the Realtime Tag Provider where the Tags reside. Simply replace **myTagProvider** with the name of your Tag Provider. - tag:myFolderOfTags - This is the name of the folder you want to search for historical Tags. Simply replace **myFolderOfTags** with the name of a folder in your Tag Browser to search through.
`system.tag.queryTagHistory(...)`	This is what will actually query for historical data. While the path parameter should be kept the same, the other parameters can be altered to fit your needs. Currently, the function is querying the last 30 mins of history, has a return size of 10, an aggregation mode of Average, and a return format of wide. Of course, all of this can be changed. See system.tag.queryTagHistory for more information on the parameters that can be used here.
`r"C:\myExports\myExport.csv"`	This is the path that the CSV file will save at. This can be changed to be any location you want. You can even use system.file.saveFile to first ask the user for a save location and filename, and then passing the path returned from that function into the writeFile function. The r at the beginning of the string is important as it allows us to not have to use double backslashes.

This script can be dangerous! It recursively looks through a folder to find all historical Tags, and then uses all of those Tags to look up Tag history. Looking through a folder that has too many history Tags, or querying history for too large a period of time can potentially result in locking up your system. Use caution when using this function, and never search through too large a set of Tags or query too much history.

Python - Recursively Browse for History and Export to CSV

```python
# Our browse function that will browse for historical tags.
# By setting this up as a function, it allows us to recursively call it to dig down through the specified folder.
# Pass in an empty list that we can add historical paths to, and the path to the top level folder.
def browse(t, path):

        # Loop through the results of the historical tag browse, and append the path to the empty list.
        for result in system.tag.browseHistoricalTags(path).getResults():
                t.append(result.getPath())

                # If the result is a folder, run it through the browse function as well.
                # This will continue until we are as deep as possible.
                if result.hasChildren():
                        browse(t, result.getPath())

# Start with an empty list to store our historical paths in.
historyPaths = []

# Call the browse function, passing in an empty list, and the folder that we want to browse for historical tags.
# This path is a placeholder. It should be replace with your valid path.
browse(historyPaths, path='histprov:myDB:/drv:myIgnitionGateway:myTagProvider:/tag:myFolderOfTags')

# Create another empty list to store our tag paths that we will pull out of the historical paths.
tagPaths = []

# Loop through the list of historical tag paths, split out just the tag path part,
# and push it into our tag path list.
for tag in historyPaths:
        tagPaths.append("[myTagProvider]" + str(tag).split("tag:")[1])

# Now that we have a list of tag paths, we need to grab the historical data from them.
# Start by creating a start and end time.
endTime = system.date.now()
startTime = system.date.addMinutes(endTime, -30)

# Then we can make our query to tag history, specifying the various parameters.
# The parameters listed for this function can be altered to fit your need.
data = system.tag.queryTagHistory(paths=tagPaths, startDate=startTime, endDate=endTime, returnSize=10,
aggregationMode="Average", returnFormat='Wide')

# Turn that history data into a CSV.
csv = system.dataset.toCSV(data)

# Export that CSV to a specific file path. The r forces it to use the raw path with backslashes.
system.file.writeFile(r"C:\myExports\myExport.csv", csv)
```

Related Topics ...

- Exporting and Importing a CSV

Parsing XML with the Etree Library

What is the xml.etree Library?

The etree (ElementTree) library is a part of the python standard library, and contains many tools that make it simple to parse through and pull information out of an XML document. There are other libraries that can parse through XML documents, but etree is commonly used and very easy to get started with. The etree library will break up the XML into easily accessible elements, each representing a single node in the entire XML tree. For more information on using the etree library beyond the scope of this page, see the python documentation.

On this page ...

Using the xml.etree Library

There are a couple of different ways to import the XML data from etree, depending on how it is being stored. It can pull the data in from an XML file using the filepath, or it can read a string. Notice how regardless of how we import the XML, we end up with a root object.

Python - Reading a File

```
# The library must first be imported no matter how we pull in the data.
import xml.etree.ElementTree as ET

# Here we can grab the filepath using Ignition's built in openFile function, parse that into a tree,
then grab the root element.
filepath = system.file.openFile()
tree = ET.parse(filepath)
root = tree.getroot()
```

Python - Reading from a String

```
# The library must first be imported no matter how we pull in the data.
import xml.etree.ElementTree as ET

# Alternately, we can start with a string of the xml data.
xmlString = """
<employee id="1234">
        <name>John Smith</name>
        <start_date>2010-11-26</start_date>
        <department>IT</department>
        <title>Tech Support</title>
</employee>
"""

# Then parse through the string using a different function that takes us straight to the root element.
root = ET.fromstring(xmlString)
```

Each Tag is considered an element object. In the example above, the root element would be the employee Tag. Elements can also have attributes, which are within the Tag itself. In the example above, the employee element has an attribute id with a value 1234. Finally, each element can also have additional data, typically in the form of a string. This additional data is usually placed in between the element's start and end Tags. In the example above, the employee element has no additional data, but its children do. The name element would have an additional data value of John Smith. All of this data can be accessed using the Element object's built-in functionality. The major functions are listed below, and each example uses the reading from a string root XML example from above.

Function	Description	Example	Output
Element.tag	Returns the name of the Element's Tag.	`print root.tag`	employee
Element.attrib	Returns a dictionary of the Element's attributes.		{'id':'1234'}

		print root.attrib	
Element.text	Returns the additional data of the Element. The example here will return nothing because the root does not have any text. The next example uses children which do have text.	print root.text	
for child in Element	Will iterate through the Element's children. Each child is then its own element, complete with Tag, attrib, and text properties.	for child in root: print child.tag, child.text	name John Smith start_date 2010-11-26 department IT title Tech Support
Element[index]	Allows direct reference to an Element's children by index. Since Tags can be nested many times, further nested children can be accessed by adding an additional index in square brackets as many times as necessary: Element[1][4][0] From the original element, we would go to the child located in the first position, that child's fourth position child, and that child's zero position child. When direct referencing child elements in this way, they still have access to the Tag, attrib, and text properties.	root[2].tag root[3].text	department Tech Support

A Simple Book Example

Using the functions above, we can now easily parse through an XML file and use the results for something. Lets keep it simple, and parse through a document and then place the values into a table. First we need to start with an XML document. We have one below for you to test with in a string form, which would need to be pasted at the top of the script.

XML String

```
document = """
<catalog>
        <book id="bk101">
                <author>Gambardella, Matthew</author>
                <title>XML Developer's Guide</title>
                <genre>Computer</genre>
                <price>44.95</price>
                <publish_date>2000-10-01</publish_date>
                <description>An in-depth look at creating applications
                with XML.</description>
        </book>
        <book id="bk102">
                <author>Ralls, Kim</author>
                <title>Midnight Rain</title>
                <genre>Fantasy</genre>
                <price>5.95</price>
                <publish_date>2000-12-16</publish_date>
                <description>A former architect battles corporate zombies,
                an evil sorceress, and her own childhood to become queen
                of the world.</description>
        </book>
        <book id="bk103">
                <author>Corets, Eva</author>
                <title>Maeve Ascendant</title>
                <genre>Fantasy</genre>
                <price>5.95</price>
                <publish_date>2000-11-17</publish_date>
                <description>After the collapse of a nanotechnology
                society in England, the young survivors lay the
                foundation for a new society.</description>
        </book>
        <book id="bk104">
                <author>Corets, Eva</author>
                <title>Oberon's Legacy</title>
                <genre>Fantasy</genre>
                <price>5.95</price>
                <publish_date>2001-03-10</publish_date>
                <description>In post-apocalypse England, the mysterious
                agent known only as Oberon helps to create a new life
                for the inhabitants of London. Sequel to Maeve
                Ascendant.</description>
        </book>
        <book id="bk105">
                <author>Corets, Eva</author>
                <title>The Sundered Grail</title>
                <genre>Fantasy</genre>
                <price>5.95</price>
                <publish_date>2001-09-10</publish_date>
                <description>The two daughters of Maeve, half-sisters,
                battle one another for control of England. Sequel to
                Oberon's Legacy.</description>
        </book>
        <book id="bk106">
                <author>Randall, Cynthia</author>
                <title>Lover Birds</title>
                <genre>Romance</genre>
                <price>4.95</price>
                <publish_date>2000-09-02</publish_date>
                <description>When Carla meets Paul at an ornithology
                conference, tempers fly as feathers get ruffled.</description>
        </book>
        <book id="bk107">
                <author>Thurman, Paula</author>
                <title>Splish Splash</title>
                <genre>Romance</genre>
                <price>4.95</price>
                <publish_date>2000-11-02</publish_date>
                <description>A deep sea diver finds true love twenty
                thousand leagues beneath the sea.</description>
        </book>
        <book id="bk108">
                <author>Knorr, Stefan</author>
                <title>Creepy Crawlies</title>
                <genre>Horror</genre>
                <price>4.95</price>
                <publish_date>2000-12-06</publish_date>
                <description>An anthology of horror stories about roaches,
                centipedes, scorpions  and other insects.</description>
        </book>
        <book id="bk109">
                <author>Kress, Peter</author>
                <title>Paradox Lost</title>
                <genre>Science Fiction</genre>
                <price>6.95</price>
                <publish_date>2000-11-02</publish_date>
                <description>After an inadvertant trip through a Heisenberg
                Uncertainty Device, James Salway discovers the problems
                of being quantum.</description>
        </book>
        <book id="bk110">
                <author>O'Brien, Tim</author>
                <title>Microsoft .NET: The Programming Bible</title>
                <genre>Computer</genre>
                <price>36.95</price>
                <publish_date>2000-12-09</publish_date>
                <description>Microsoft's .NET initiative is explored in
                detail in this deep programmer's reference.</description>
        </book>
        <book id="bk111">
                <author>O'Brien, Tim</author>
                <title>MSXML3: A Comprehensive Guide</title>
                <genre>Computer</genre>
                <price>36.95</price>
                <publish_date>2000-12-01</publish_date>
                <description>The Microsoft MSXML3 parser is covered in
                detail, with attention to XML DOM interfaces, XSLT processing,
                SAX and more.</description>
        </book>
        <book id="bk112">
                <author>Galos, Mike</author>
                <title>Visual Studio 7: A Comprehensive Guide</title>
                <genre>Computer</genre>
                <price>49.95</price>
                <publish_date>2001-04-16</publish_date>
                <description>Microsoft Visual Studio 7 is explored in depth,
                looking at how Visual Basic, Visual C++, C#, and ASP+ are
                integrated into a comprehensive development
                environment.</description>
        </book>
</catalog>
"""
```

We can then place a Table component and a Button component on the window, and place this script on the Button's actionPerformed event.

Python - Complete XML Parsing

```
# Start by importing the library
import xml.etree.ElementTree as ET

######
# Here is where you would paste in the document string.
# Simply remove this comment, and paste in the document string.
######

# We can then parse the string into useable elements.
root = ET.fromstring(document)

# This creates empty header and row lists that we will add to later.
# These are used to create the dataset that will go into the Table.
# We could fill in the names of the headers beforehand, since we know what each will be.
# However, this allows us to add or remove children keys, and the script will automatically adjust.
headers = []
rows = []

# Now we can loop through each child of the root.
# Since the root is catalog, each child element is an individual book.
# We also create a single row empty list. We can add all of the data for a single book to this list.
for child in root:
        oneRow = []

        # Check if the book has any attributes.
        if child.attrib != {}:

                # If it does contain attributes, we want to loop through all of them.
                for key in child.attrib:

                        # Since we only want to add the attributes to our header list once, first check
if it is there.
                        # If it isn't add it.
                        if key not in headers:
                                headers.append(key)

                        # Add the attribute value to the oneRow list.
                        oneRow.append(child.attrib[key])

        # Loop through the children of the book.
        for child2 in child:

                # Similar to above, we check if the tag is present in the header list before adding it.
                if child2.tag not in headers:
                        headers.append(child2.tag)

                # We can then add the text of the Element to the oneRow list.
                oneRow.append(child2.text)
        # Finally, we want to add the oneRow list to our list of rows.
        rows.append(oneRow)

# Once the loop is complete, this will print out the headers and rows list so we can manually check them
in the console.
print headers
print rows

# Convert to a dataset, and insert into the Table.
data = system.dataset.toDataSet(headers, rows)
event.source.parent.getComponent('Table').data = data
```

Related Topics ...

- system.file.openFile
- Libraries

Audit Log and Profiles

Ignition's built-in auditing system automatically records certain actions that occur in the system, such as a tag writes or User Source authentication, into a SQL database table. Utilizing the system involves creating an Audit Profile, followed by enabling auditing in a project. Once both prerequisites have been met, the Gateway will automatically create a database table named AUDIT_EVENTS, and use the table to start tracking user actions.

On this page ...

The following feature is new in Ignition version **8.0.7**
Click here to check out the other new features

Remote Audit Log configuration option was introduced in 8.0.7 allowing audit events to be automatically sent to a remote gateway's audit log. The remote gateway you plan to connect to must have a Audit Profile created. To learn more about sending audit events to a remote gateway, refer to section Creating a Remote Gateway Audit Profile on this page.

The following feature is new in Ignition version **8.0.10**
Click here to check out the other new features

Ignition version 8.0.10 introduced an Internal Audit Profile option allowing an Ignition Gateway to store audit records without an external SQL database. The only way to interact with the Internal Audit Profile is via the Status page of the Gateway webpage. There are only two settings for the Internal Audit Profile type: Name and Description (which are identical to those same properties on the Database audit profile type).

Auditing Actions

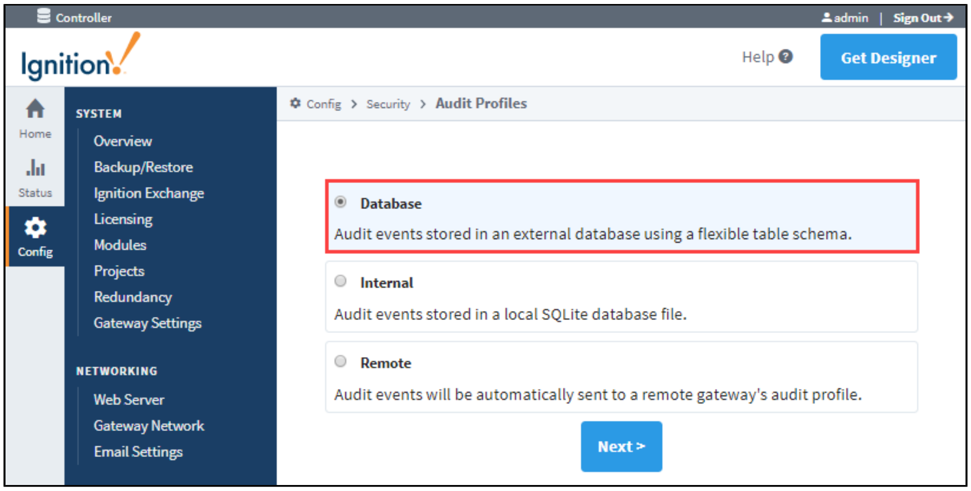
for a list of actions that are recorded by an audit profile, see the Auditing Actions Reference page.

Creating an Audit Profile for a Database

1. Go to the **Config** section of the Gateway.
2. Choose **Security > Auditing** from the menu on the left. The Audit Profiles page is displayed.
3. Click the **Create a new Audit Profile** link.

The following feature is new in Ignition version **8.0.7**
Click here to check out the other new features

4. You have the option of storing audit logs into an external database or sending them to a remote gateway. In this example, select **Database**. (Configuring audit events to be sent to a remote gateway's audit log is addressed in Creating a Remote Gateway Audit Profile section on this page).

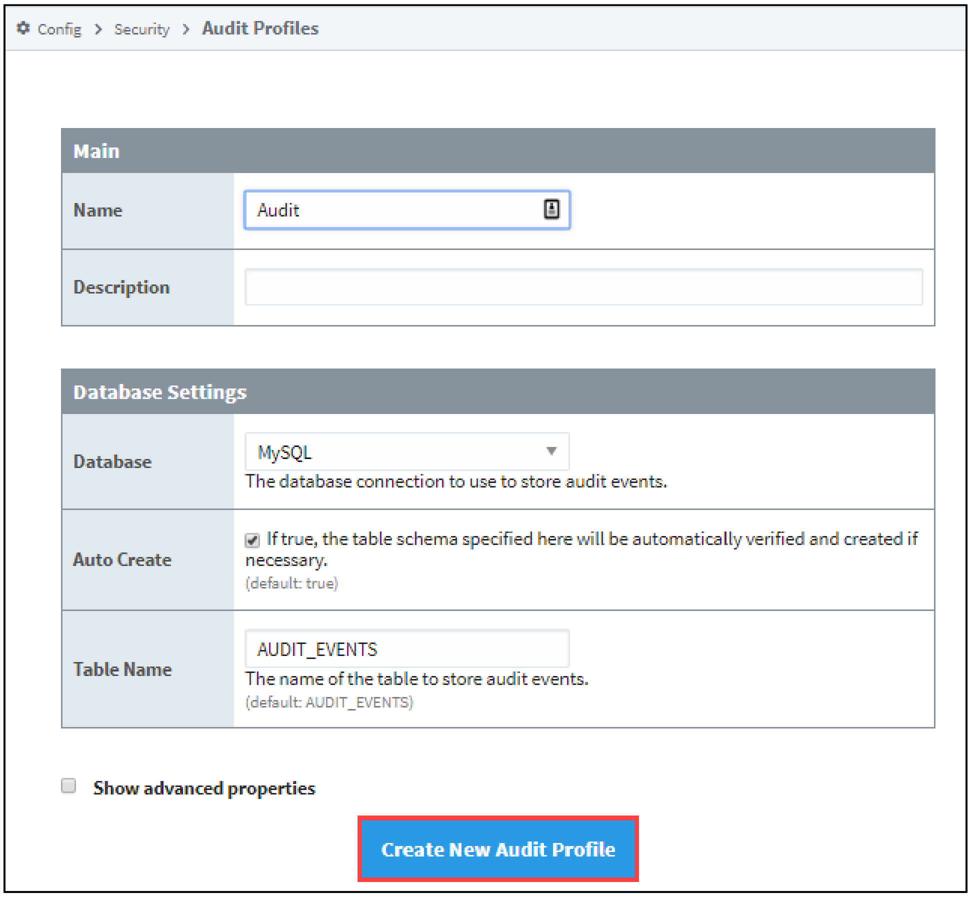


● Database
Audit events stored in an external database using a flexible table schema.

○ Internal
Audit events stored in a local SQLite database file.

○ Remote
Audit events will be automatically sent to a remote gateway's audit profile.

5. Enter the **Name** of the audit log and **Description** (optional).
6. Under the **Database Settings**, select the **Database** where the table will be stored, select the **Auto Create** check box, and enter the desired **Table Name**.
7. Click **Create New Audit Profile**.



Main

Name: Audit

Description:

Database Settings

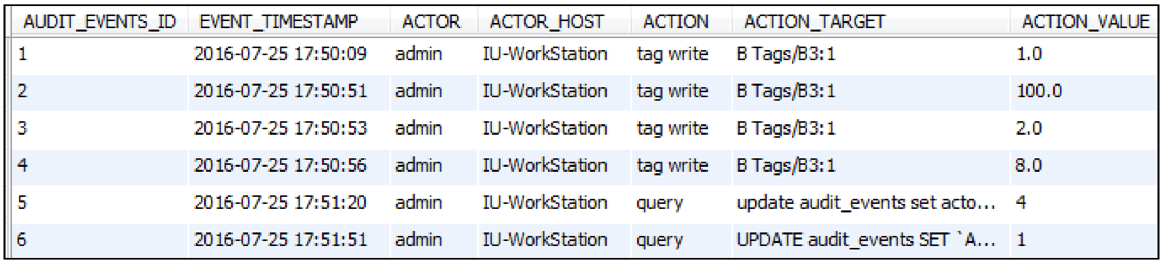
Database: MySQL
The database connection to use to store audit events.

Auto Create: ☑ If true, the table schema specified here will be automatically verified and created if necessary.
(default: true)

Table Name: AUDIT_EVENTS
The name of the table to store audit events.
(default: AUDIT_EVENTS)

☐ Show advanced properties

Once some changes have been made to a Tag or a Database table, Ignition will begin recording.

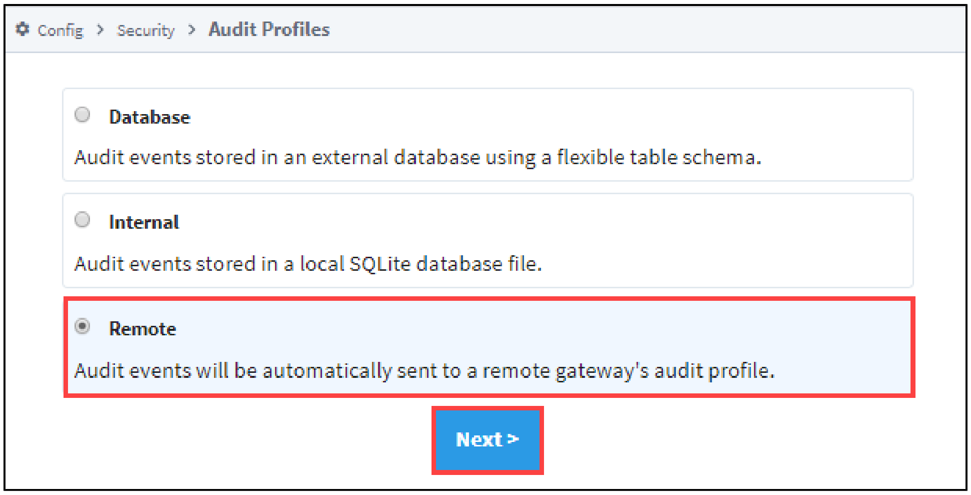


AUDIT_EVENTS_ID	EVENT_TIMESTAMP	ACTOR	ACTOR_HOST	ACTION	ACTION_TARGET	ACTION_VALUE
1	2016-07-25 17:50:09	admin	IU-WorkStation	tag write	B Tags/B3:1	1.0
2	2016-07-25 17:50:51	admin	IU-WorkStation	tag write	B Tags/B3:1	100.0
3	2016-07-25 17:50:53	admin	IU-WorkStation	tag write	B Tags/B3:1	2.0
4	2016-07-25 17:50:56	admin	IU-WorkStation	tag write	B Tags/B3:1	8.0
5	2016-07-25 17:51:20	admin	IU-WorkStation	query	update audit_events set acto...	4
6	2016-07-25 17:51:51	admin	IU-WorkStation	query	UPDATE audit_events SET `A...	1

Creating a Remote Gateway Audit Profile

Just like configuring audit events to be logged into an external database, it is done from the Gateway webpage, **Config > Security > Auditing.**

1. To have your audit events automatically sent to a remote gateway's audit profile, select **Remote**, and click **Next**.



Config > Security > **Audit Profiles**

○ **Database**

Audit events stored in an external database using a flexible table schema.

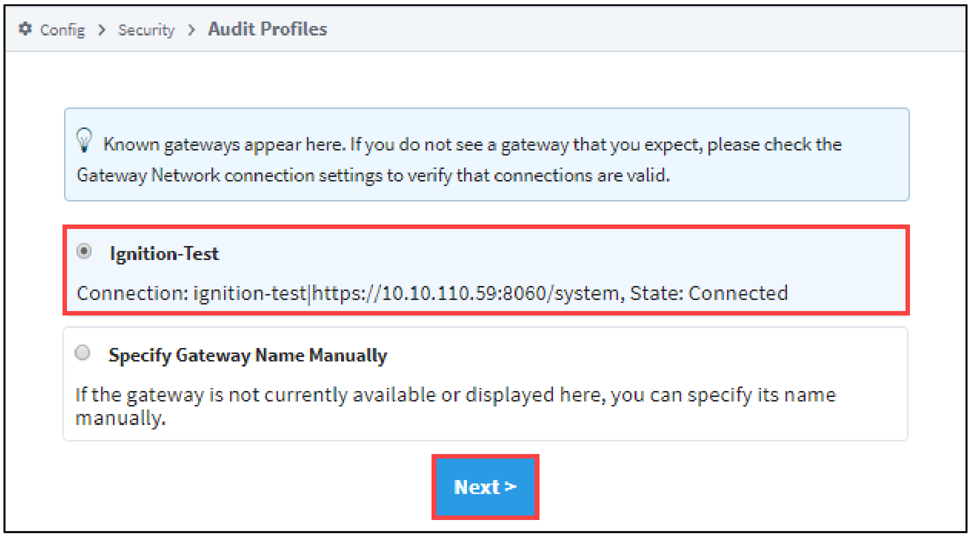
○ **Internal**

Audit events stored in a local SQLite database file.

● **Remote**

Audit events will be automatically sent to a remote gateway's audit profile.

2. A list of known gateways will be displayed. If you don't see a gateway that you expected to see, check your Gateway Network settings to verify that the connections are valid. You also have the option to specify a gateway manually. This example selects a valid gateway. Click **Next**.



Config > Security > **Audit Profiles**

Known gateways appear here. If you do not see a gateway that you expect, please check the Gateway Network connection settings to verify that connections are valid.

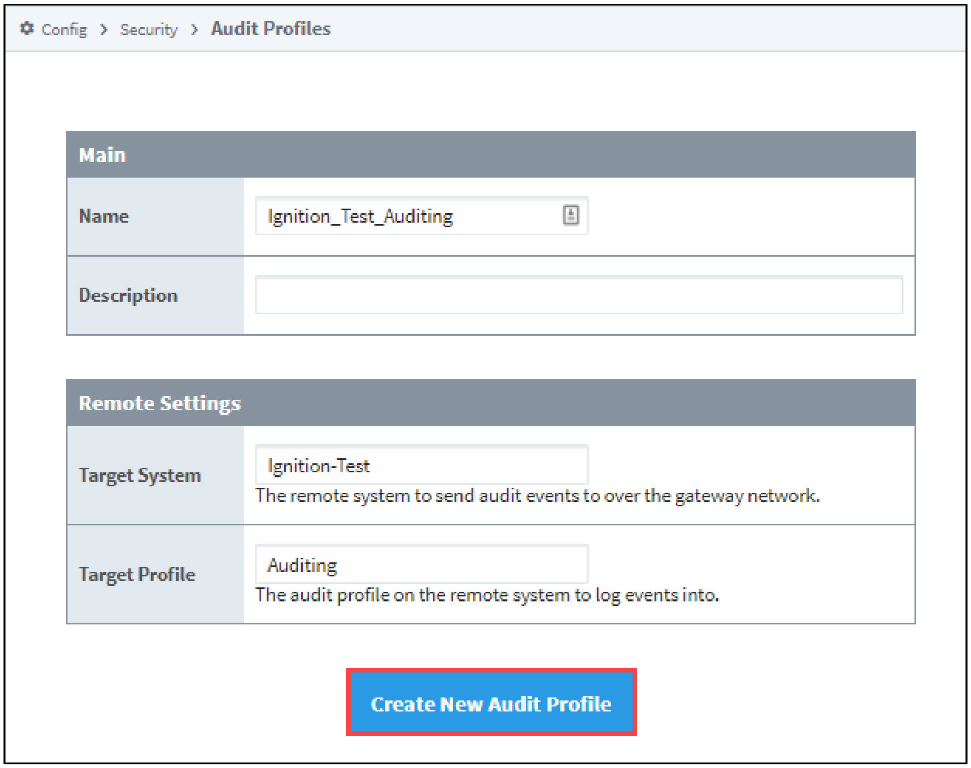
● **Ignition-Test**

Connection: ignition-test|https://10.10.110.59:8060/system, State: Connected

○ **Specify Gateway Name Manually**

If the gateway is not currently available or displayed here, you can specify its name manually.

3. If an Audit profile exists, the fields will auto-populate. The name of the gateway will appear in the **Name** field prefaced with the audit profile name,(i.e., Ignition_Test_Auditing), as shown in the following example. Click **Create New Audit Profile**.



Config > Security > Audit Profiles

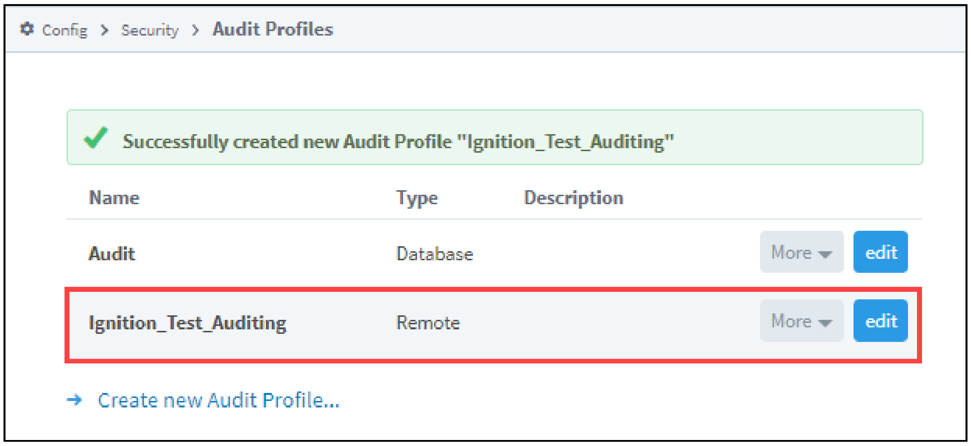
Main

Name	Ignition_Test_Auditing
Description	

Remote Settings

Target System	Ignition-Test The remote system to send audit events to over the gateway network.
Target Profile	Auditing The audit profile on the remote system to log events into.

4. You will receive a successful message stating your new Audit Profile was created.



Config > Security > Audit Profiles

✓ Successfully created new Audit Profile "Ignition_Test_Auditing"

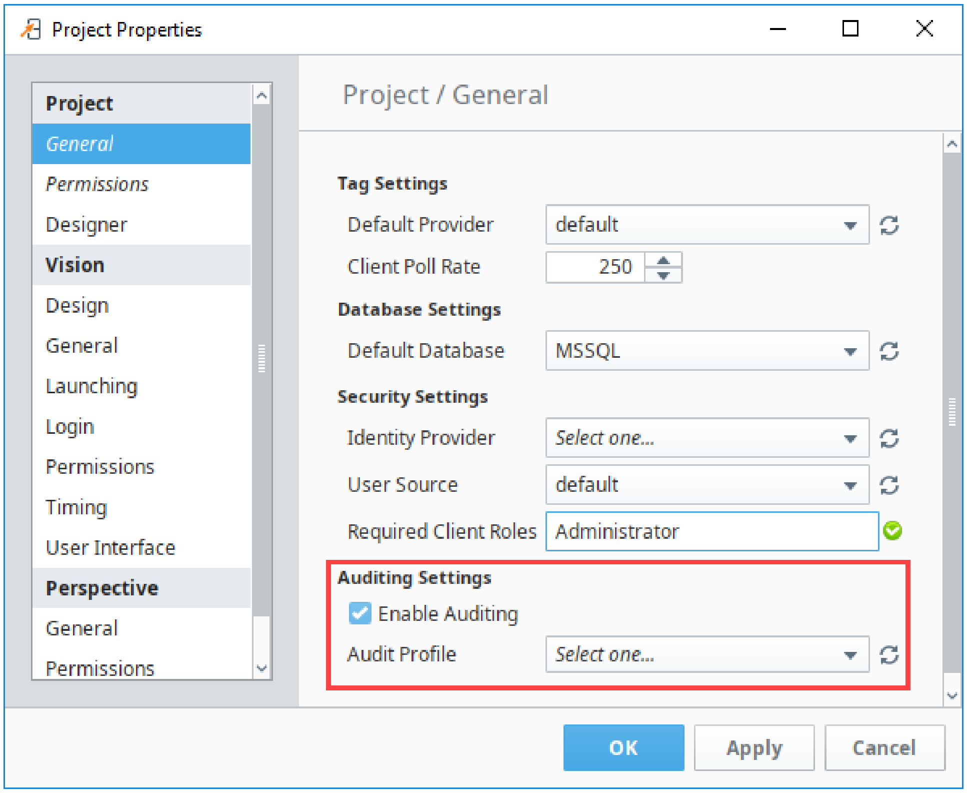
Name	Type	Description	
Audit	Database		More ▾ edit
Ignition_Test_Auditing	Remote		More ▾ edit

Remote Gateway Audit Profile Properties Table

Main	
Name	The default name, is the name of the Remote Gateway and Audit Profile.
Description	Description of the audit profile. Optional.
Enabled	By default. the journal profile is enabled.
Remote Settings	
Target System	The remote system to send audit events to over the gateway network.
Target Profile	The audit profile on the remote system to log events into.

Enabling Auditing in a Project

1. Go to the **Designer**, open the project that you want to enable auditing on, and go to **Project > Properties**.
2. Go to the **General** section, select the **Enable Auditing** check box, and select your **Audit Profile** from the drop-down menu. The audit profile is used to record audit actions for your project. If the new audit profile does not show up, click **Refresh**.
3. Click **OK**.
4. **Save** your Project.



Project / General

Tag Settings

Default Provider: default

Client Poll Rate: 250

Database Settings

Default Database: MSSQL

Security Settings

Identity Provider: *Select one...*

User Source: default

Required Client Roles: Administrator

Auditing Settings

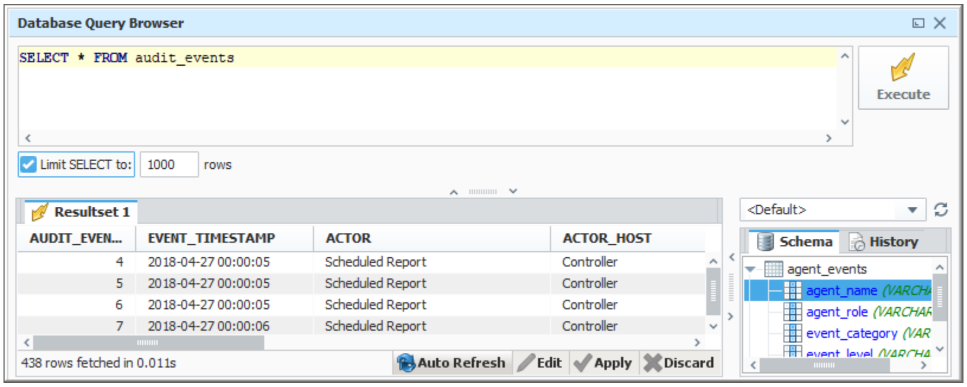
☑ Enable Auditing

Audit Profile: *Select one...*

Viewing Information in an Audit Log

There are a few ways to view audit information: using a Table component, interface on the Gateway, or the Database Query Browser. Here is one example of viewing an Audit Log using the Database Query Browser.

1. Go to **Tools > Database Query Browser.**
2. Under the **Schema** area, double click on a **table**, and it will expand the query in the Database Query Browser area.
3. Click **Execute,** and all the audit log data will be displayed in the Resultset1 area.



```
SELECT * FROM audit_events
```

☑ Limit SELECT to: 1000 rows

AUDIT_EVEN...	EVENT_TIMESTAMP	ACTOR	ACTOR_HOST
4	2018-04-27 00:00:05	Scheduled Report	Controller
5	2018-04-27 00:00:05	Scheduled Report	Controller
6	2018-04-27 00:00:05	Scheduled Report	Controller
7	2018-04-27 00:00:06	Scheduled Report	Controller

438 rows fetched in 0.011s

agent_events
- agent_name (VARCHA
- agent_role (VARCHAR
- event_category (VAR
- event_level (VARCHA

Related Topics ...

- Database Query Browser
- Audit Log Display

In This Section ...

Alarm Notification Auditing

Alarm Notification profiles can be set to store information in an Audit Profile.

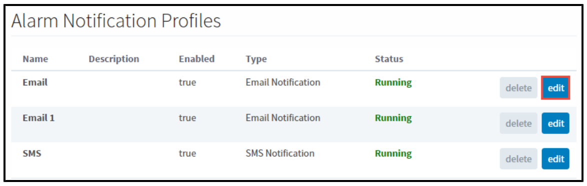
Alarm Notification Auditing

Watch the Video

How to Store Alarm Notifications in an Audit Profile

Once you have an audit profile created in Ignition, you can configure your Alarm Notification Profile to start using it.

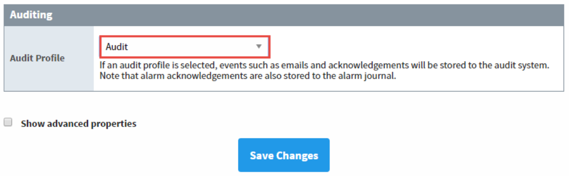
1. Go to the **Config** section of the Gateway.
2. Choose **Alarming > Notification** from the menu on the left.
   The Alarm Notification Profiles page is displayed.
3. Edit the appropriate notification profile by clicking the **edit** link.



Alarm Notification Profiles

Name	Description	Enabled	Type	Status	
Email		true	Email Notification	Running	delete edit
Email 1		true	Email Notification	Running	delete edit
SMS		true	SMS Notification	Running	delete edit

4. The Edit Alarm Notification Profile page is displayed.
   Scroll down to the **Auditing** section, and select an **Audit Profile** from the drop-down menu.
   Click **Save Changes**, and Ignition will automatically begin storing information.



Auditing

Audit Profile: Audit

If an audit profile is selected, events such as emails and acknowledgements will be stored to the audit system. Note that alarm acknowledgements are also stored to the alarm journal.

□ Show advanced properties

Now that the Alarm Notification Profile is storing data into the Audit system, you have a complete log of all alarm emails and acknowledgements that you can review.
See Audit Log Display for more info on how to retrieve information from the Audit Log.

Related Topics ...

- Alarm Notification
- Database Connections
- Audit Log Display

Audit Log Display

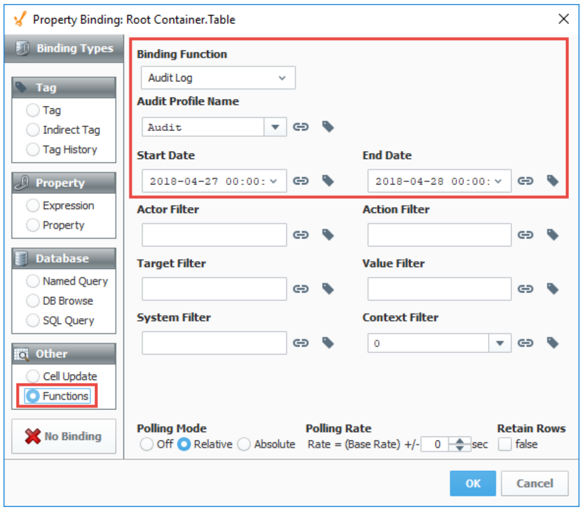
This page documents several ways to view results from the Audit Log System. For more information on how the Audit Log works, see the Audit Log and Profiles page.

Access the Audit Log with Table Functions

Since Ignition makes accessing data from databases seamless, it is possible to bind a data property of a table directly to database table. Alternatively, it is possible to access the contents of the audit log with table functions.

Once you have an Audit Log setup and attached to a project, you can go back to that project and see what information is in that audit log. This example uses the Table Functions to extract data from the Audit Log.

1. In **Designer**, drag a **Table** component on to a window.
2. In the **Property Editor**, click on the binding icon from the table's **Data** property. The Property Binding window is displayed.
3. Select the **Functions** Binding Type.
4. Select **Functions** from the various binding options.
    a. **Binding Function**, select the **Audit Log** from the drop-down menu.
    b. **Audit Profile Name**, select **Audit** or the name of your Audit Profile.
    c. **Start Date**, enter the appropriate start date.
    d. **End Date**, enter the appropriate end date.
    e. **Polling Mode**, select **Relative**.
    f. Select any other appropriate function options from the menu, and click **OK**.



Property Binding: Root Container.Table

Binding Types

Tag: ○ Tag ○ Indirect Tag ○ Tag History

Property: ○ Expression ○ Property

Database: ○ Named Query ○ DB Browse ○ SQL Query

Other: ○ Cell Update ● Functions

No Binding

Binding Function: Audit Log

Audit Profile Name: Audit

Start Date: 2018-04-27 00:00:

End Date: 2018-04-28 00:00:

Actor Filter:

Action Filter:

Target Filter:

Value Filter:

System Filter:

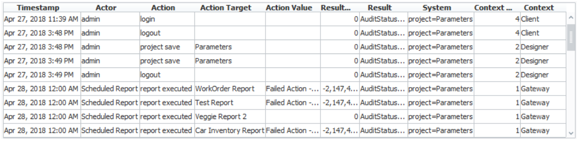
Context Filter: 0

Polling Mode: ○ Off ● Relative ○ Absolute

Polling Rate: Rate = (Base Rate) +/- 0 sec

Retain Rows: □ false

5. The table will populate with all the information stored in the audit log based on the Table Functions you selected. You can use the Table Customizer to configure how you want the table to look by reorganizing and hiding columns, making columns sortable, and assigning meaningful headers.



Timestamp	Actor	Action	Action Target	Action Value	Result...	Result	System	Context ...	Context
Apr 27, 2018 11:39 AM	admin	login			0	AuditStatus...	project=Parameters	4	Client
Apr 27, 2018 3:48 PM	admin	logout			0	AuditStatus...	project=Parameters	4	Client
Apr 27, 2018 3:48 PM	admin	project save	Parameters		0	AuditStatus...	project=Parameters	2	Designer
Apr 27, 2018 3:49 PM	admin	project save	Parameters		0	AuditStatus...	project=Parameters	2	Designer
Apr 27, 2018 3:49 PM	admin	logout			0	AuditStatus...	project=Parameters	2	Designer
Apr 28, 2018 12:00 AM	Scheduled Report	report executed	WorkOrder Report	Failed Action -...	-2,147,4...	AuditStatus...	project=Parameters	1	Gateway
Apr 28, 2018 12:00 AM	Scheduled Report	report executed	Test Report	Failed Action -...	-2,147,4...	AuditStatus...	project=Parameters	1	Gateway
Apr 28, 2018 12:00 AM	Scheduled Report	report executed	Veggie Report 2		0	AuditStatus...	project=Parameters	1	Gateway
Apr 28, 2018 12:00 AM	Scheduled Report	report executed	Car Inventory Report	Failed Action -...	-2,147,4...	AuditStatus...	project=Parameters	1	Gateway

INDUCTIVE UNIVERSITY

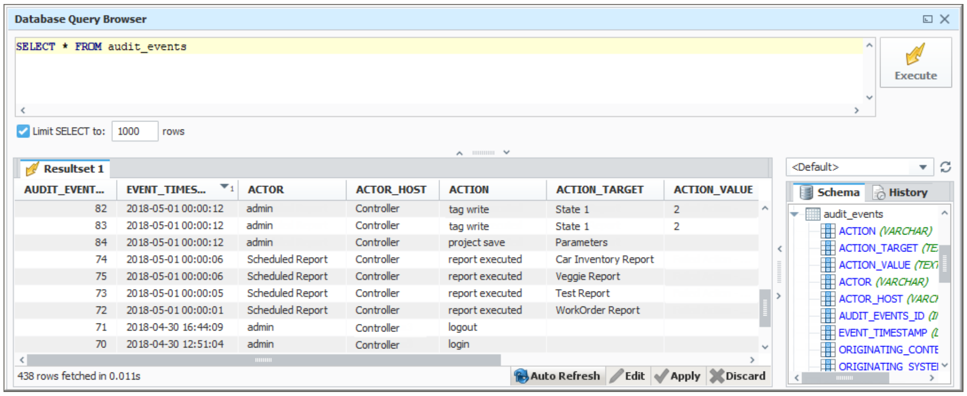
View Audit Information

Watch the Video

Access the Audit Log Using the Database Query Browser

The Database Query Browser makes it easy to search your database tables to view audit information.

1. In the **Designer** under the **Tools** menubar, select **Database Query Browser**. The Database Query Browser opens.
2. On the right side of Browser window under **Schema**, will be a list of tables from the currently selected database. Double click on the **audit_events** table, and click **Execute.**
   The data from the audit_events table will appear in the Resultset 1 tab.



Database Query Browser

```
SELECT * FROM audit_events
```

☑ Limit SELECT to: 1000 rows

Resultset 1

AUDIT_EVENT...	EVENT_TIMES...	ACTOR	ACTOR_HOST	ACTION	ACTION_TARGET	ACTION_VALUE
82	2018-05-01 00:00:12	admin	Controller	tag write	State 1	2
83	2018-05-01 00:00:12	admin	Controller	tag write	State 1	2
84	2018-05-01 00:00:12	admin	Controller	project save	Parameters	
74	2018-05-01 00:00:06	Scheduled Report	Controller	report executed	Car Inventory Report	
75	2018-05-01 00:00:06	Scheduled Report	Controller	report executed	Veggie Report	
73	2018-05-01 00:00:05	Scheduled Report	Controller	report executed	Test Report	
72	2018-05-01 00:00:01	Scheduled Report	Controller	report executed	WorkOrder Report	
71	2018-04-30 16:44:09	admin	Controller	logout		
70	2018-04-30 12:51:04	admin	Controller	login		

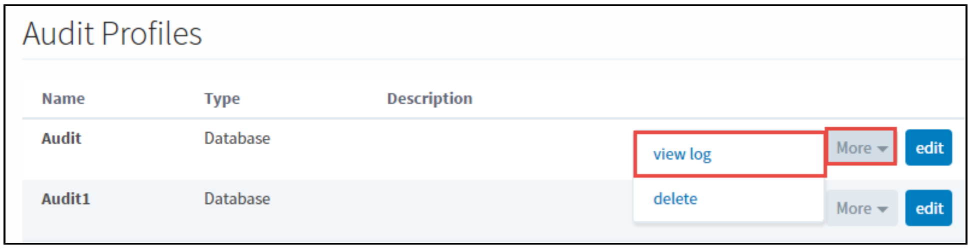
438 rows fetched in 0.011s

Refer to the Database Query Browser to learn more about how it works.

Access the Audit Log on the Gateway

Ignition provides a simple interface to view Audit Logs on the Gateway.

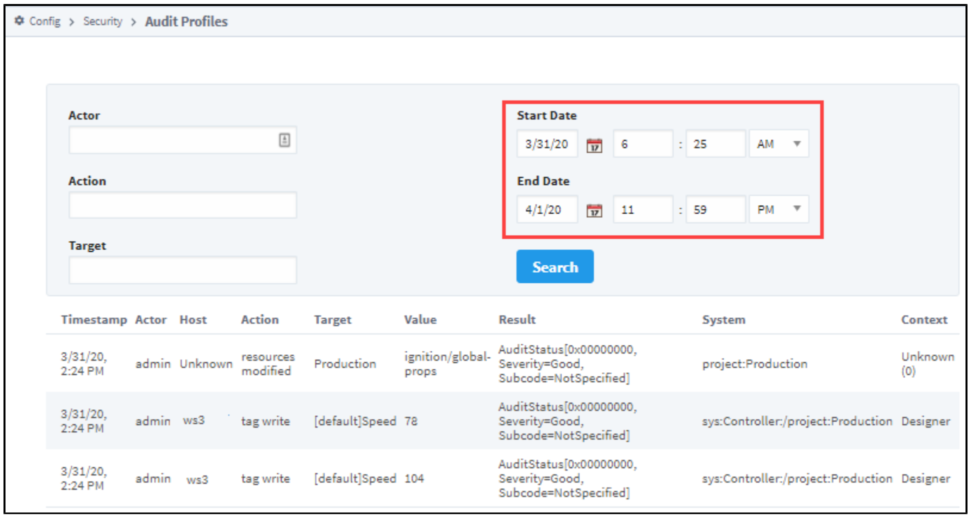
1. On the Gateway webpage in the **Config** section, scroll down to **Security > Auditing**.
2. The Audit Profile page will be displayed. Select the Audit Profile where your information is stored, and click **More > view log**.



Audit Profiles

Name	Type	Description
Audit	Database	
Audit1	Database	

3. Choose the parameter settings if you're looking for something specific, otherwise, enter a **Start Date** and **End Date**, and click **Search** .



Config > Security > Audit Profiles

Actor:

Action:

Target:

Start Date: 3/31/20 6 : 25 AM

End Date: 4/1/20 11 : 59 PM

Timestamp	Actor	Host	Action	Target	Value	Result	System	Context
3/31/20, 2:24 PM	admin	Unknown	resources modified	Production	ignition/global-props	AuditStatus[0x00000000, Severity=Good, Subcode=NotSpecified]	project:Production	Unknown (0)
3/31/20, 2:24 PM	admin	ws3	tag write	[default]Speed	78	AuditStatus[0x00000000, Severity=Good, Subcode=NotSpecified]	sys:Controller:/project:Production	Designer
3/31/20, 2:24 PM	admin	ws3	tag write	[default]Speed	104	AuditStatus[0x00000000, Severity=Good, Subcode=NotSpecified]	sys:Controller:/project:Production	Designer

Related Topics ...

- Scripting
- Database Query Browser
- Table Customizer
- Security